to helping acquire neighbors, Hello packets also act as keep-alives to let routers know that other routers are still functional.

Each router periodically sends an LSA to provide information on a router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly and the network's topology altered appropriately. From the topological database generated from LSAs, each router calculates a shortest-path tree, with itself as root. The shortest-path tree, in turn, yields a routing table.[11]

Border Gateway Protocol BGP performs interdomain routing in TCP/IP networks. BGP is an EGP, which means that it performs routing between multiple autonomous systems or domains and exchanges routing and reachability information with other BGP systems. BGP was developed to replace its predecessor, the now obsolete EGP, as the standard exterior gateway-routing protocol used on the global Internet. BGP solves serious problems with EGP and scales to Internet growth more efficiently.

BGP performs three types of routing: interautonomous system routing, intra-autonomous system routing, and passthrough autonomous system routing. Interautonomous system routing occurs between two or more BGP routers in different autonomous systems. Peer routers in these systems use BGP to maintain a consistent view of the internetwork topology. BGP neighbors communicating between autonomous systems must reside on the same physical network. The Internet serves as an example of an entity that uses this type of routing because it is comprised of autonomous systems or administrative domains. Many of these domains represent the various institutions, corporations, and entities that make up the Internet. BGP is frequently used to provide path determination to provide optimal routing within the Internet.

Intra-autonomous system routing occurs between two or more BGP routers located within the same autonomous system. Peer routers within the same autonomous system use BGP to maintain a consistent view of the system topology. BGP also is used to determine which router will serve as the connection point for specific external autonomous systems. Once again, the Internet provides an example of interautonomous system routing. An organization, such as a university, could make use of BGP to provide optimal routing within its own administrative domain or autonomous system. The BGP protocol can provide both inter- and intra-autonomous system routing services.

---

[11]Cisco Systems. "Open Shortest Path First." A white paper from Cisco Systems, www.cisco.com.

Passthrough autonomous system routing occurs between two or more BGP peer routers that exchange traffic across an autonomous system that does not run BGP. In a passthrough autonomous system environment, the BGP traffic does not originate within the autonomous system in question and is not destined for a node in the autonomous system. BGP must interact with whatever intra-autonomous system routing protocol is being used to successfully transport BGP traffic through that autonomous system.

BGP Routing As with any routing protocol, BGP maintains routing tables, transmits routing updates, and bases routing decisions on routing metrics. The primary function of a BGP system is to exchange network-reachability information, including information about the list of autonomous system paths, with other BGP systems. This information can be used to construct a graph of autonomous system connectivity from which routing loops can be pruned and with which autonomous-system-level policy decisions can be enforced.

Each BGP router maintains a routing table that lists all feasible paths to a particular network. The router does not refresh the routing table, however. Instead, routing information received from peer routers is retained until an incremental update is received. BGP devices exchange routing information upon an initial data exchange and after incremental updates. When a router first connects to the network, BGP routers exchange their entire BGP routing tables. Similarly, when the routing table changes, routers send the portion of their routing table that has changed. BGP routers do not send regularly scheduled routing updates, and BGP routing updates advertise only the optimal path to a network.

BGP uses a single routing metric to determine the best path to a given network. This metric consists of an arbitrary unit number that specifies the degree of preference of a particular link. The BGP metric typically is assigned to each link by the network administrator. The value assigned to a link can be based on any number of criteria, including the number of autonomous systems through which the path passes, stability, speed, delay, or cost.[12]

Resource Reservation Protocol (RSVP) The *Resource Reservation Protocol* (RSVP) is a network control protocol that enables Internet applications to obtain special *qualities of service* (QoSs) for their data flows.

[12]Cisco Systems. "Border Gateway Protocol." A white paper from Cisco Systems. www.cisco.com, June 1999.

RSVP is not a routing protocol; instead, it works in conjunction with routing protocols and installs the equivalent of dynamic access lists along the routes that routing protocols calculate. RSVP occupies the place of a transport protocol in the OSI Model seven-layer protocol stack. The IETF is now working toward standardization through an RSVP working group. RSVP operational topics discussed in this chapter include data flows, QoS, session startup, reservation style, and soft state implementation.

In RSVP, a data flow is a sequence of messages that have the same source, destination (one or more), and QoS. QoS requirements are communicated through a network via a flow specification, which is a data structure used by internetwork hosts to request special services from the internetwork. A flow specification often guarantees how the internetwork will handle some of its host traffic.

RSVP supports three traffic types: best-effort, rate-sensitive, and delay-sensitive. The type of data-flow service used to support these traffic types depends on the QoS implemented. The following paragraphs address these traffic types and associated services. For information regarding QoS, refer to the appropriate section later in this chapter.

Best-effort traffic is traditional IP traffic. Applications include file transfers, such as mail transmissions, disk mounts, interactive logins, and transaction traffic. The service supporting best-effort traffic is called best-effort service. Rate-sensitive traffic is willing to give up timeliness for guaranteed rate. Rate-sensitive traffic, for example, might request 100 Kbps of bandwidth. If it actually sends 200 Kbps for an extended period, a router can delay traffic. Rate-sensitive traffic is not intended to run over a circuit-switched network; however, it usually is associated with an application that has been ported from a circuit-switched network (such as ISDN) and is running on a datagram network (IP).

An example of such an application is H.323 videoconferencing, which is designed to run on ISDN (H.320) or ATM (H.310) but is found on the Internet. H.323 encoding is a constant rate or nearly constant rate, and it requires a constant transport rate. The RSVP service supporting rate-sensitive traffic is called *guaranteed bit-rate service*. Delay-sensitive traffic is traffic that requires the timeliness of delivery and varies its rate accordingly. MPEG-II video, for example, averages about 3 to 7 Mbps depending on the amount of change in the picture. As an example, 3 Mbps might be a picture of a painted wall, although 7 Mbps would be required for a picture of waves on the ocean. MPEG-II video sources send key and delta frames. Typically, 1 or 2 key frames per second describe the whole picture, and 13 or 28 frames describe the change from the key frame. Delta frames are usually substantially smaller than key frames. As a result, rates vary quite a

bit from frame to frame. A single frame, however, requires delivery within a frame time or the codec is unable to do its job. A specific priority must be negotiated for delta-frame traffic. RSVP services supporting delay-sensitive traffic are referred to as controlled-delay service (nonreal time service) and predictive service (real-time service).

In the context of RSVP, QoS is an attribute specified in flow specifications that is used to determine the way in which data interchanges are handled by participating entities (routers, receivers, and senders). RSVP is used to specify the QoS by both hosts and routers. Hosts use RSVP to request a QoS level from the network on behalf of an application data stream. Routers use RSVP to deliver QoS requests to other routers along the path(s) of the data stream. In doing so, RSVP maintains the router and host state to provide the requested service.[13]

Transport Protocols

Figure 4-1 shows that the actual VoIP conversation takes place over a channel separate from the signaling channel. The following paragraphs describe RTP, the transport protocol.

Real-Time Protocol (RTP) RTP is the most popular of the VoIP transport protocols. It is specified in RFC 1889 under the title of "RTP: A Transport Protocol for Real Time Applications." This RFC describes both RTP and another protocol, RTCP. As the name would suggest, these two protocols are necessary to support real-time applications like voice and video. RTP operates on the layer above UDP, which does not avoid packet loss or guarantee the correct order for the delivery of packets. RTP packets overcome those shortcomings by including sequence numbers that help applications using RTP to detect lost packets and ensure packet delivery in the correct order.

RTP packets include a timestamp that gives the time when the packet is sampled from its source media stream. This timestamp assists the destination application to determine the synchronized playout to the destination user and to calculate delay and jitter, two very important detractors of voice quality. RTP does not have the capacity to correct delay and jitter, but it does provide additional information to a higher-layer application so that the application can make determinations as to how a packet of voice or data is best handled.

[13]Cisco Systems. "Resource Reservation Protocol." A white paper from Cisco Systems, www.ciscosystems.com, June 1999.

RTCP provides a number of messages that are exchanged between session users and that provide feedback regarding the quality of the session. The type of information includes details such as the numbers of lost RTP packets, delays, and interarrival jitter. As voice packets are transported in RTP packets, RTCP packets transfer quality feedback. Whenever an RTP session opens, an RTCP session is also opened. That is, when a UDP port number is assigned to an RTP session for the transfer of media packets, another port number is assigned for RTCP messages.

RTP Payloads RTP carries the digitally encoded voice by taking one or more digitally encoded voice samples and attaching an RTP header to provide RTP packets, which are made up of an RTP header and a payload of the voice samples. These RTP packets are sent to UDP, where a UDP header is attached. This combination then goes to IP where an IP header is attached and the resulting IP datagram is routed to the destination. At the destination, the headers are used to pass the packet up the stack to the appropriate application.

RTP Headers RTP carries the carried voice in a packet. The RTP payload is comprised of digitally coded samples. The RTP header is attached to this payload and the packet is sent to the UDP layer. The RTP header contains the necessary information for the destination to reconstruct the original voice sample.

RTP Control Protocol (RTCP) RTCP enables exchanges of control information between session participants with the goal of providing quality-related feedback. This feedback is used to detect and correct distribution issues. The combination of RTCP and IP multicast enables a network operator to monitor session quality. RTCP provides information on the quality of an RTP session. RTCP empowers network operators to obtain information about delay, jitter, and packet loss and to take corrective action where possible to improve quality.

IPv6

The previous discussion assumed the use of *Internet Protocol version 4* (IPv4), the predominant version of IP in use today. A new version, IPv6, is now coming on the market. The explosion of Internet addresses necessitates

the deployment of IPv6. IPv6 makes possible infinitely more addresses than IPv4. Enhancements offered by IPv6 over IPv4 include

- **Expanded address space** Each address is allocated 128 bits instead of 32 bits in IPv4.
- **Simplified header format** This enables easier processing of IP datagrams.
- **Improved support for headers and extensions** This enables greater flexibility for the introduction of new options.
- **Flow-labeling capability** This enables the identification of traffic flows for real-time applications.
- **Authentication and privacy** Support for authentication, data integrity, and data confidentiality are supported at the IP level rather than through separate protocols or mechanisms above IP.

Conclusion

This chapter addressed the building blocks of VoIP. It will be necessary in future chapters to understand many of the concepts contained in this chapter. Just as the PSTN and softswitch networks can be broken down into the three elements of access, switching, and transport, VoIP can be summarized as a study of three types of protocols: signaling, routing, and transport. The proper selection of VoIP signaling protocols for a network is an issue of almost religious proportions among network builders. Although protocols will continue to evolve and new protocols will emerge, those addressed in this chapter will constitute the predominant structure of softswitch architecture for the next few years.

Softswitch
Architecture for VoIP

Franklin D. Ohrtman, Jr.

McGraw-Hill
New York Chicago San Francisco Lisbon
London Madrid Mexico City Milan New Delhi
San Juan Seoul Singapore Sydney Toronto

The McGraw-Hill Companies

Cataloging-in-Publication Data is on file with the Library of Congress.

2 3 4 5 6 7 8 9 0 DOC/DOC 0 9 8 7 6 5 4

ISBN 0-07-140977-7

The sponsoring editor for this book was Marjorie Spencer and the production supervisor was Pamela A. Pelton. It was set in New Century Schoolbook by MacAllister Publishing Services, LLC.

Printed and bound by RR Donnelley.

CHAPTER 5

SIP: Alternative Softswitch Architecture?

If the worldwide *Public Switched Telephone Network* (PSTN) could be replaced overnight, the best candidate architecture, at the time of this writing, would be based on *Voice over IP* (VoIP) and the *Session Initiation Protocol* (SIP). Much of the VoIP industry has been based on offering solutions that leverage existing circuit-switched infrastructure (such as VoIP gateways that interface a *private branch exchange* [PBX] and an *Internet Protocol* [IP] network). At best, these solutions offer a compromise between circuit- and packet-switching architectures with resulting liabilities of limited features, expensive-to-maintain circuit-switched gear, and questionable *quality of service* (QoS) as a call is routed between networks based on those technologies. SIP is an architecture that potentially offers more features than a circuit-switched network.

SIP is a signaling protocol. It uses a text-based syntax similar to the *Hypertext Transfer Protocol* (HTTP) like that used in web addresses. Programs that are designed for the parsing of HTTP can be adapted easily for use with SIP. SIP addresses, known as SIP *uniform resource locators* (URLs) take the form of web addresses. A web address can be the equivalent of a telephone number in an SIP network. In addition, PSTN phone numbers can be incorporated into an SIP address for interfacing with the PSTN. An email address is portable. Using the proxy concept, one can check his or her email from any Internet-connected terminal in the world. Telephone numbers, simply put, are not portable. They only ring at one physical location. SIP offers a mobility function that can follow subscribers to whatever phone they are nearest to at a given time.

Like H.323, SIP handles the setup, modification, and teardown of multimedia sessions, including voice. Although it works with most transport protocols, its optimal transport protocol is the *Real Time Protocol* (RTP) (refer to Chapter 3, "Softswitch Architecture or 'It's the Architecture, Stupid!'" for more information on RTP). Figure 5-1 shows how SIP functions as a signaling protocol while RTP is the transport protocol for a voice conversation. SIP was designed as a part of the *Internet Engineering Task Force* (IETF) multimedia data and control architecture. It is designed to interwork with other IETF protocols such as the *Session Description Protocol* (SDP), RTP, and the *Session Announcement Protocol* (SAP). It is described in the IETF's RFC 2543. Many in the VoIP and softswitch industry believe that SIP will replace H.323 as the standard signaling protocol for VoIP.

SIP is part of the IETF standards process and is modeled upon other Internet protocols such as the *Simple Mail Transfer Protocol* (SMTP) and HTTP. It is used to establish, change, and tear down (end) calls between one or more users in an IP-based network. In order to provide telephony services, a number of different standards and protocols must come together—

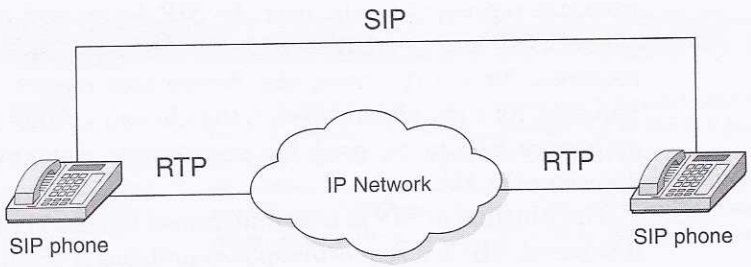


Figure 5-1
SIP is a signaling protocol and RTP transports the conversation protocol.

specifically to ensure transport (RTP), provide signaling with the PSTN, guarantee voice quality (*Resource Reservation Setup Protocol* [RSVP]), provide directories (*Lightweight Directory Access Protocol* [LDAP]), authenticate users (*Remote Access Dial-In User Service* [RADIUS]), and scale to meet anticipated growth curves.

What Is SIP?

SIP is focused on two classes of network entities: clients (also called *user agents* [UAs]) and servers. VoIP calls on SIP to originate at a client and terminate at a server. Types of clients in the technology currently available for SIP telephony include a PC loaded with a telephony agent or an SIP telephone. Clients can also reside on the same platform as a server. For example, a PC on a corporate *wide area network* (WAN) might be the server for the SIP telephony application, but it may also be used as a user's telephone (client).

SIP Architecture

SIP is a client-server architecture. The client in this architecture is the UA, which interacts with the user. It usually has an interface towards the user in the form of a PC or an IP phone (an SIP phone in this case). Four types of SIP servers exist. The type of SIP server used determines the architecture of the network. Those servers are the *user agent server* (UAS), the redirect server, the proxy server, and a registrar.

SIP Calls via a UA Server A UA server accepts SIP requests and contacts the user. A response from the user to the UA server results in an SIP

response representing the user. An SIP device will function as both a *UA client* (UAC) and as a UA server. As a UAC, the SIP device can initiate SIP requests. As a UA server, the device can receive and respond to SIP requests. As a standalone device, the UA can initiate and receive calls that empowers SIP to be used for peer-to-peer communications. Figure 5-2 describes the UA server.

The function of SIP is best understood via the HTTP model upon which it is based. SIP is a request/response protocol. A client is an SIP entity that generates a request. A server is an SIP entity that receives requests and returns responses. When a web site is desired, the client generates a request by typing in the URL, such as www.mcgrawhill.com. The server upon which the web site is hosted responds with McGraw-Hill's web page. SIP uses the same procedure. The UA sending the request is known as a UAC, and the UA returning the response is the UAS. This exchange is known as an SIP transaction.

Per Figure 5-2, a call initiates with the UA of the calling party sending an INVITE command caller@righthere.org to the called party, callee@theotherside.com. The UA for the caller has translated the name for the called party into an IP address via a *Domain Name System* (DNS) query accessible via their own domain. The INVITE command is sent to an SIP *User Datagram Protocol* (UDP) port and contains information such as media format and the From, To, and Via information. The TRYING informational response (180) from the calling party's call agent is analogous to the Q.931 CALL PROCEEDING message used in the PSTN, indicating the call is being routed. In the direct call model, a TRYING response is unlikely, but for proxy and redirect models it is used to monitor call progress. SIP

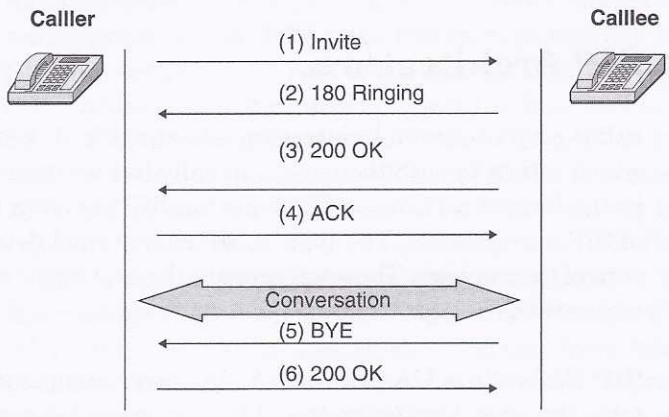


Figure 5-2
SIP UA server to UA server call

Table 5-1
SIP signaling methods

SIP method	Description
INVITE	The first message sent by a calling party to invite users to participate in a session. It contains information in the SIP header that identifies the calling party, call ID, called party, and call and sequence numbers. It indicates a call is being initiated. When a multiple choice of SDP parameters is offered, the ones chosen are returned with the success (200) code in the response message.
ACK	This is used to acknowledge the reception of a final response to an INVITE. A client originating an INVITE request issues an ACK request when it receives a final response for the INVITE, providing a three-way handshake.
OPTIONS	This queries a server about its capabilities, including which methods and which SDPs it supports. This determines which media types a remote user supports before placing the call.
BYE	This is used to abandon sessions. In two-party sessions, abandonment by one of the parties implies that the session is terminated. A return BYE from the other party is not necessary.
CANCEL	This method cancels pending transactions. The CANCEL method identifies the call via the call ID, call sequence, and To and From values in the SIP header.
REGISTER	These requests are sent by users to inform a server about their current location. SIP servers are co-located with SIP registrars. This enables the SIP server to find a user.

Source: Camarillo

uses six methods of signaling, as shown in Table 5-1. Additional methods are under consideration by the IETF at this time.

When the call arrives at the remote endpoint, the phone rings and a new response is sent to that endpoint: RINGING (180). This is analogous to the Q.931 ALERTING message used in the PSTN. The time between the user dialing the last digit and the time RINGING is received by the caller is known as the *postdial delay* (PDD) for SIP call setup. If a telephone number is involved in addressing the call, the numbers must be translated into an IP address. Table 5-2 provides a comparison of SIP and PSTN signals.

When the called party answers the phone, a 200 response is sent to the calling party's UA. The UA sends another request, ACK, acknowledging the response to the INVITE request. At that moment, media begins to flow on the transport addresses of the two endpoints. The ACK may carry the final SDP parameters for the media type and format provided by the receiving

Table 5-2

Comparison of SIP and PSTN signals

SIP	PSTN
TRYING	Q.931 Call Proceeding
RINGING	Q.931 Alerting
ACK	Q.931 Connect
XINVITE	Q.931 Connect

endpoint. The sequence of the INVITE and following ACK messages is similar to the Q.931 CONNECT message. ACKs do not require a response. Table 5-3 displays the response codes for SIP.

At this point in the call sequence, as seen in Figure 5-1, media flows over RTP, with the *Real-Time Control Protocol* (RTCP) providing the monitoring of the quality of the connection and its associated statistics. Next, as the name would imply, a BYE request from either party ends the call. As all messages are sent via UDP, no further action is required.

SIP Calls via a Proxy Server Proxy servers in the SIP sense are similar in function to proxy servers that serve a web site (mail relay via SMTP) for a corporate *local area network* (LAN). An SIP client in this case would send a request to the proxy server that would either handle it or pass it on to another proxy server that, after some translation, would take the call. The secondary servers would see the call as coming from the client. By virtue of receiving and sending requests, the proxy server is both a server and a client. Proxy servers function well in call forwarding and follow-me services.

Proxy servers can be classified by the amount of state information they store in a session. SIP defines three types of proxy servers: call stateful, stateful, and stateless. Call stateful proxies need to be informed of all the SIP transactions that occur during the session and are always in the path taken by SIP messages traveling between end users. These proxies store state information from the moment the session is established until the moment it ends. A stateful proxy is sometimes called a transaction stateful proxy as the transaction is its sole concern. A stateful proxy stores a state related to a given transaction until the transaction concludes. It does not need to be in the path taken by the SIP messages for subsequent transactions. Forking proxies are good examples of stateful proxies. Forking proxies are used when a proxy server tries more than one location for the user; that is, it "forks" the invitation.

Table 5-3
SIP response codes

Number code	Description	Number code	Description
100	Trying	413	Request, entity too large
180	Ringing	414	Request, URI too large
181	Call is being forwarded	415	Unsupported media type
182	Queued	420	Bad extension
183	Session progress	480	Temporarily not available
200	OK	481	Call leg/transaction does not exist
202	Accepted	482	Loop detected
300	Multiple choices	483	Too many hops
301	Moved permanently	484	Address incomplete
302	Moved temporarily	485	Ambiguous
305	Use proxy	486	Busy here
380	Alternative service	487	Request cancelled
400	Bad request	488	Not acceptable here
401	Unauthorized	500	Internal severe error
402	Payment required	501	Not implemented
403	Forbidden	502	Bad gateway
404	Not found	503	Service unavailable
405	Method not allowed	504	Gateway timeout
406	Not acceptable	505	SIP version not supported
407	Proxy authentication	600	Busy everywhere
408	Request timeout	603	Decline
409	Conflict	604	Does not exist anywhere
410	Gone	606	Not acceptable
411	Length required		

Source: Camarillo

Stateless proxies keep with no state. They receive a request, forward it to the next hop, and immediately delete all states related to that request. When a stateless proxy receives a response, it determines routing based solely on the Via header and it does not maintain a state for it.[1]

An SIP call using a proxy server is a little more complicated than the simple SIP call model described previously (see Figure 5-3). In this call, the caller is configured with the called party's SIP server. An INVITE is sent to the called party's SIP server with the called party's text address in the To field. The called party's server determines if the called party is registered in that server. If the called party is registered on that server, it then determines the called party's current location on the network. This is called the mobility feature.

Once the called party is located via the mobility feature, the proxy server generates an INVITE request with no alteration of the headers of the request, except to add its own name in the Via field. Multiple servers may be involved in tracking down the called party.

Next the server must retain state information on the call. The server does this by correlating Cseq numbers, call IDs, and other elements of the headers as they pass through the proxy server. The server sends a TRYING message back to the calling party's agent. When the called party answers at the new location, a RINGING response is sent to the proxy server via the remote server (the called party's server). Both servers have Via entries in the response message to the calling party. Finally, ACK messages are

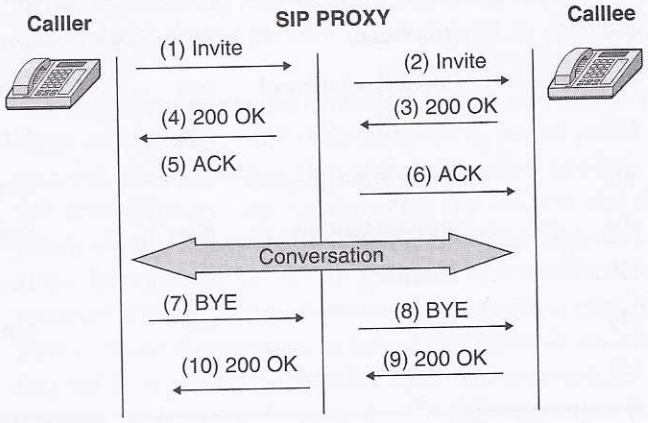


Figure 5-3
SIP call using a proxy server

[1]Camarillo, Gonzalo. *SIP Demystified*. New York: McGraw-Hill, 2002, p. 126–129.

exchanged, the call is established, and the media flow over RTP can begin. The call is terminated via a BYE request.

What makes the proxy server marketable is its user mobility feature. The called party can be logged in at multiple locations at once. This results in the proxy server generating the INVITE to all names on the list until the called party is found (preferably RINGING, but also TRYING or OK).[2]

A redirect server accepts SIP requests, maps the destination address to zero or more new addresses, and returns the translated address to the originator of the request. After that, the originator of the request can send requests to the addresses returned by the redirect server. The redirect server originates no requests of its own. Redirect servers pose an alternative means of call forwarding and follow-me services. What differentiates the redirect server from a proxy server is that the originating client redirects the call. The redirect server provides the intelligence to enable the originating client to perform this task as the redirect server is no longer involved.

The redirect server call model is a mix of the two previously described call models. Here a proxy model reverts to the direct call model once the called party is located. The redirect server returns a redirection response to the INVITE with code 301 or 302 indicating the called party is at the location listed in the Contact field of the message body. The calling party's *Media Gateway Controller* (MGC) closes its signaling with the redirect server and initiates another INVITE to the located returned in the redirect response. After that, the call flow is that of the direct model. If the called party is registered at a number of locations, the redirect server will return a list of names (URI) to be contacted. The calling party can then contact those addresses directly.

Registrar A registrar is a server that accepts SIP REGISTER requests. SIP includes the concept of user registration where a user tells the network that he is available at a given address. This registration occurs by issuing a REGISTER request by the user to the registrar. A registrar is often combined with a proxy or redirect server. Practical implementations often combine the UAC and UAS with registrars with either proxy servers or redirection servers. This can result in a network having only UAs and redirection or proxy servers.[3]

---

[2]Douskalis, Bill. *IP Telephony: The Integration of Robust VoIP Services*. New York: Prentice Hall, 2000, pg. 74.

[3]Collins, Daniel. *Carrier Grade Voice over IP*. New York: McGraw Hill, 2000, pp. 167–168.

Location Servers Location servers are not SIP entities but are an important part of SIP architecture. A location server stores and returns possible locations for users. It can make use of information from registrars or from other databases. Most registrars upload location updates to a location server upon receipt. SIP is not used between location servers and SIP servers. Some location servers use the *Lightweight Directory Access Protocol* (LDAP, see IETF RFC 1777) to communicate with SIP servers.[4]

New Standards for SIP

New standards efforts are aimed toward using the IP network for nonvoice interactions. Here are a few of the more exciting standards developments:

- ***PSTN to Internet Interworking* (PINT)** The PINT (RFC 2848) standards effort defines "click-to-dial" services—interworking between a web page and a PSTN gateway element, or PINT server. PINT is helping to standardize service delivery, including request to call, request to fax, request to hear content, and, in the future, request to conference.
- ***Services in the PSTN/IN Requesting Internet Services* (SPIRITS)** This working group is providing a framework for standardizing the mechanisms for controlling critical call altering and call-completion processes from the Internet by exposing the Internet domain to the PSTN's call model. The goal is to create a framework for Internet call-waiting-type services that will be applicable to wireline, wireless, and broadband (cable and *x-Type Digital Subscriber Line* [xDSL]) environments. Proposed SPIRITS-enabled network events include voice mail arrival, incoming call notification, attempts to dial numbers, dropping dialed connection, completing *Internet service provider* (ISP) connections, attempts to forward calls, and attempts to subscribe/unsubscribe to a PSTN service.
- ***Signaling Translation* (SIGTRAN)** The IETF standard that deals with *Signaling System 7* (SS7; specifically *Transaction Capabilities Application Part* or TCAP over IP). A later chapter will cover SIGTRAN in greater detail.
- ***Telephone Number Mapping* (ENUM)** The IETF standard that defines address mapping among phone numbers and Internet devices.

[4]Ibid., p.105.

It provides a way to reach multiple communication services via a single phone number. ENUM provides a process for converting a phone number into a DNS address (URL). It translates e.164 telephone numbers into Internet addresses.[5]

- ***Telephony Routing over IP* (TRIP)** TRIP is a policy-driven interadministrative domain protocol for advertising the reachability of telephony destinations between location servers and for advertising attributes of the routes to those destinations. TRIP's operation is independent of any signaling protocol; hence, TRIP can serve as the telephony routing protocol for any signaling protocol.[6]

Some SIP Configurations

SIP, given its simplicity and flexibility (discussed later in this chapter), simplifies the establishment of an alternative voice network. Figure 5-4 suggests a number of applications for SIP. The figure illustrates a corporate voice network that utilizes its corporate WAN to transport its interoffice voice traffic. This company can protect their investment in legacy phones and PBXs by installing an SIP gateway to interface the TDM technology with the SIP-based VoIP network. The SIP gateways also provide an interface with the PSTN.

In new offices and for remote workers, SIP phones provide a simple means of eliminating long-distance charges and connecting those workers to the rest of the network. Failing an expensive SIP phone, PC-based UAs such as those contained in Microsoft's XP (see the end of this chapter) can also function as a telephony agent connecting the worker to the network.

Providing the intelligence for this network is the SIP proxy server to control the calls. The application server provides critical corporate voice services such as voice mail, follow me, conferencing, call forwarding, unified messaging, voice-activated services, web-based provisioning, and new features that can be custom developed by the corporation, the SIP equipment vendor, or third-party vendors.

---

[5]Wolter, Charlotte. "ENUM Brings VoIP into Telephone Mainstream." *Sounding Board Magazine*, June 2001, p.12

[6]"Interworking Switched Circuit and Voice-over-IP Networks." A white paper by Performance Technologies and the International Engineering Consortium, available online at http://www.iec.org/online/tutorials/ip_in/topic05.html, pp. 5–6.

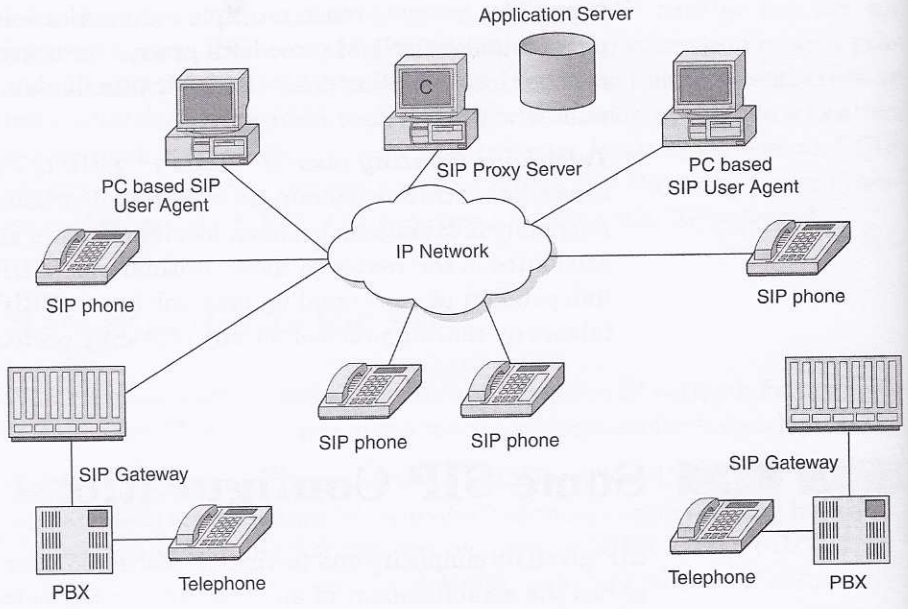


Figure 5-4
SIP gateways in long-distance bypass and enterprise telephony

Comparison of SIP to H.323

H.323 is a blanket specification, meaning that it is not a protocol by itself, but rather defines how to use other protocols in a specific manner to create a service. H.323 was developed by the *International Telecommunications Union* (ITU) in the mid-1990s and consists of several protocols, including H.225 *registration, admission, and status* (RAS) signaling, H.225.0 call signaling, H.245 control signaling, and RTP. It also includes several standards for voice and video digitization and compression.

H.323 was originally designed to implement multimedia conferences on a LAN. In its original form, it was intended that certain conferences would be announced, or advertised in advance, and interested parties would subscribe or register to participate in the conference. These conferences may be interactive or may be broadcast events with participants viewing or listening only.

The applications were expanded to include participants who were not located on the same LAN as well as more general calling capabilities. This resulted in more complexity to the protocol and underlying applications. H.323 is a very large protocol suite, requiring a large amount of memory for

code and data space. H.323 used a messaging scheme called *Abstract Syntax Notation* (ASN.1). ASN.1 is widely used by the ITU, but it is difficult for users to read and understand directly. The use of ASN.1 requires special test equipment to be built that can decode the ASN.1 messages and translate them into human-readable format for easier network testing and debugging.

The IETF sought to create a much simpler, yet powerful protocol that could be used for call setup and management in support of VoIP applications. A working group was formed, and the result of that effort was the development of the SIP, which was ratified by the IETF as RFC 2543 in 1999. H.323 and SIP have four fields of comparison: complexity, extensibility, scalability, and services.

Complexity of H.323 Versus SIP

As VoIP protocols evolve, they become more efficient. Simplicity fosters acceptance. In this case, SIP is a marked improvement over H.323 primarily due to its greater simplicity, which translates to greater reliability. Given its simplicity relative to H.323, SIP enjoys a faster call setup, a prerequisite for carrier-grade voice applications.

H.323 is a rather complex protocol. The sum total of the base specifications alone is 736 pages. SIP, on the other hand, along with its call control extensions and SDPs totals 276 pages in RFC 3261. H.323 defines hundreds of elements, while SIP has only 37 headers (32 in the base specification, 5 in the call control extensions), each with a small number of values and parameters, but that contain more information.

A basic but interoperable SIP Internet telephony implementation can get by with four headers (To, From, Call-ID, and CSeq) and three request types (IN-VITE, ACK, and BYE) and is small enough to be assigned as a homework programming problem. A fully functional SIP client agent with a *graphical user interface* (GUI) has been implemented in just two man-months.

H.323 uses a binary representation for its messages, based on ASN.1 and the *packed encoding rules* (PER). SIP, on the other hand, encodes its messages as text, similar to HTTP[7] and the *Real-Time Streaming Protocol*

[7]Fielding, R., J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee. Request for Comments (Proposed Standard) 2068: "Hypertext transfer protocol—HTTP/1.1." Internet Engineering Task Force, January 1997.

(RTSP).[8] This leads to simple parsing and generation, particularly when done with powerful text-processing languages such as Perl. The textual encoding also simplifies debugging, allowing for manual entry and the perusing of messages. Its similarity to HTTP also allows for code reuse; existing HTTP parsers can be quickly modified for SIP usage.[9]

One of the biggest criticisms of H.323 is that it can result in the transmission of many unnecessary messages across the network. This causes H.323 to not scale well. That is, for a large system, the overhead associated with handling a large number of messages has a significant effect on the system performance. A more heavily loaded system will result in poorer voice quality. QoS measures do not address network traffic and call setup issues in H.323.

H.323's complexity also stems from its use of several protocol components. There is no clean separation of these components; many services require interactions between several of them. (Call forward, for example, requires components of H.450, H.225.0, and H.245.) The use of several different protocols also complicates firewall traversal. Firewalls must act as application-level proxies,[10] parsing the entire message to arrive at the required fields. The operation is stateful since several messages are involved in call setup. SIP, on the other hand, uses a single request that contains all necessary information.

Figure 5-5 illustrates the messages required to set up a call, assuming that the caller already knows the address of the callee. Without going into the details of the messages, it can be seen that setting up the call requires the exchange of 16 messages across the network. This number grows considerably if the two terminals are located on different LANs or are managed by the Terminal Capability Set messages. This allows the two ends to agree on the parameters to be used for the call. In the case of SIP, this information is exchanged in the Invite and 200:OK messages (see Figure 5-6). Separate messages are not required. Likewise, in H.323, separate messages are exchanged to create logical connections between the two endpoints over which the voice information is passed (using the Open Logical Channel messages). In SIP, this information is also passed in the Invite and the 200:OK message.[11]

---

[8]Schulzrinne, H., R. Lanphier, and A. Rao. Request for Comments (Proposed Standard) 2326: "Real-time streaming protocol (RTSP)." Internet Engineering Task Force, April 1998.

[9]Schulzrenne, Henning, and Jonathan Rosenberg. "A Comparison of SIP and H.323 for Internet Telephony." White paper available at www.cs.columbia.edu/sip/papers.html.

[10]"Packet-Based Multimedia Communications Systems," H.323, ITU, February 1998.

[11]"SIP vs. H.323, A Business Analysis." White paper from Windriver, available at www.sipcenter.com/files/Wind_River_SIP_H323.pdf.

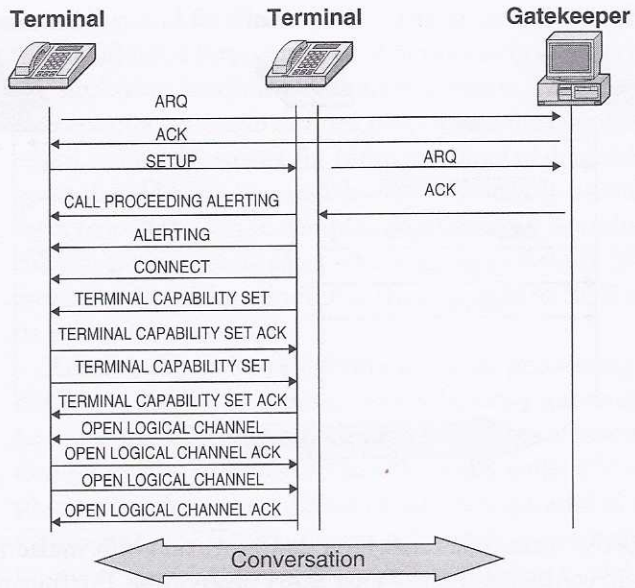


Figure 5-5
Call setup process for H.323

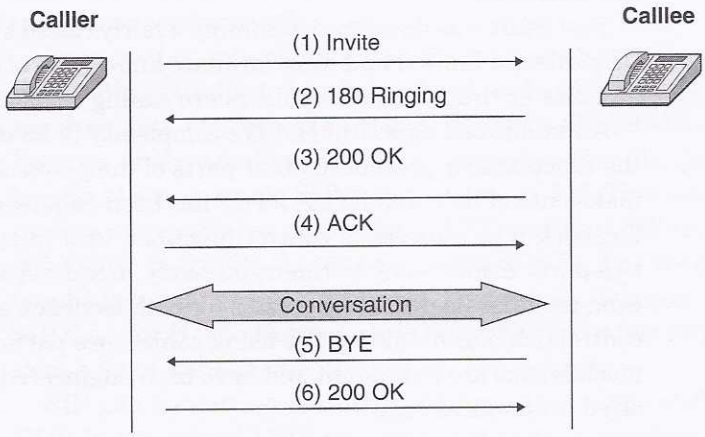


Figure 5-6
Call setup process for SIP

To counter this obvious weakness in H.323, procedures were added to enable H.323 to connect basic calls much quicker. This procedure is referred to as Fast Start and is illustrated in Figure 5-7. Fast Start eliminates several of the messages by requiring each side to have detailed knowledge of the parameters for the call and to have preassigned ports for communicating. This significantly limits the ability to make calls to multiple locations,

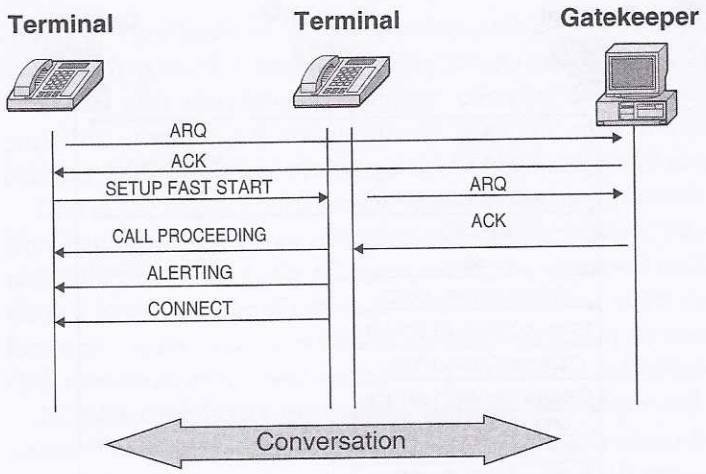


Figure 5-7
H.323 using Fast Start

as the terminals must have configuration information for all locations they will contact using the Fast Start procedures. Furthermore, not all terminals support the Fast Start procedures, so this further limits the usefulness of this procedure.

Fast Start was developed assuming a fairly closed system, where the participants are limited and have intimate knowledge of each other, such as in an office environment where users are calling each other.[12]

An additional aspect of H.323's complexity is its duplication of some of the functionality present in other parts of the protocol. In particular, H.323 makes use of RTP and RTCP. RTCP has been engineered to provide various feedback and conference control functions in a manner that scales from two-party conferences to thousand-party broadcast sessions. H.245, however, provides its own mechanisms for both feedback and simple conference control (such as obtaining the list of conference participants). These H.245 mechanisms are redundant and have been engineered for small to medium-sized conferences only.[13]

Scalability

H.323 and SIP differ in terms of scalability. The three main concerns are managing a large number of domains, server-processing capabilities, con-

[12]Ibid.
[13]Schulzrenne and Rosenberg.

ferencing, and feedback. H.323 was originally conceived for use on a single LAN and not a large number of domains. Issues such as wide area addressing and user location were not a concern. The newest version defines the concept of a zone and defines procedures for user locations across zones for email names. However, for large numbers of domains, and complex location operations, H.323 has scalability problems. It provides no easy way to perform loop detection in complex multidomain searches (it can be done statefully by storing messages, which is not scalable). SIP, however, uses a loop detection algorithm similar to the one used in BGP, which can be performed in a stateless manner.

Another factor for scalability is server processing. As regards server processing in an H.323 system, both telephony gateways and gatekeepers will be required to handle calls from a multitude of users. Similarly, SIP servers and gateways will need to handle many calls. For large, backbone IP telephony providers, the number of calls being handled by a large server can be significant. In SIP, a transaction through several servers and gateways can be either stateful or stateless. In the stateless model, a server receives a call request, performs some operation, forwards the request, and completely forgets about it. SIP messages contain sufficient state to enable the response to be forwarded correctly.[14]

Because the protocol is simpler, SIP requires less code to implement than H.323. This is reflected in lower fixed (code space) and dynamic memory requirements for both the protocol stack itself and the host application. Because of the smaller number of messages that the processor needs to handle to set up a call, SIP is faster than H.323. This means that SIP can process more calls per second than H.323 and that SIP requires a less powerful CPU to handle the same number of calls. The result is higher calls handled for a given system's usage. This results in larger message sizes for SIP than for H.323. However, it is better to send a few large frames of data than a lot of small frames except for slow links.[15]

SIP can be carried on either *Transmission Control Protocol* (TCP) or UDP. In the case of UDP, no connection state is required. This means that large, backbone servers can be based on UDP and operate in a stateless fashion, reducing significantly the memory requirements and improving scalability. H.323, on the other hand, requires gatekeepers (when they are in the call loop) to be stateful. They must keep the call state for the entire duration of a call. Furthermore, the connections are TCP based, which

[14]Ibid., pg. 3.
[15]"SIP vs. H.323, A Business Analysis."

means a gatekeeper must hold its TCP connections for the entire duration of a call. This can pose serious scalability problems for large gatekeepers.

Conferencing is another concern for scalability. H.323 supports multiparty conferences with multicast data distribution. However, it requires a central control point (called an MC) for processing all signaling, even for the smallest conferences. This presents several difficulties. Firstly, should the user providing the MC functionality leave the conference and exit their application, the entire conference terminates. In addition, since MC and gatekeeper functionality is optional, H.323 cannot support even three party conferences in some cases. We note that the MC is a bottleneck for larger conferences.

H.323 Version 2 has defined the concept of cascaded MCs, allowing for a very limited, application-layer, multicast distribution tree of control messaging. This improves scaling somewhat, but for even larger conferences, the H.332 protocol defines additional procedures. This means that three distinct mechanisms exist to support conferences of different sizes. SIP, however, scales to all different conference sizes. There is no requirement for a central MC; conference coordination is fully distributed. This improves scalability and complexity. Furthermore, as it can use UDP as well as TCP, SIP supports native multicast signaling, enabling a single protocol to scale from sessions with two to millions of members. Table 5-4 compares the SIP and H.323 call features.

Feedback is another concern when comparing H.323 and SIP. H.245 defines procedures that enable receivers to control media encodings, trans-

Table 5-4

Comparison of SIP and H.323 call features

Feature	SIP	H.323
Blind transfer	Yes	Yes
Operator-assisted transfer	Yes	No
Hold	Yes, through SDP	No
Multicast conferences	Yes	Yes
Multiunicast conferences	Yes	Yes
Bridged conferences	Yes	Yes
Forward	Yes	Yes
Call park	Yes	No
Directed call pickup	Yes	No

Source: Schulzrenne and Rosenberg

mission rates, and error recovery. This kind of feedback makes sense in point-to-point scenarios, but it ceases to be functional in multipoint conferencing. SIP, instead, relies on RTCP for providing feedback on reception quality (and also for obtaining group membership lists). RTCP, like SIP, operates in a fully distributed fashion. The feedback it provides automatically scales from a two-person point-to-point conference to huge broadcast-style conferences with millions of participants.[16]

Extensibility

Extensibility is a key metric for measuring an IP telephony signaling protocol. Telephony is a tremendously popular, critical service, and Internet telephony is poised to supplant the existing circuit-switched infrastructure developed to support it. As with any heavily used service, the features provided evolve over time as new applications are developed. This makes compatibility among versions a complex issue. As the Internet is an open, distributed, and evolving entity, one can expect extensions to IP telephony protocols to be widespread and uncoordinated.

This makes it critical to include powerful extension mechanisms from the outset. SIP has learned the lessons of HTTP and SMTP (both of which are widely used protocols that have evolved over time) and built in a rich set of extensibility and compatibility functions. By default, unknown headers and values are ignored. Using the Require header, clients can indicate named feature sets that the server must understand. When a request arrives at a server, it checks the list of named features in the Requires header. If any of them are not supported, the server returns an error code and lists the set of features it does understand. The client can then determine the problematic feature and fall back to a simpler operation.

To further enhance extensibility, numerical error codes are hierarchically organized, as in HTTP. Six basic classes exist, each of which is identified by the hundreds digit in the response code (refer to Table 5-3). Basic protocol operation is dictated solely by the class, and terminals need only understand the class of the response. The other digits provide additional information, usually useful but not critical. This allows for additional features to be added by defining semantics for the error codes in a class while achieving compatibility. The textual encoding means that header fields are self-describing.

[16]Schulzrenne and Rosenberg.

As new header fields are added in various different implementations, developers in other vendors can determine usage just from the name and add support for the field. This kind of distributed, documentation-less standardization has been common in the SMTP, which has evolved tremendously over the years. As SIP is similar to HTTP, mechanisms being developed for HTTP extensibility can also be used in SIP. Among these are the *Protocol Extensions Protocol* (PEP), which contains pointers to the documentation for various features within the HTTP messages themselves.

H.323 provides extensibility mechanisms with some limitations. First, extensions are limited only to those places where a nonstandard parameter has been added. If a vendor wants to add a new value to some existing parameter, and no placeholder exists for a nonstandard element, one cannot be added. Secondly, H.323 has no mechanisms for enabling terminals to exchange information about which extensions each supports. As the values in nonstandard parameters are not self-describing, this limits interoperability among terminals from different manufacturers.

H.323 requires full backwards compatibility from each version to the next. As various features come and go, the size of the encodings will only increase. However, SIP enables older headers and features to gradually disappear as they are no longer needed, keeping the protocol and its encoding clean and concise. A critical issue for extensibility are audio and video *coder/decoders* (codecs). Hundreds of codecs have been developed, many of which are proprietary. SIP uses the SDP to convey the codecs supported by an endpoint in a session. This is due to the differences of ASN.1 versus text.

This means that SIP can work with any codec, and other implementations can determine the name of the codec and contact information for it. In H.323, each codec must be centrally registered and standardized. SIP enables new services to be defined through a few powerful third-party call-control mechanisms. These mechanisms enable a third party to instruct another entity to create and destroy calls to other entities. As the controlled party executes the instructions, status messages are passed back to the controller. This allows the controller to take further actions based on some local program execution. SIP enables these services to be deployed by basing them on simple, standardized mechanisms. These mechanisms can be used to construct a variety of services, including blind transfers, operator-assisted transfers, three-party calling, bridged calling, dial-in bridging, multiunicast to multicast transitions, ad hoc bridge invitations and transitions, and various forwarding variations.[17]

[17]Schulzrinne, Henning, and Jonathan Rosenberg. "Signaling for Internet Telephony." Technical Report CUCS-005-98. New York: Columbia University, February 1998.

Another aspect of extensibility is modularity. Internet telephony requires a large number of different functions; these include basic signaling, conference control, QoS, directory access, service discovery, and so on. One can be certain that mechanisms for accomplishing these functions will evolve over time (especially with regards to QoS). This makes it critical to apportion these functions to separate modular, orthogonal components, which can be swapped in and out over time. It is also critical to use separate, general protocols for each of these functions.

H.323 is less modular than SIP. It defines a vertically integrated protocol suite for a single application. The mix of services provided by the H.323 components encompass capability exchanges, conference control, maintenance operations, basic signaling, QoS, registration, and service discovery. Furthermore, these are intertwined within the various subprotocols within H.323. SIP's modularity enables it to be used in conjunction with H.323. A user can use SIP to locate another user, taking advantage of its rich multihop search facilities. When the user is finally located, he or she can use a redirect response to an H.323 URL, indicating that the actual communication should take place with H.323.[18] Table 5-5 compares H.323's call control service with that of SIP.

Table 5-5

Call control service comparison H.323 to SIP

Criteria	H.323	SIP
Complexity	Very complex	Simple
Message set	Many messages	Few messages
Debugging	Have to alter tools when a protocol is extended	Simple tools
Extensibility	Extensible	Very extensible
User extendable	ASN.1—complex	Text based—easy
Elements that must maintain states	Clients, gatekeepers, MCU, gateways, UAs, some proxy servers	
Processor usage	More overhead	Less overhead
Telephony features	Robust	Robust
Host application	Very complex	Much simpler
Code size	Large	Small
Dynamic memory usage	Large	Small to medium

[18]Ibid.

Services

What makes SIP and softswitch architectures so revolutionary is the ease with which they offer services and features. At the minimum, SIP supports personal mobility via its use of registration such that a corporate user can move his or her SIP phone anywhere on the corporate LAN or WAN, and the network will instantly know where to route that user's calls. One-number service is made possible by SIP forking proxies.

SIP messages carry *Multipurpose Internet Mail Extension* (MIME) content in addition to an SDP description. A response to an INVITE message could include a piece of text, an HTML document, an image, and so on. As an SIP address, a URL can be included in web content for click-to-call applications. SIP was designed for IP-based applications and can leverage those applications to create new telephony-oriented services.[19]

Almost as exciting as new services is the fact that SIP can be used to implement existing *Custom Local Area Signaling Service* (CLASS) services found in the PSTN (for example call forwarding, conferencing, caller ID, and so on). One significant objection service providers have had about VoIP and softswitch is the notion that softswitch architectures did not replicate the legendary 3,500 features that came with the 5ESS CLASS 5 switch. Given the flexibility of SIP and *voice XML* (VXML) software, those 3,500 features, in theory, can be matched or exceeded. This will be addressed in a later chapter comparing softswitch to CLASS 4 and 5 switches.

Signaling for SIP calls is routed via a proxy. This enables the proxy to utilize advanced feature logic. The feature logic may be resident at the proxy or in a separate feature server or database. The proxy may have access to other functions such as the policy server, an authentication server, and so on, each of which may be distributed throughout an IP network or all in one location. For example, the policy server may hold call-routing information or QoS-related information while the feature server holds subscriber information. The SIP proxy server can interface with SS7 signaling at the *service control point* (SCP) where services are offered via the *Intelligent Network Application Part* (INAP).[20] Interfacing SS7 and IP networks is addressed in a later chapter.

Given that SIP is a text-based language, many programmers in the industry are familiar with web applications, text parsers, and scripting languages that contribute to SIP. No special knowledge is necessary to create

[19]"SIP vs. H.323, A Business Analysis."
[20]Collins, pg. 207–208.

an SIP service. Also, any PC can be an SIP server to test new applications. SIP enables programmers familiar with the functional needs of a certain community to create their own services. Gone are the days when a service provider has to wait 18 months from the request to the delivery of a single new feature.

H.323 Versus SIP Conclusion

If SIP is better, why is H.323 important? A large, installed base of H.323 equipment exists, and vendors building new VoIP products are concerned about being backwards compatible with existing VoIP equipment and services. Carriers building VoIP networks also want to be able to enable their users to communicate with users on other VoIP networks. This would imply that H.323 is still an important protocol. Based on the technical advantages of SIP, the widespread support of SIP by H.323 vendors, the large number of SIP vendors already in the market, and the nearly universal availability of SIP services from the service providers, there is little argument in favor of continuing to develop H.323-based products.

The Big "So What?" about SIP

SIP offers some immediate impact on the telecommunications industry. Given its simlicity and ease of use, it has aleady found a numer of applications that could catapault its use into the mainstream in just a few years.

SIP on Windows XP

In March 2001, Microsoft announced that its new version of Windows, Windows XP, would contain SIP. This has immense implications. If every PC running Windows XP is an SIP UA, then every one of those computers is potentially a telephone and is at least somewhat independent of the PSTN. This would be especially true of PCs on corporate WANs where PC-to-PC calls would become the norm rather than the exception. For users with anthropological barriers to using a PC as a phone, very inexpensive handsets are available that interface the PC to the user.

Microsoft projects upwards of 245 million XP users by the end of 2002. If, eventually, all 245 million XP users shifted at least some of their telephony

usage to their PC via Windows XP SIP applications, there would be some significant loss of traffic to the legacy telephone companies. No telephone company in the world has 245 million subscribers. What if Windows XP SIP becomes to telephony what Microsoft Word has become to word processing?

How Does That Work?

Microsoft, by developing and distributing Windows XP with SIP, has facilitated some deconstruction of the telecom industry as we know it. To make an SIP call via Windows XP, the user gains access to the Internet (potentially via Microsoft Network, Microsoft's ISP) or their IP corporate WAN. It should be noted here that the MSN ISP has some 5 million subscribers. With the acquisition of a few large U.S. ISPs, Microsoft Network could be one of the largest if not the largest ISP in the United States.

Using their account with one of the partner IP telephony service providers (Net2Phone, DialPad, or Delta3), users can place a call to any phone in the world. WorldCom plans to offer VoIP services to XP users in the enterprise market. In the United Kingdom, Callserve Communications plans to do the same in the residential market. In Canada, Microsoft has partnered with Telus, offering voice-over-Internet services to high-speed subscribers.

Referring back to the PSTN model comprised of access, switching, and transport, Microsoft assumes the position of providing access to the voice network. Specifically, the Windows-XP-powered PC potentially replaces the handset; assuming the subscriber is using Microsoft Network as its ISP, Microsoft is the de facto local loop. Switching is provided by the IP telephony carriers (DialPad, Net2Phone, and so on) and transport is via an IP backbone service provider (Level3 or Cable and Wireless).[21] As a result, the central office is potentially replaced.

The specific agent in Windows XP that contains the SIP package is Windows Messenger. To deliver Internet telephony to the Windows user, Windows Messenger offers three broad technological improvements. First, it significantly reduces voice delay, the chief quality issue with VoIP, to a min-

[21]Arnold, Jon. "Where Does Microsoft Want To Go Today?" *Sounding Board Magazine*, December 2001. www.soundingboardmag.com/articles/1c1sb.html.

imum of about 70 milliseconds. When communicating clients are on one well-engineered LAN, this delay is not noticeable. When clients communicate over the Internet, there is substantial headroom (at least 80 milliseconds) before delays begin to interfere with conversational dynamics.

Second, Windows XP comes with a variety of voice codecs that Windows Messenger uses. When network conditions favor wide bandwidth and low delay, these make voice sound significantly better than telephone quality. When network throughput degrades or delay increases, coders are able to gracefully fall back to lower bit rates. When network congestion eases again, coders dynamically step up to higher quality. This happens automatically, under application control.

Third, Windows Messenger includes acoustic echo cancellation software, which reduces echo and eliminates the need for a headset in average-sized offices. Windows Messenger doesn't solve Internet delay or reachability problems, but it appears at the right time. Today the Internet is significantly faster than it was five or six years ago. Many, many more people have PCs. Increasingly, these PCs are always on. In the ideal workplace, these PCs are usually connected to the Internet via 100 Mbps Ethernet. At home, residential always-on connections (DSL, cable, and wireless) are becoming more and more prevalent. Increasingly, PCs can place IP phone calls to any other phone in the world using gateway providers, as mentioned previously.[22]

This product may find a very fertile market outside the United States, particularly in nations where even local calls (which is most of the world outside the United States) are billed by the minute and it takes years to get a landline phone installed (assuming alternatives to the last mile). Assuming a similar proportion of long-distance business worldwide is comparable to the 70 percent figure in the United States, non-U.S. businesses will want to move their interoffice phone traffic, both local and long distance, on to their corporate WAN courtesy of the SIP package contained in Windows XP. For many in the developing world, Internet access is gained at a local cyber café. PCs in those cyber cafés could be configured to perform as telephones. This will have the effect of introducing competition *to* the local loop in those countries.

---

[22]Isenberg, David. "Windows Messenger: New Waves of Innovation." A white paper for Microsoft, www.microsoft.com/windowsxp/pro/techinfo/planning/networking/windowsmessenger.asp.

Conclusion

This chapter discussed SIP as its own softswitch architecture. Rather than be a compromise between TDM and IP networks, SIP was designed from the start as a signaling protocol for VoIP. Its architecture is more scalable, extensible, and simple. The consensus in the industry is that SIP poses a better value proposition for service providers and corporate networks than does H.323. The prospect of offering services via an SIP infrastructure that cannot be provided by TDM switches has many service providers considering SIP as an easy-to-deploy VoIP architecture solution.

It should be noted here that in 1995 Microsoft introduced its Windows 95 product containing networking protocol TCP/IP. Within approximately two years, TCP/IP became the de facto networking protocol for enterprise networks as almost all corporate PCs were loaded with and networked with Windows 95. H.323 also came into prominence via its inclusion in Microsoft's NetMeeting product. Given these precedences, a strong possibility exists that the inclusion of SIP in Windows XP, due to the sheer volume of the number of PCs that will ultimately be loaded with Windows XP, could well establish SIP as the dominant signaling protocol for VoIP.

Telecommunications Essentials

The Complete Global Source for Communications Fundamentals, Data Networking and the Internet, and Next-Generation Networks

Lillian Goleniewski

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

The publisher offers discounts on this book when ordered in quantity for special sales. For more information, please contact:

Pearson Education Corporate Sales Division
201 W. 103rd Street
Indianapolis, IN 46290
(800) 428-5331
corpsales@pearsoned.com

Visit AW on the Web: www.aw.com/cseng/

Library of Congress Cataloging-in-Publication Data

Goleniewski, Lillian.
Telecommunications essentials : the complete global source for communications fundamentals, data networking and the Internet, and next-generation networks / Lillian Goleniewski.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-76032-0
1. Telecommunication. I. Title.

TK5101 G598 2002
621.382—dc21 2001053752

ISBN 0-201-76032-0
Text printed on recycled paper

1 2 3 4 5 6 7 8 9 10—CRS—0504030201
First printing, December 2001

Table 8.2 LAN Technologies and Cabling Requirements

Technology	Type of Cable
	Ethernet (10Mbps)
10Base5	Thick coax
10Base2	Thin coax
10BaseT	2-pair UTP
10BaseFL	2 strands of multimode optical fiber
	Fast Ethernet (100Mbps)
100BaseTX	2-pair Cat 5 UTP
100BaseT4	4-pair Cat 3 UTP
100BaseT2	2-pair Cat 3 UTP
100BaseFX	2 strands of multimode optical fiber
	Gigabit Ethernet (1Gbps)
1000BaseSX	Short-wavelength multimode optical fiber
1000BaseLX	Long-wavelength single-mode optical fiber
1000BaseCX	Coax patch cable
1000BaseT	4-pair Cat 5 or Cat 5e UTP
1000BaseTX	2-pair Cat 6 (currently a TIA draft proposal)

The prevailing standard in the world is Ethernet. It generally appears as Fast Ethernet or Gigabit Ethernet in the backbone, connecting together individual Fast Ethernet or 10Mbps Ethernet LAN segments (see Figure 8.3).

LAN Access Methods

The third main LAN characteristic is the access methods, which are involved in determining who gets to use the network and when they get to use it. There are two main approaches: token passing and carrier-sense multiple-access/collision detect (CSMA/CD).

Network Working Group
Request for Comments: 1122

Internet Engineering Task Force
R. Braden, Editor
October 1989

Requirements for Internet Hosts -- Communication Layers

Status of This Memo

This RFC is an official specification for the Internet community. It incorporates by reference, amends, corrects, and supplements the primary protocol standards documents relating to hosts. Distribution of this document is unlimited.

Summary

This is one RFC of a pair that defines and discusses the requirements for Internet host software. This RFC covers the communications protocol layers: link layer, IP layer, and transport layer; its companion RFC-1123 covers the application and support protocols.

Table of Contents

end through the Internet, a message must be encapsulated inside a datagram.

IP Datagram
An IP datagram is the unit of end-to-end transmission in the IP protocol. An IP datagram consists of an IP header followed by transport layer data, i.e., of an IP header followed by a message.

In the description of the internet layer (Section 3), the unqualified term "datagram" should be understood to refer to an IP datagram.

Packet
A packet is the unit of data passed across the interface between the internet layer and the link layer. It includes an IP header and data. A packet may be a complete IP datagram or a fragment of an IP datagram.

Frame
A frame is the unit of transmission in a link layer protocol, and consists of a link-layer header followed by a packet.

Connected Network
A network to which a host is interfaced is often known as the "local network" or the "subnetwork" relative to that host. However, these terms can cause confusion, and therefore we use the term "connected network" in this document.

Multihomed
A host is said to be multihomed if it has multiple IP addresses. For a discussion of multihoming, see Section 3.3.4 below.

Physical network interface
This is a physical interface to a connected network and has a (possibly unique) link-layer address. Multiple physical network interfaces on a single host may share the same link-layer address, but the address must be unique for different hosts on the same physical network.

Logical [network] interface
We define a logical [network] interface to be a logical path, distinguished by a unique IP address, to a connected network. See Section 3.3.4.

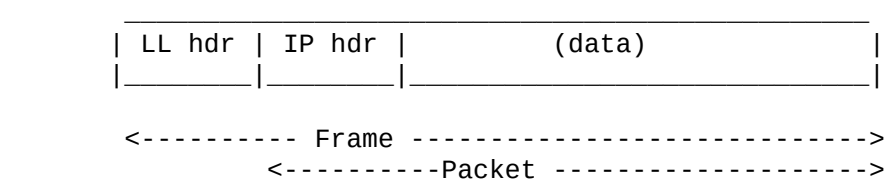
Specific-destination address
This is the effective destination address of a datagram, even if it is broadcast or multicast; see Section 3.2.1.3.

Path
At a given moment, all the IP datagrams from a particular source host to a particular destination host will typically traverse the same sequence of gateways. We use the term "path" for this sequence. Note that a path is uni-directional; it is not unusual to have different paths in the two directions between a given host pair.

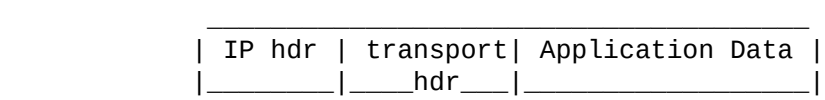
MTU
The maximum transmission unit, i.e., the size of the largest packet that can be transmitted.

The terms frame, packet, datagram, message, and segment are illustrated by the following schematic diagrams:

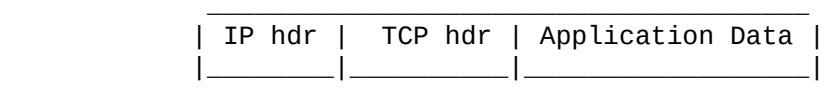
A. Transmission on connected network:



B. Before IP fragmentation or after IP reassembly:



or, for TCP:



1.4 Acknowledgments

This document incorporates contributions and comments from a large group of Internet protocol experts, including representatives of university and research labs, vendors, and government agencies. It was assembled primarily by the Host Requirements Working Group of the Internet Engineering Task Force (IETF).

The Editor would especially like to acknowledge the tireless dedication of the following people, who attended many long meetings and generated 3 million bytes of electronic mail over the past 18 months in pursuit of this document: Philip Almquist, Dave Borman (Cray Research), Noel Chiappa, Dave Crocker (DEC), Steve Deering (Stanford), Mike Karels (Berkeley), Phil Karn (Bellcore), John Lekashman (NASA), Charles Lynn (BBN), Keith McCloghrie (TWG), Paul Mockapetris (ISI), Thomas Narten (Purdue), Craig Partridge (BBN), Drew Perkins (CMU), and James Van Bokkelen (FTP Software).

In addition, the following people made major contributions to the effort: Bill Barns (Mitre), Steve Bellovin (AT&T), Mike Brescia (BBN), Ed Cain (DCA), Annette DeSchon (ISI), Martin Gross (DCA), Phill Gross (NRI), Charles Hedrick (Rutgers), Van Jacobson (LBL), John Klensin (MIT), Mark Lottor (SRI), Milo Medin (NASA), Bill Melohn (Sun Microsystems), Greg Minshall (Kinetics), Jeff Mogul (DEC), John Mullen (CMC), Jon Postel (ISI), John Romkey (Epilogue Technology), and Mike StJohns (DCA). The following also made significant contributions to particular areas: Eric Allman (Berkeley), Rob Austein (MIT), Art Berggreen (ACC), Keith Bostic (Berkeley), Vint Cerf (NRI), Wayne Hathaway (NASA), Matt Korn (IBM), Erik Naggum (Naggum Software, Norway), Robert Ullmann (Prime Computer), David Waitzman (BBN), Frank Wancho (USA), Arun Welch (Ohio State), Bill Westfield (Cisco), and Rayan Zachariassen (Toronto).

We are grateful to all, including any contributors who may have been inadvertently omitted from this list.

2. LINK LAYER

2.1 INTRODUCTION

All Internet systems, both hosts and gateways, have the same requirements for link layer protocols. These requirements are given in Chapter 3 of "Requirements for Internet Gateways" [INTRO:2], augmented with the material in this section.

2.2 PROTOCOL WALK-THROUGH

None.

2.3 SPECIFIC ISSUES

2.3.1 Trailer Protocol Negotiation

The trailer protocol [LINK:1] for link-layer encapsulation MAY be used, but only when it has been verified that both systems (host or gateway) involved in the link-layer communication implement trailers. If the system does not dynamically negotiate use of the trailer protocol on a per-destination basis, the default configuration MUST disable the protocol.

DISCUSSION:

The trailer protocol is a link-layer encapsulation technique that rearranges the data contents of packets sent on the physical network. In some cases, trailers improve the throughput of higher layer protocols by reducing the amount of data copying within the operating system. Higher layer protocols are unaware of trailer use, but both the sending and receiving host MUST understand the protocol if it is used.

Improper use of trailers can result in very confusing symptoms. Only packets with specific size attributes are encapsulated using trailers, and typically only a small fraction of the packets being exchanged have these attributes. Thus, if a system using trailers exchanges packets with a system that does not, some packets disappear into a black hole while others are delivered successfully.

IMPLEMENTATION:

On an Ethernet, packets encapsulated with trailers use a distinct Ethernet type [LINK:1], and trailer negotiation is performed at the time that ARP is used to discover the link-layer address of a destination system.

Specifically, the ARP exchange is completed in the usual manner using the normal IP protocol type, but a host that wants to speak trailers will send an additional "trailer ARP reply" packet, i.e., an ARP reply that specifies the trailer encapsulation protocol type but otherwise has the format of a normal ARP reply. If a host configured to use trailers receives a trailer ARP reply message from a remote machine, it can add that machine to the list of machines that understand trailers, e.g., by marking the corresponding entry in the ARP cache.

Hosts wishing to receive trailer encapsulations send trailer ARP replies whenever they complete exchanges of normal ARP messages for IP. Thus, a host that received an ARP request for its IP protocol address would send a trailer ARP reply in addition to the normal IP ARP reply; a host that sent the IP ARP request would send a trailer ARP reply when it received the corresponding IP ARP reply. In this way, either the requesting or responding host in an IP ARP exchange may request that it receive trailer encapsulations.

This scheme, using extra trailer ARP reply packets rather than sending an ARP request for the trailer protocol type, was designed to avoid a continuous exchange of ARP packets with a misbehaving host that, contrary to any specification or common sense, responded to an ARP reply for trailers with another ARP reply for IP. This problem is avoided by sending a trailer ARP reply in response to an IP ARP reply only when the IP ARP reply answers an outstanding request; this is true when the hardware address for the host is still unknown when the IP ARP reply is received. A trailer ARP reply may always be sent along with an IP ARP reply responding to an IP ARP request.

2.3.2 Address Resolution Protocol -- ARP

2.3.2.1 ARP Cache Validation

An implementation of the Address Resolution Protocol (ARP) [LINK:2] MUST provide a mechanism to flush out-of-date cache entries. If this mechanism involves a timeout, it SHOULD be possible to configure the timeout value.

A mechanism to prevent ARP flooding (repeatedly sending an ARP Request for the same IP address, at a high rate) MUST be included. The recommended maximum rate is 1 per second per

destination.

DISCUSSION:

The ARP specification [LINK:2] suggests but does not require a timeout mechanism to invalidate cache entries when hosts change their Ethernet addresses. The prevalence of proxy ARP (see Section 2.4 of [INTRO:2]) has significantly increased the likelihood that cache entries in hosts will become invalid, and therefore some ARP-cache invalidation mechanism is now required for hosts. Even in the absence of proxy ARP, a long-period cache timeout is useful in order to automatically correct any bad ARP data that might have been cached.

IMPLEMENTATION:

Four mechanisms have been used, sometimes in combination, to flush out-of-date cache entries.

(1) Timeout -- Periodically time out cache entries, even if they are in use. Note that this timeout should be restarted when the cache entry is "refreshed" (by observing the source fields, regardless of target address, of an ARP broadcast from the system in question). For proxy ARP situations, the timeout needs to be on the order of a minute.

(2) Unicast Poll -- Actively poll the remote host by periodically sending a point-to-point ARP Request to it, and delete the entry if no ARP Reply is received from N successive polls. Again, the timeout should be on the order of a minute, and typically N is 2.

(3) Link-Layer Advice -- If the link-layer driver detects a delivery problem, flush the corresponding ARP cache entry.

(4) Higher-layer Advice -- Provide a call from the Internet layer to the link layer to indicate a delivery problem. The effect of this call would be to invalidate the corresponding cache entry. This call would be analogous to the "ADVISE_DELIVPROB()" call from the transport layer to the Internet layer (see Section 3.4), and in fact the ADVISE_DELIVPROB routine might in turn call the link-layer advice routine to invalidate

the ARP cache entry.

Approaches (1) and (2) involve ARP cache timeouts on the order of a minute or less. In the absence of proxy ARP, a timeout this short could create noticeable overhead traffic on a very large Ethernet. Therefore, it may be necessary to configure a host to lengthen the ARP cache timeout.

2.3.2.2 ARP Packet Queue

The link layer SHOULD save (rather than discard) at least one (the latest) packet of each set of packets destined to the same unresolved IP address, and transmit the saved packet when the address has been resolved.

DISCUSSION:
Failure to follow this recommendation causes the first packet of every exchange to be lost. Although higher-layer protocols can generally cope with packet loss by retransmission, packet loss does impact performance. For example, loss of a TCP open request causes the initial round-trip time estimate to be inflated. UDP-based applications such as the Domain Name System are more seriously affected.

2.3.3 Ethernet and IEEE 802 Encapsulation

The IP encapsulation for Ethernets is described in RFC-894 [LINK:3], while RFC-1042 [LINK:4] describes the IP encapsulation for IEEE 802 networks. RFC-1042 elaborates and replaces the discussion in Section 3.4 of [INTRO:2].

Every Internet host connected to a 10Mbps Ethernet cable:

- MUST be able to send and receive packets using RFC-894 encapsulation;
- SHOULD be able to receive RFC-1042 packets, intermixed with RFC-894 packets; and
- MAY be able to send packets using RFC-1042 encapsulation.

An Internet host that implements sending both the RFC-894 and the RFC-1042 encapsulations MUST provide a configuration switch to select which is sent, and this switch MUST default to RFC-894.

Note that the standard IP encapsulation in RFC-1042 does not use the protocol id value (K1=6) that IEEE reserved for IP; instead, it uses a value (K1=170) that implies an extension (the "SNAP") which can be used to hold the Ether-Type field. An Internet system MUST NOT send 802 packets using K1=6.

Address translation from Internet addresses to link-layer addresses on Ethernet and IEEE 802 networks MUST be managed by the Address Resolution Protocol (ARP).

The MTU for an Ethernet is 1500 and for 802.3 is 1492.

DISCUSSION:

The IEEE 802.3 specification provides for operation over a 10Mbps Ethernet cable, in which case Ethernet and IEEE 802.3 frames can be physically intermixed. A receiver can distinguish Ethernet and 802.3 frames by the value of the 802.3 Length field; this two-octet field coincides in the header with the Ether-Type field of an Ethernet frame. In particular, the 802.3 Length field must be less than or equal to 1500, while all valid Ether-Type values are greater than 1500.

Another compatibility problem arises with link-layer broadcasts. A broadcast sent with one framing will not be seen by hosts that can receive only the other framing.

The provisions of this section were designed to provide direct interoperation between 894-capable and 1042-capable systems on the same cable, to the maximum extent possible. It is intended to support the present situation where 894-only systems predominate, while providing an easy transition to a possible future in which 1042-capable systems become common.

Note that 894-only systems cannot interoperate directly with 1042-only systems. If the two system types are set up as two different logical networks on the same cable, they can communicate only through an IP gateway. Furthermore, it is not useful or even possible for a dual-format host to discover automatically which format to send, because of the problem of link-layer broadcasts.

2.4 LINK/INTERNET LAYER INTERFACE

The packet receive interface between the IP layer and the link layer MUST include a flag to indicate whether the incoming packet was addressed to a link-layer broadcast address.

RFC: 791

INTERNET PROTOCOL

DARPA INTERNET PROGRAM

PROTOCOL SPECIFICATION

September 1981

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

TABLE OF CONTENTS

PREFACE

This document specifies the DoD Standard Internet Protocol. This document is based on six earlier editions of the ARPA Internet Protocol Specification, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition revises aspects of addressing, error handling, option codes, and the security, precedence, compartments, and handling restriction features of the internet protocol.

Jon Postel

Editor

September 1981

RFC: 791
Replaces: RFC 760
IENs 128, 123, 111,
80, 54, 44, 41, 28, 26

INTERNET PROTOCOL

DARPA INTERNET PROGRAM
PROTOCOL SPECIFICATION

1. INTRODUCTION

1.1. Motivation

The Internet Protocol is designed for use in interconnected systems of packet-switched computer communication networks. Such a system has been called a "catenet" [1]. The internet protocol provides for transmitting blocks of data called datagrams from sources to destinations, where sources and destinations are hosts identified by fixed length addresses. The internet protocol also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through "small packet" networks.

1.2. Scope

The internet protocol is specifically limited in scope to provide the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. There are no mechanisms to augment end-to-end data reliability, flow control, sequencing, or other services commonly found in host-to-host protocols. The internet protocol can capitalize on the services of its supporting networks to provide various types and qualities of service.

1.3. Interfaces

This protocol is called on by host-to-host protocols in an internet environment. This protocol calls on local network protocols to carry the internet datagram to the next gateway or destination host.

For example, a TCP module would call on the internet module to take a TCP segment (including the TCP header and user data) as the data portion of an internet datagram. The TCP module would provide the addresses and other parameters in the internet header to the internet module as arguments of the call. The internet module would then create an internet datagram and call on the local network interface to transmit the internet datagram.

In the ARPANET case, for example, the internet module would call on a

local net module which would add the 1822 leader [2] to the internet datagram creating an ARPANET message to transmit to the IMP. The ARPANET address would be derived from the internet address by the local network interface and would be the address of some host in the ARPANET, that host might be a gateway to other networks.

1.4. Operation

The internet protocol implements two basic functions: addressing and fragmentation.

The internet modules use the addresses carried in the internet header to transmit internet datagrams toward their destinations. The selection of a path for transmission is called routing.

The internet modules use fields in the internet header to fragment and reassemble internet datagrams when necessary for transmission through "small packet" networks.

The model of operation is that an internet module resides in each host engaged in internet communication and in each gateway that interconnects networks. These modules share common rules for interpreting address fields and for fragmenting and assembling internet datagrams. In addition, these modules (especially in gateways) have procedures for making routing decisions and other functions.

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

The Type of Service is used to indicate the quality of the service desired. The type of service is an abstract or generalized set of parameters which characterize the service choices provided in the networks that make up the internet. This type of service indication is to be used by gateways to select the actual transmission parameters for a particular network, the network to be used for the next hop, or the next gateway when routing an internet datagram.

The Time to Live is an indication of an upper bound on the lifetime of an internet datagram. It is set by the sender of the datagram and reduced at the points along the route where it is processed. If the time to live reaches zero before the internet datagram reaches its destination, the internet datagram is destroyed. The time to live can be thought of as a self destruct time limit.

The Options provide for control functions needed or useful in some situations but unnecessary for the most common communications. The options include provisions for timestamps, security, and special routing.

The Header Checksum provides a verification that the information used in processing internet datagram has been transmitted correctly. The data may contain errors. If the header checksum fails, the internet datagram is discarded at once by the entity which detects the error.

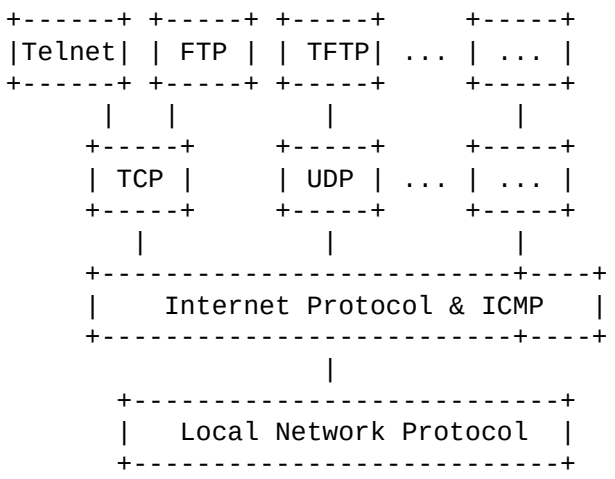
The internet protocol does not provide a reliable communication facility. There are no acknowledgments either end-to-end or hop-by-hop. There is no error control for data, only a header checksum. There are no retransmissions. There is no flow control.

Errors detected may be reported via the Internet Control Message Protocol (ICMP) [3] which is implemented in the internet protocol module.

2. OVERVIEW

2.1. Relation to Other Protocols

The following diagram illustrates the place of the internet protocol in the protocol hierarchy:



Protocol Relationships

Figure 1.

Internet protocol interfaces on one side to the higher level host-to-host protocols and on the other side to the local network protocol. In this context a "local network" may be a small network in a building or a large network such as the ARPANET.

2.2. Model of Operation

The model of operation for transmitting a datagram from one application program to another is illustrated by the following scenario:

We suppose that this transmission will involve one intermediate gateway.

The sending application program prepares its data and calls on its local internet module to send that data as a datagram and passes the destination address and other parameters as arguments of the call.

The internet module prepares a datagram header and attaches the data to it. The internet module determines a local network address for this internet address, in this case it is the address of a gateway.

It sends this datagram and the local network address to the local network interface.

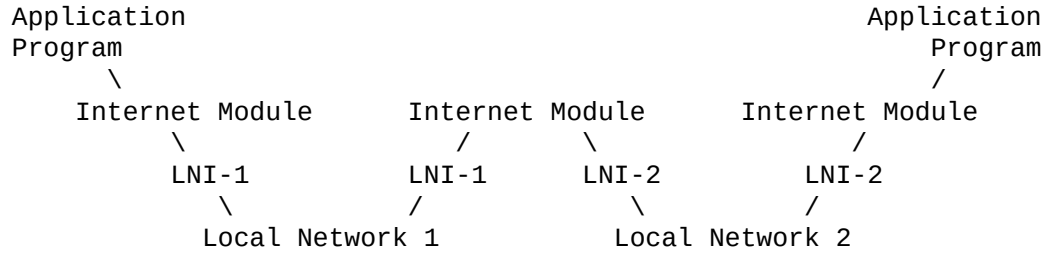
The local network interface creates a local network header, and attaches the datagram to it, then sends the result via the local network.

The datagram arrives at a gateway host wrapped in the local network header, the local network interface strips off this header, and turns the datagram over to the internet module. The internet module determines from the internet address that the datagram is to be forwarded to another host in a second network. The internet module determines a local net address for the destination host. It calls on the local network interface for that network to send the datagram.

This local network interface creates a local network header and attaches the datagram sending the result to the destination host.

At this destination host the datagram is stripped of the local net header by the local network interface and handed to the internet module.

The internet module determines that the datagram is for an application program in this host. It passes the data to the application program in response to a system call, passing the source address and other parameters as results of the call.



Transmission Path

Figure 2

2.3. Function Description

The function or purpose of Internet Protocol is to move datagrams through an interconnected set of networks. This is done by passing the datagrams from one internet module to another until the destination is reached. The internet modules reside in hosts and gateways in the internet system. The datagrams are routed from one internet module to another through individual networks based on the interpretation of an internet address. Thus, one important mechanism of the internet protocol is the internet address.

In the routing of messages from one internet module to another, datagrams may need to traverse a network whose maximum packet size is smaller than the size of the datagram. To overcome this difficulty, a fragmentation mechanism is provided in the internet protocol.

Addressing

A distinction is made between names, addresses, and routes [4]. A name indicates what we seek. An address indicates where it is. A route indicates how to get there. The internet protocol deals primarily with addresses. It is the task of higher level (i.e., host-to-host or application) protocols to make the mapping from names to addresses. The internet module maps internet addresses to local net addresses. It is the task of lower level (i.e., local net or gateways) procedures to make the mapping from local net addresses to routes.

Addresses are fixed length of four octets (32 bits). An address begins with a network number, followed by local address (called the "rest" field). There are three formats or classes of internet addresses: in class a, the high order bit is zero, the next 7 bits are the network, and the last 24 bits are the local address; in class b, the high order two bits are one-zero, the next 14 bits are the network and the last 16 bits are the local address; in class c, the high order three bits are one-one-zero, the next 21 bits are the network and the last 8 bits are the local address.

Care must be taken in mapping internet addresses to local net addresses; a single physical host must be able to act as if it were several distinct hosts to the extent of using several distinct internet addresses. Some hosts will also have several physical interfaces (multi-homing).

That is, provision must be made for a host to have several physical interfaces to the network with each having several logical internet addresses.

Examples of address mappings may be found in "Address Mappings" [5].

Fragmentation

Fragmentation of an internet datagram is necessary when it originates in a local net that allows a large packet size and must traverse a local net that limits packets to a smaller size to reach its destination.

An internet datagram can be marked "don't fragment." Any internet datagram so marked is not to be internet fragmented under any circumstances. If internet datagram marked don't fragment cannot be delivered to its destination without fragmenting it, it is to be discarded instead.

Fragmentation, transmission and reassembly across a local network which is invisible to the internet protocol module is called intranet fragmentation and may be used [6].

The internet fragmentation and reassembly procedure needs to be able to break a datagram into an almost arbitrary number of pieces that can be later reassembled. The receiver of the fragments uses the identification field to ensure that fragments of different datagrams are not mixed. The fragment offset field tells the receiver the position of a fragment in the original datagram. The fragment offset and length determine the portion of the original datagram covered by this fragment. The more-fragments flag indicates (by being reset) the last fragment. These fields provide sufficient information to reassemble datagrams.

The identification field is used to distinguish the fragments of one datagram from those of another. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system. The originating protocol module of a complete datagram sets the more-fragments flag to zero and the fragment offset to zero.

To fragment a long internet datagram, an internet protocol module (for example, in a gateway), creates two new internet datagrams and copies the contents of the internet header fields from the long datagram into both new internet headers. The data of the long datagram is divided into two portions on a 8 octet (64 bit) boundary (the second portion might not be an integral multiple of 8 octets, but the first must be). Call the number of 8 octet blocks in the first portion NFB (for Number of Fragment Blocks). The first portion of the data is placed in the first new internet datagram, and the total length field is set to the length of the first

datagram. The more-fragments flag is set to one. The second portion of the data is placed in the second new internet datagram, and the total length field is set to the length of the second datagram. The more-fragments flag carries the same value as the long datagram. The fragment offset field of the second new internet datagram is set to the value of that field in the long datagram plus NFB.

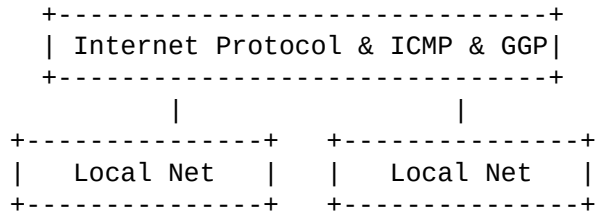
This procedure can be generalized for an n-way split, rather than the two-way split described.

To assemble the fragments of an internet datagram, an internet protocol module (for example at a destination host) combines internet datagrams that all have the same value for the four fields: identification, source, destination, and protocol. The combination is done by placing the data portion of each fragment in the relative position indicated by the fragment offset in that fragment's internet header. The first fragment will have the fragment offset zero, and the last fragment will have the more-fragments flag reset to zero.

2.4. Gateways

Gateways implement internet protocol to forward datagrams between networks. Gateways also implement the Gateway to Gateway Protocol (GGP) [7] to coordinate routing and other internet control information.

In a gateway the higher level protocols need not be implemented and the GGP functions are added to the IP module.



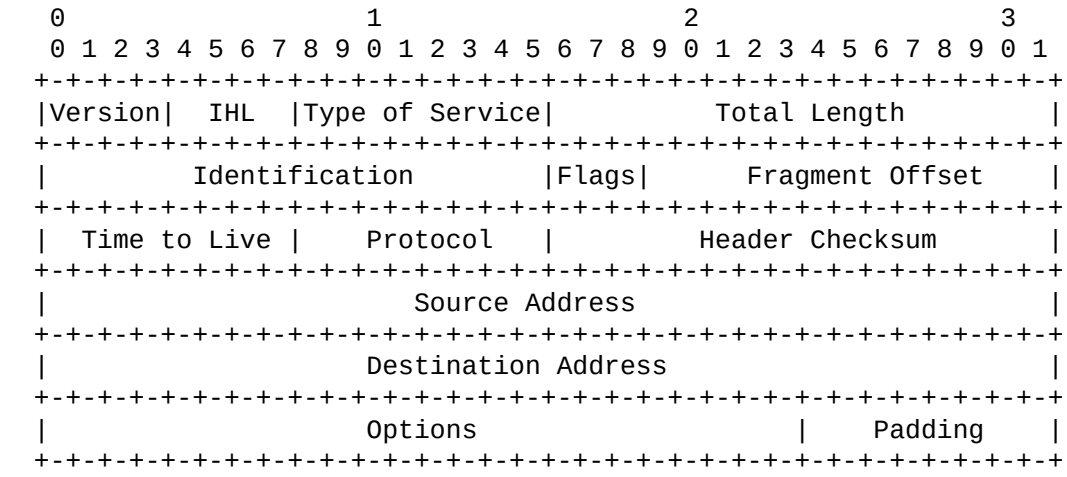
Gateway Protocols

Figure 3.

3. SPECIFICATION

3.1. Internet Header Format

A summary of the contents of the internet header follows:



Example Internet Datagram Header

Figure 4.

Note that each tick mark represents one bit position.

Version: 4 bits

The Version field indicates the format of the internet header. This document describes version 4.

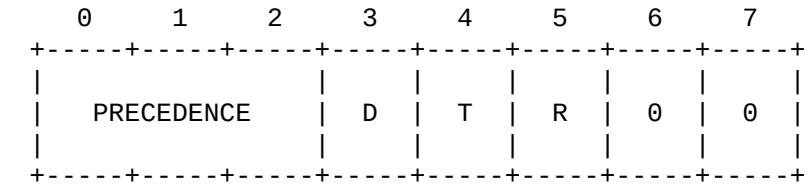
IHL: 4 bits

Internet Header Length is the length of the internet header in 32 bit words, and thus points to the beginning of the data. Note that the minimum value for a correct header is 5.

Type of Service: 8 bits

The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are to be used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which somehow treats high precedence traffic as more important than other traffic (generally by accepting only traffic above a certain precedence at time of high load). The major choice is a three way tradeoff between low-delay, high-reliability, and high-throughput.

```
Bits 0-2:  Precedence.
Bit    3:  0 = Normal Delay,      1 = Low Delay.
Bits   4:  0 = Normal Throughput, 1 = High Throughput.
Bits   5:  0 = Normal Relibility, 1 = High Relibility.
Bit  6-7:  Reserved for Future Use.
```



Precedence

```
111 - Network Control
110 - Internetwork Control
101 - CRITIC/ECP
100 - Flash Override
011 - Flash
010 - Immediate
001 - Priority
000 - Routine
```

The use of the Delay, Throughput, and Reliability indications may increase the cost (in some sense) of the service. In many networks better performance for one of these parameters is coupled with worse performance on another. Except for very unusual cases at most two of these three indications should be set.

The type of service is used to specify the treatment of the datagram during its transmission through the internet system. Example mappings of the internet type of service to the actual service provided on networks such as AUTODIN II, ARPANET, SATNET, and PRNET is given in "Service Mappings" [8].

The Network Control precedence designation is intended to be used within a network only. The actual use and control of that designation is up to each network. The Internetwork Control designation is intended for use by gateway control originators only. If the actual use of these precedence designations is of concern to a particular network, it is the responsibility of that network to control the access to, and use of, those precedence designations.

Total Length: 16 bits

Total Length is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets (whether they arrive whole or in fragments). It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.

The number 576 is selected to allow a reasonable sized data block to be transmitted in addition to the required header information. For example, this size allows a data block of 512 octets plus 64 header octets to fit in a datagram. The maximal internet header is 60 octets, and a typical internet header is 20 octets, allowing a margin for headers of higher level protocols.

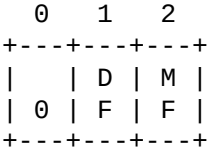
Identification: 16 bits

An identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags: 3 bits

Various Control Flags.

Bit 0: reserved, must be zero
Bit 1: (DF) 0 = May Fragment, 1 = Don't Fragment.
Bit 2: (MF) 0 = Last Fragment, 1 = More Fragments.



Fragment Offset: 13 bits

This field indicates where in the datagram this fragment belongs.

The fragment offset is measured in units of 8 octets (64 bits). The first fragment has offset zero.

Time to Live: 8 bits

This field indicates the maximum time the datagram is allowed to remain in the internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing. The time is measured in units of seconds, but since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Protocol: 8 bits

This field indicates the next level protocol used in the data portion of the internet datagram. The values for various protocols are specified in "Assigned Numbers" [9].

Header Checksum: 16 bits

A checksum on the header only. Since some header fields change (e.g., time to live), this is recomputed and verified at each point that the internet header is processed.

The checksum algorithm is:

The checksum field is the 16 bit one’s complement of the one’s complement sum of all 16 bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

This is a simple to compute checksum and experimental evidence indicates it is adequate, but it is provisional and may be replaced by a CRC procedure, depending on further experience.

Source Address: 32 bits

The source address. See section 3.2.

Destination Address: 32 bits

The destination address. See section 3.2.

Options: variable

The options may appear or not in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation.

In some environments the security option may be required in all datagrams.

The option field is variable in length. There may be zero or more options. There are two cases for the format of an option:

Case 1: A single octet of option-type.

Case 2: An option-type octet, an option-length octet, and the actual option-data octets.

The option-length octet counts the option-type octet and the option-length octet as well as the option-data octets.

The option-type octet is viewed as having 3 fields:

1 bit copied flag,
2 bits option class,
5 bits option number.

The copied flag indicates that this option is copied into all fragments on fragmentation.

0 = not copied
1 = copied

The option classes are:

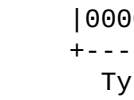
0 = control
1 = reserved for future use
2 = debugging and measurement
3 = reserved for future use

The following internet options are defined:

CLASS	NUMBER	LENGTH	DESCRIPTION
0	0	-	End of Option list. This option occupies only 1 octet; it has no length octet.
0	1	-	No Operation. This option occupies only 1 octet; it has no length octet.
0	2	11	Security. Used to carry Security, Compartmentation, User Group (TCC), and Handling Restriction Codes compatible with DOD requirements.
0	3	var.	Loose Source Routing. Used to route the internet datagram based on information supplied by the source.
0	9	var.	Strict Source Routing. Used to route the internet datagram based on information supplied by the source.
0	7	var.	Record Route. Used to trace the route an internet datagram takes.
0	8	4	Stream ID. Used to carry the stream identifier.
2	4	var.	Internet Timestamp.

Specific Option Definitions

End of Option List



This option indicates the end of the option list. This might not coincide with the end of the internet header according to the internet header length. This is used at the end of all options, not the end of each option, and need only be used if the end of the options would not otherwise coincide with the end of the internet header.

May be copied, introduced, or deleted on fragmentation, or for any other reason.

No Operation

```
  +--------+
  |00000001|
  +--------+
    Type=1
```

This option may be used between options, for example, to align the beginning of a subsequent option on a 32 bit boundary.

May be copied, introduced, or deleted on fragmentation, or for any other reason.

Security

This option provides a way for hosts to send security, compartmentation, handling restrictions, and TCC (closed user group) parameters. The format for this option is as follows:

```
  +--------+--------+---//---+---//---+---//---+---//---+
  |10000010|00001011|SSS  SSS|CCC  CCC|HHH  HHH|  TCC   |
  +--------+--------+---//---+---//---+---//---+---//---+
   Type=130 Length=11
```

Security (S field): 16 bits

Specifies one of 16 levels of security (eight of which are reserved for future use).

```
  00000000 00000000 - Unclassified
  11110001 00110101 - Confidential
  01111000 10011010 - EFTO
  10111100 01001101 - MMMM
  01011110 00100110 - PROG
  10101111 00010011 - Restricted
  11010111 10001000 - Secret
  01101011 11000101 - Top Secret
  00110101 11100010 - (Reserved for future use)
  10011010 11110001 - (Reserved for future use)
  01001101 01111000 - (Reserved for future use)
  00100100 10111101 - (Reserved for future use)
  00010011 01011110 - (Reserved for future use)
  10001001 10101111 - (Reserved for future use)
  11000100 11010110 - (Reserved for future use)
  11100010 01101011 - (Reserved for future use)
```

Compartments (C field): 16 bits

An all zero value is used when the information transmitted is not compartmented. Other values for the compartments field may be obtained from the Defense Intelligence Agency.

Handling Restrictions (H field): 16 bits

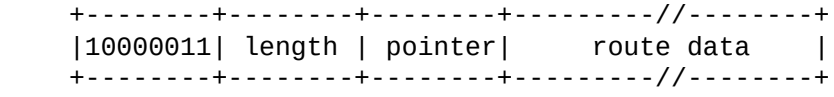
The values for the control and release markings are alphanumeric digraphs and are defined in the Defense Intelligence Agency Manual DIAM 65-19, "Standard Security Markings".

Transmission Control Code (TCC field): 24 bits

Provides a means to segregate traffic and define controlled communities of interest among subscribers. The TCC values are trigraphs, and are available from HQ DCA Code 530.

Must be copied on fragmentation. This option appears at most once in a datagram.

Loose Source and Record Route



The loose source and record route (LSRR) option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and pointer is increased by four.

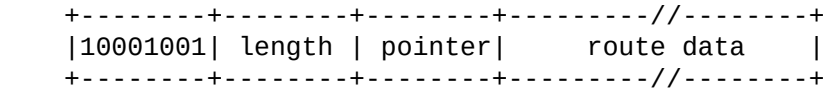
The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in the reverse of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the internet.

This option is a loose source route because the gateway or host IP is allowed to use any route of any number of other intermediate gateways to reach the next address in the route.

Must be copied on fragmentation. Appears at most once in a datagram.

Strict Source and Record Route



The strict source and record route (SSRR) option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next source address to be processed. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A route data is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is greater than the length, the source route is empty (and the

recorded route full) and the routing is to be based on the destination address field.

If the address in destination address field has been reached and the pointer is not greater than the length, the next address in the source route replaces the address in the destination address field, and the recorded route address replaces the source address just used, and pointer is increased by four.

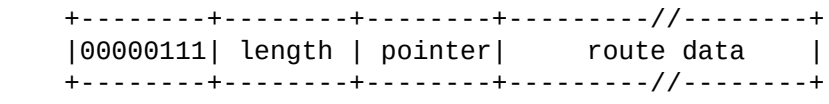
The recorded route address is the internet module's own internet address as known in the environment into which this datagram is being forwarded.

This procedure of replacing the source route with the recorded route (though it is in the reverse of the order it must be in to be used as a source route) means the option (and the IP header as a whole) remains a constant length as the datagram progresses through the internet.

This option is a strict source route because the gateway or host IP must send the datagram directly to the next address in the source route through only the directly connected network indicated in the next address to reach the next gateway or host specified in the route.

Must be copied on fragmentation. Appears at most once in a datagram.

Record Route



The record route option provides a means to record the route of an internet datagram.

The option begins with the option type code. The second octet is the option length which includes the option type code and the length octet, the pointer octet, and length-3 octets of route data. The third octet is the pointer into the route data indicating the octet which begins the next area to store a route address. The pointer is relative to this option, and the smallest legal value for the pointer is 4.

A recorded route is composed of a series of internet addresses. Each internet address is 32 bits or 4 octets. If the pointer is

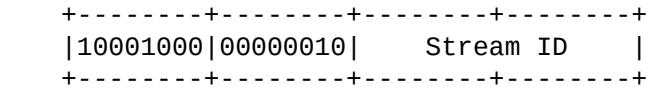
greater than the length, the recorded route data area is full. The originating host must compose this option with a large enough route data area to hold all the address expected. The size of the option does not change due to adding addresses. The intitial contents of the route data area must be zero.

When an internet module routes a datagram it checks to see if the record route option is present. If it is, it inserts its own internet address as known in the environment into which this datagram is being forwarded into the recorded route begining at the octet indicated by the pointer, and increments the pointer by four.

If the route data area is already full (the pointer exceeds the length) the datagram is forwarded without inserting the address into the recorded route. If there is some room but not enough room for a full address to be inserted, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host [3].

Not copied on fragmentation, goes in first fragment only. Appears at most once in a datagram.

Stream Identifier

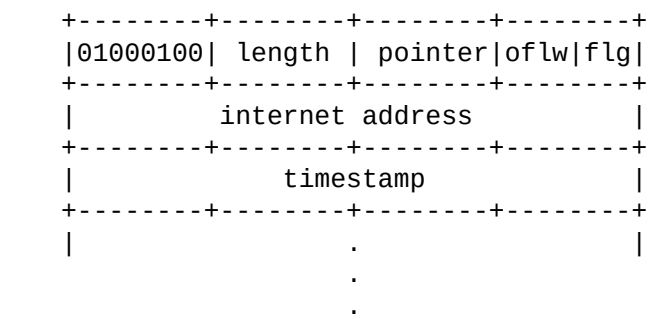


Type=136 Length=4

This option provides a way for the 16-bit SATNET stream identifier to be carried through networks that do not support the stream concept.

Must be copied on fragmentation. Appears at most once in a datagram.

Internet Timestamp



Type = 68

The Option Length is the number of octets in the option counting the type, length, pointer, and overflow/flag octets (maximum length 40).

The Pointer is the number of octets from the beginning of this option to the end of timestamps plus one (i.e., it points to the octet beginning the space for next timestamp). The smallest legal value is 5. The timestamp area is full when the pointer is greater than the length.

The Overflow (oflw) [4 bits] is the number of IP modules that cannot register timestamps due to lack of space.

The Flag (flg) [4 bits] values are

0 -- time stamps only, stored in consecutive 32-bit words,

1 -- each timestamp is preceded with internet address of the registering entity,

3 -- the internet address fields are prespecified. An IP module only registers its timestamp if it matches its own address with the next specified internet address.

The Timestamp is a right-justified, 32-bit timestamp in milliseconds since midnight UT. If the time is not available in milliseconds or cannot be provided with respect to midnight UT then any time may be inserted as a timestamp provided the high order bit of the timestamp field is set to one to indicate the use of a non-standard value.

The originating host must compose this option with a large enough timestamp data area to hold all the timestamp information expected. The size of the option does not change due to adding

timestamps. The intitial contents of the timestamp data area must be zero or internet address/zero pairs.

If the timestamp data area is already full (the pointer exceeds the length) the datagram is forwarded without inserting the timestamp, but the overflow count is incremented by one.

If there is some room but not enough room for a full timestamp to be inserted, or the overflow count itself overflows, the original datagram is considered to be in error and is discarded. In either case an ICMP parameter problem message may be sent to the source host [3].

The timestamp option is not copied upon fragmentation. It is carried in the first fragment. Appears at most once in a datagram.

Padding: variable

The internet header padding is used to ensure that the internet header ends on a 32 bit boundary. The padding is zero.

3.2. Discussion

The implementation of a protocol must be robust. Each implementation must expect to interoperate with others created by different individuals. While the goal of this specification is to be explicit about the protocol there is the possibility of differing interpretations. In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear).

The basic internet service is datagram oriented and provides for the fragmentation of datagrams at gateways, with reassembly taking place at the destination internet protocol module in the destination host. Of course, fragmentation and reassembly of datagrams within a network or by private agreement between the gateways of a network is also allowed since this is transparent to the internet protocols and the higher-level protocols. This transparent type of fragmentation and reassembly is termed "network-dependent" (or intranet) fragmentation and is not discussed further here.

Internet addresses distinguish sources and destinations to the host level and provide a protocol field as well. It is assumed that each protocol will provide for whatever multiplexing is necessary within a host.

Addressing

To provide for flexibility in assigning address to networks and allow for the large number of small to intermediate sized networks the interpretation of the address field is coded to specify a small number of networks with a large number of host, a moderate number of networks with a moderate number of hosts, and a large number of networks with a small number of hosts. In addition there is an escape code for extended addressing mode.

Address Formats:

High Order Bits	Format	Class
0	7 bits of net, 24 bits of host	a
10	14 bits of net, 16 bits of host	b
110	21 bits of net, 8 bits of host	c
111	escape to extended addressing mode	

A value of zero in the network field means this network. This is only used in certain ICMP messages. The extended addressing mode is undefined. Both of these features are reserved for future use.

The actual values assigned for network addresses is given in "Assigned Numbers" [9].

The local address, assigned by the local network, must allow for a single physical host to act as several distinct internet hosts. That is, there must be a mapping between internet host addresses and network/host interfaces that allows several internet addresses to correspond to one interface. It must also be allowed for a host to have several physical interfaces and to treat the datagrams from several of them as if they were all addressed to a single host.

Address mappings between internet addresses and addresses for ARPANET, SATNET, PRNET, and other networks are described in "Address Mappings" [5].

Fragmentation and Reassembly.

The internet identification field (ID) is used together with the source and destination address, and the protocol fields, to identify datagram fragments for reassembly.

The More Fragments flag bit (MF) is set if the datagram is not the last fragment. The Fragment Offset field identifies the fragment location, relative to the beginning of the original unfragmented datagram. Fragments are counted in units of 8 octets. The

fragmentation strategy is designed so than an unfragmented datagram has all zero fragmentation information (MF = 0, fragment offset = 0). If an internet datagram is fragmented, its data portion must be broken on 8 octet boundaries.

This format allows 2**13 = 8192 fragments of 8 octets each for a total of 65,536 octets. Note that this is consistent with the the datagram total length field (of course, the header is counted in the total length and not in the fragments).

When fragmentation occurs, some options are copied, but others remain with the first fragment only.

Every internet module must be able to forward a datagram of 68 octets without further fragmentation. This is because an internet header may be up to 60 octets, and the minimum fragment is 8 octets.

Every internet destination must be able to receive a datagram of 576 octets either in one piece or in fragments to be reassembled.

The fields which may be affected by fragmentation include:

(1) options field
(2) more fragments flag
(3) fragment offset
(4) internet header length field
(5) total length field
(6) header checksum

If the Don't Fragment flag (DF) bit is set, then internet fragmentation of this datagram is NOT permitted, although it may be discarded. This can be used to prohibit fragmentation in cases where the receiving host does not have sufficient resources to reassemble internet fragments.

One example of use of the Don't Fragment feature is to down line load a small host. A small host could have a boot strap program that accepts a datagram stores it in memory and then executes it.

The fragmentation and reassembly procedures are most easily described by examples. The following procedures are example implementations.

General notation in the following pseudo programs: "=<" means "less than or equal", "#" means "not equal", "=" means "equal", "<-" means "is set to". Also, "x to y" includes x and excludes y; for example, "4 to 7" would include 4, 5, and 6 (but not 7).

An Example Fragmentation Procedure

The maximum sized datagram that can be transmitted through the next network is called the maximum transmission unit (MTU).

If the total length is less than or equal the maximum transmission unit then submit this datagram to the next step in datagram processing; otherwise cut the datagram into two fragments, the first fragment being the maximum size, and the second fragment being the rest of the datagram. The first fragment is submitted to the next step in datagram processing, while the second fragment is submitted to this procedure in case it is still too large.

Notation:

```
FO    -  Fragment Offset
IHL   -  Internet Header Length
DF    -  Don't Fragment flag
MF    -  More Fragments flag
TL    -  Total Length
OFO   -  Old Fragment Offset
OIHL  -  Old Internet Header Length
OMF   -  Old More Fragments flag
OTL   -  Old Total Length
NFB   -  Number of Fragment Blocks
MTU   -  Maximum Transmission Unit
```

Procedure:

```
IF TL =< MTU THEN Submit this datagram to the next step
     in datagram processing ELSE IF DF = 1 THEN discard the
datagram ELSE
To produce the first fragment:
(1)  Copy the original internet header;
(2)  OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF;
(3)  NFB <- (MTU-IHL*4)/8;
(4)  Attach the first NFB*8 data octets;
(5)  Correct the header:
     MF <- 1;  TL <- (IHL*4)+(NFB*8);
     Recompute Checksum;
(6)  Submit this fragment to the next step in
     datagram processing;
To produce the second fragment:
(7)  Selectively copy the internet header (some options
     are not copied, see option definitions);
(8)  Append the remaining data;
(9)  Correct the header:
     IHL <- (((OIHL*4)-(length of options not copied))+3)/4;
```

```
        TL <- OTL - NFB*8 - (OIHL-IHL)*4);
        FO <- OFO + NFB;  MF <- OMF;  Recompute Checksum;
   (10) Submit this fragment to the fragmentation test; DONE.
```

In the above procedure each fragment (except the last) was made the maximum allowable size. An alternative might produce less than the maximum size datagrams. For example, one could implement a fragmentation procedure that repeatly divided large datagrams in half until the resulting fragments were less than the maximum transmission unit size.

An Example Reassembly Procedure

For each datagram the buffer identifier is computed as the concatenation of the source, destination, protocol, and identification fields. If this is a whole datagram (that is both the fragment offset and the more fragments fields are zero), then any reassembly resources associated with this buffer identifier are released and the datagram is forwarded to the next step in datagram processing.

If no other fragment with this buffer identifier is on hand then reassembly resources are allocated. The reassembly resources consist of a data buffer, a header buffer, a fragment block bit table, a total data length field, and a timer. The data from the fragment is placed in the data buffer according to its fragment offset and length, and bits are set in the fragment block bit table corresponding to the fragment blocks received.

If this is the first fragment (that is the fragment offset is zero) this header is placed in the header buffer. If this is the last fragment (that is the more fragments field is zero) the total data length is computed. If this fragment completes the datagram (tested by checking the bits set in the fragment block table), then the datagram is sent to the next step in datagram processing; otherwise the timer is set to the maximum of the current timer value and the value of the time to live field from this fragment; and the reassembly routine gives up control.

If the timer runs out, the all reassembly resources for this buffer identifier are released. The initial setting of the timer is a lower bound on the reassembly waiting time. This is because the waiting time will be increased if the Time to Live in the arriving fragment is greater than the current timer value but will not be decreased if it is less. The maximum this timer value could reach is the maximum time to live (approximately 4.25 minutes). The current recommendation for the initial timer setting is 15 seconds. This may be changed as experience with

this protocol accumulates. Note that the choice of this parameter value is related to the buffer capacity available and the data rate of the transmission medium; that is, data rate times timer value equals buffer size (e.g., 10Kb/s X 15s = 150Kb).

Notation:

```
  FO    -  Fragment Offset
  IHL   -  Internet Header Length
  MF    -  More Fragments flag
  TTL   -  Time To Live
  NFB   -  Number of Fragment Blocks
  TL    -  Total Length
  TDL   -  Total Data Length
  BUFID -  Buffer Identifier
  RCVBT -  Fragment Received Bit Table
  TLB   -  Timer Lower Bound
```

Procedure:

```
  (1)  BUFID <- source|destination|protocol|identification;
  (2)  IF FO = 0 AND MF = 0
  (3)     THEN IF buffer with BUFID is allocated
  (4)             THEN flush all reassembly for this BUFID;
  (5)          Submit datagram to next step; DONE.
  (6)     ELSE IF no buffer with BUFID is allocated
  (7)             THEN allocate reassembly resources
                       with BUFID;
                       TIMER <- TLB; TDL <- 0;
  (8)          put data from fragment into data buffer with
               BUFID from octet FO*8 to
                                 octet (TL-(IHL*4))+FO*8;
  (9)          set RCVBT bits from FO
                                to FO+((TL-(IHL*4)+7)/8);
  (10)         IF MF = 0 THEN TDL <- TL-(IHL*4)+(FO*8)
  (11)         IF FO = 0 THEN put header in header buffer
  (12)         IF TDL # 0
  (13)          AND all RCVBT bits from 0
                                     to (TDL+7)/8 are set
  (14)            THEN TL <- TDL+(IHL*4)
  (15)                 Submit datagram to next step;
  (16)                 free all reassembly resources
                       for this BUFID; DONE.
  (17)         TIMER <- MAX(TIMER,TTL);
  (18)         give up until next fragment or timer expires;
  (19) timer expires: flush all reassembly with this BUFID; DONE.
```

In the case that two or more fragments contain the same data

either identically or through a partial overlap, this procedure will use the more recently arrived copy in the data buffer and datagram delivered.

Identification

The choice of the Identifier for a datagram is based on the need to provide a way to uniquely identify the fragments of a particular datagram. The protocol module assembling fragments judges fragments to belong to the same datagram if they have the same source, destination, protocol, and Identifier. Thus, the sender must choose the Identifier to be unique for this source, destination pair and protocol for the time the datagram (or any fragment of it) could be alive in the internet.

It seems then that a sending protocol module needs to keep a table of Identifiers, one entry for each destination it has communicated with in the last maximum packet lifetime for the internet.

However, since the Identifier field allows 65,536 different values, some host may be able to simply use unique identifiers independent of destination.

It is appropriate for some higher level protocols to choose the identifier. For example, TCP protocol modules may retransmit an identical TCP segment, and the probability for correct reception would be enhanced if the retransmission carried the same identifier as the original transmission since fragments of either datagram could be used to construct a correct TCP segment.

Type of Service

The type of service (TOS) is for internet service quality selection. The type of service is specified along the abstract parameters precedence, delay, throughput, and reliability. These abstract parameters are to be mapped into the actual service parameters of the particular networks the datagram traverses.

Precedence. An independent measure of the importance of this datagram.

Delay. Prompt delivery is important for datagrams with this indication.

Throughput. High data rate is important for datagrams with this indication.

Reliability. A higher level of effort to ensure delivery is important for datagrams with this indication.

For example, the ARPANET has a priority bit, and a choice between "standard" messages (type 0) and "uncontrolled" messages (type 3), (the choice between single packet and multipacket messages can also be considered a service parameter). The uncontrolled messages tend to be less reliably delivered and suffer less delay. Suppose an internet datagram is to be sent through the ARPANET. Let the internet type of service be given as:

```
Precedence:    5
Delay:         0
Throughput:    1
Reliability:   1
```

In this example, the mapping of these parameters to those available for the ARPANET would be to set the ARPANET priority bit on since the Internet precedence is in the upper half of its range, to select standard messages since the throughput and reliability requirements are indicated and delay is not. More details are given on service mappings in "Service Mappings" [8].

Time to Live

The time to live is set by the sender to the maximum time the datagram is allowed to be in the internet system. If the datagram is in the internet system longer than the time to live, then the datagram must be destroyed.

This field must be decreased at each point that the internet header is processed to reflect the time spent processing the datagram. Even if no local information is available on the time actually spent, the field must be decremented by 1. The time is measured in units of seconds (i.e. the value 1 means one second). Thus, the maximum time to live is 255 seconds or 4.25 minutes. Since every module that processes a datagram must decrease the TTL by at least one even if it process the datagram in less than a second, the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

Some higher level reliable connection protocols are based on assumptions that old duplicate datagrams will not arrive after a certain time elapses. The TTL is a way for such protocols to have an assurance that their assumption is met.

Options

The options are optional in each datagram, but required in implementations. That is, the presence or absence of an option is the choice of the sender, but each internet module must be able to parse every option. There can be several options present in the option field.

The options might not end on a 32-bit boundary. The internet header must be filled out with octets of zeros. The first of these would be interpreted as the end-of-options option, and the remainder as internet header padding.

Every internet module must be able to act on every option. The Security Option is required if classified, restricted, or compartmented traffic is to be passed.

Checksum

The internet header checksum is recomputed if the internet header is changed. For example, a reduction of the time to live, additions or changes to internet options, or due to fragmentation. This checksum at the internet level is intended to protect the internet header fields from transmission errors.

There are some applications where a few data bit errors are acceptable while retransmission delays are not. If the internet protocol enforced data correctness such applications could not be supported.

Errors

Internet protocol errors may be reported via the ICMP messages [3].

3.3. Interfaces

The functional description of user interfaces to the IP is, at best, fictional, since every operating system will have different facilities. Consequently, we must warn readers that different IP implementations may have different user interfaces. However, all IPs must provide a certain minimum set of services to guarantee that all IP implementations can support the same protocol hierarchy. This section specifies the functional interfaces required of all IP implementations.

Internet protocol interfaces on one side to the local network and on the other side to either a higher level protocol or an application program. In the following, the higher level protocol or application

program (or even a gateway program) will be called the "user" since it is using the internet module. Since internet protocol is a datagram protocol, there is minimal memory or state maintained between datagram transmissions, and each call on the internet protocol module by the user supplies all information necessary for the IP to perform the service requested.

An Example Upper Level Interface

The following two example calls satisfy the requirements for the user to internet protocol module communication ("=>" means returns):

SEND (src, dst, prot, TOS, TTL, BufPTR, len, Id, DF, opt => result)

where:

src = source address
dst = destination address
prot = protocol
TOS = type of service
TTL = time to live
BufPTR = buffer pointer
len = length of buffer
Id = Identifier
DF = Don't Fragment
opt = option data
result = response
OK = datagram sent ok
Error = error in arguments or local network error

Note that the precedence is included in the TOS and the security/compartment is passed as an option.

RECV (BufPTR, prot, => result, src, dst, TOS, len, opt)

where:

BufPTR = buffer pointer
prot = protocol
result = response
OK = datagram received ok
Error = error in arguments
len = length of buffer
src = source address
dst = destination address
TOS = type of service
opt = option data

When the user sends a datagram, it executes the SEND call supplying all the arguments. The internet protocol module, on receiving this call, checks the arguments and prepares and sends the message. If the arguments are good and the datagram is accepted by the local network, the call returns successfully. If either the arguments are bad, or the datagram is not accepted by the local network, the call returns unsuccessfully. On unsuccessful returns, a reasonable report must be made as to the cause of the problem, but the details of such reports are up to individual implementations.

When a datagram arrives at the internet protocol module from the local network, either there is a pending RECV call from the user addressed or there is not. In the first case, the pending call is satisfied by passing the information from the datagram to the user. In the second case, the user addressed is notified of a pending datagram. If the user addressed does not exist, an ICMP error message is returned to the sender, and the data is discarded.

The notification of a user may be via a pseudo interrupt or similar mechanism, as appropriate in the particular operating system environment of the implementation.

A user's RECV call may then either be immediately satisfied by a pending datagram, or the call may be pending until a datagram arrives.

The source address is included in the send call in case the sending host has several addresses (multiple physical connections or logical addresses). The internet module must check to see that the source address is one of the legal address for this host.

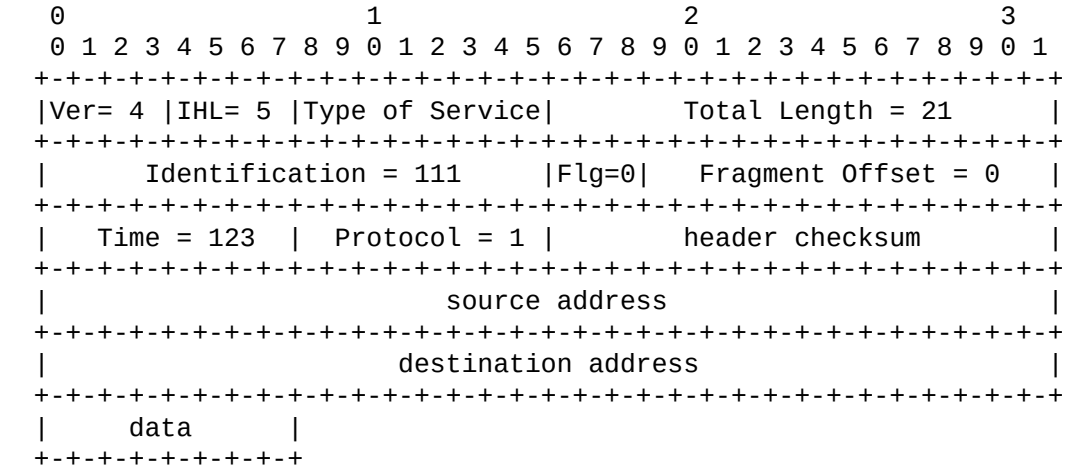
An implementation may also allow or require a call to the internet module to indicate interest in or reserve exclusive use of a class of datagrams (e.g., all those with a certain value in the protocol field).

This section functionally characterizes a USER/IP interface. The notation used is similar to most procedure of function calls in high level languages, but this usage is not meant to rule out trap type service calls (e.g., SVCs, UUOs, EMTs), or any other form of interprocess communication.

APPENDIX A: Examples & Scenarios

Example 1:

  This is an example of the minimal data carrying internet datagram:



Example Internet Datagram

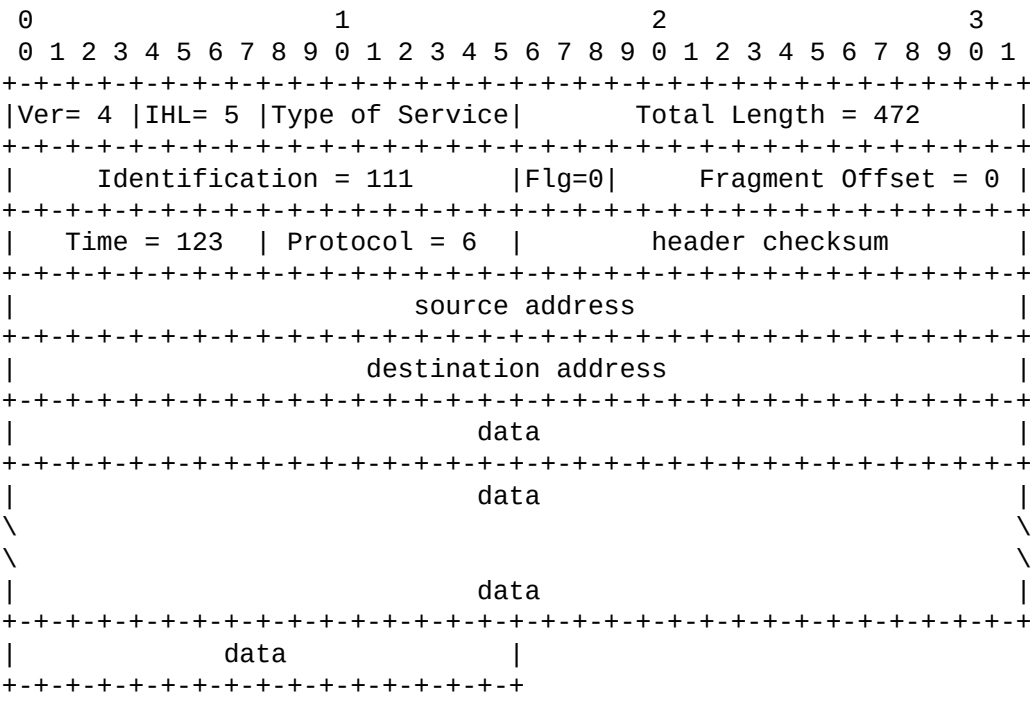
Figure 5.

  Note that each tick mark represents one bit position.

  This is a internet datagram in version 4 of internet protocol; the internet header consists of five 32 bit words, and the total length of the datagram is 21 octets. This datagram is a complete datagram (not a fragment).

Example 2:

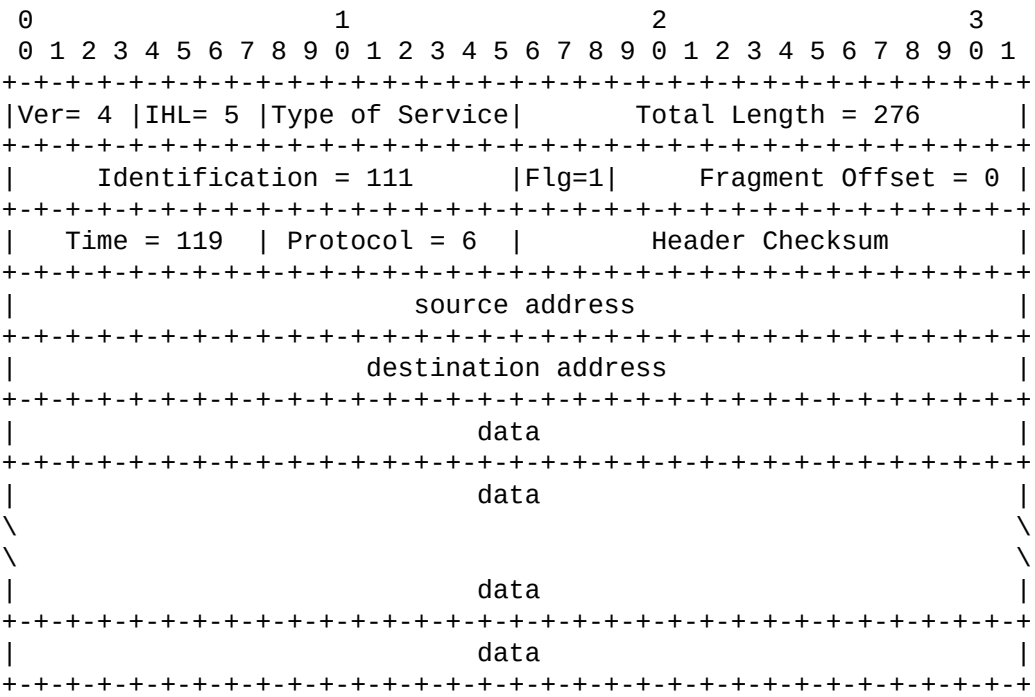
In this example, we show first a moderate size internet datagram (452 data octets), then two internet fragments that might result from the fragmentation of this datagram if the maximum sized transmission allowed were 280 octets.



Example Internet Datagram

Figure 6.

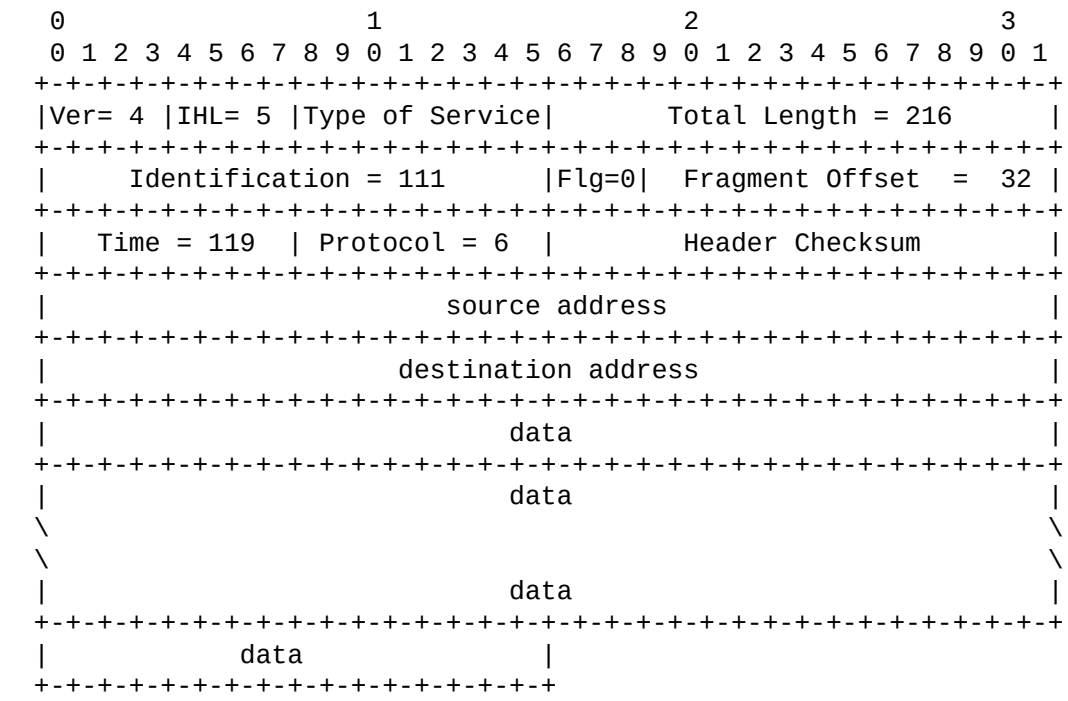
Now the first fragment that results from splitting the datagram after 256 data octets.



Example Internet Fragment

Figure 7.

And the second fragment.

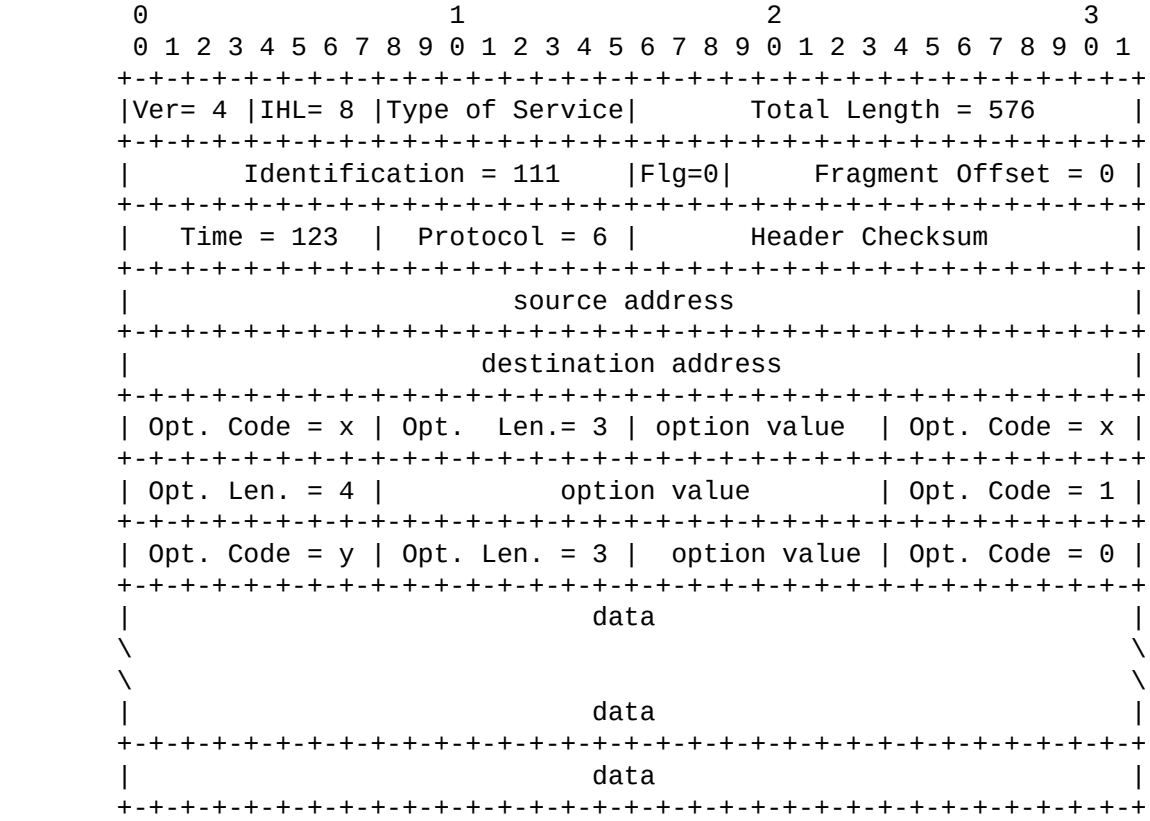


Example Internet Fragment

Figure 8.

Example 3:

Here, we show an example of a datagram containing options:

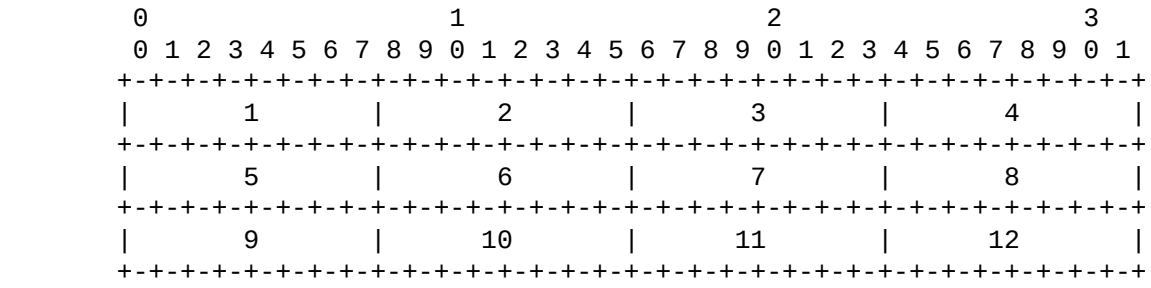


Example Internet Datagram

Figure 9.

APPENDIX B: Data Transmission Order

The order of transmission of the header and data described in this document is resolved to the octet level. Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram the octets are transmitted in the order they are numbered.



Transmission Order of Bytes

Figure 10.

Whenever an octet represents a numeric quantity the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).

Significance of Bits

Figure 11.

Similarly, whenever a multi-octet field represents a numeric quantity the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

GLOSSARY

1822
BBN Report 1822, "The Specification of the Interconnection of a Host and an IMP". The specification of interface between a host and the ARPANET.

ARPANET leader
The control information on an ARPANET message at the host-IMP interface.

ARPANET message
The unit of transmission between a host and an IMP in the ARPANET. The maximum size is about 1012 octets (8096 bits).

ARPANET packet
A unit of transmission used internally in the ARPANET between IMPs. The maximum size is about 126 octets (1008 bits).

Destination
The destination address, an internet header field.

DF
The Don’t Fragment bit carried in the flags field.

Flags
An internet header field carrying various control flags.

Fragment Offset
This internet header field indicates where in the internet datagram a fragment belongs.

GGP
Gateway to Gateway Protocol, the protocol used primarily between gateways to control routing and other gateway functions.

header
Control information at the beginning of a message, segment, datagram, packet or block of data.

ICMP
Internet Control Message Protocol, implemented in the internet module, the ICMP is used from gateways to hosts and between hosts to report errors and make routing suggestions.

Identification
: An internet header field carrying the identifying value assigned by the sender to aid in assembling the fragments of a datagram.

IHL
: The internet header field Internet Header Length is the length of the internet header measured in 32 bit words.

IMP
: The Interface Message Processor, the packet switch of the ARPANET.

Internet Address
: A four octet (32 bit) source or destination address consisting of a Network field and a Local Address field.

internet datagram
: The unit of data exchanged between a pair of internet modules (includes the internet header).

internet fragment
: A portion of the data of an internet datagram with an internet header.

Local Address
: The address of a host within a network. The actual mapping of an internet local address on to the host addresses in a network is quite general, allowing for many to one mappings.

MF
: The More-Fragments Flag carried in the internet header flags field.

module
: An implementation, usually in software, of a protocol or other procedure.

more-fragments flag
: A flag indicating whether or not this internet datagram contains the end of an internet datagram, carried in the internet header Flags field.

NFB
: The Number of Fragment Blocks in a the data portion of an internet fragment. That is, the length of a portion of data measured in 8 octet units.

octet
An eight bit byte.

Options
The internet header Options field may contain several options, and each option may be several octets in length.

Padding
The internet header Padding field is used to ensure that the data begins on 32 bit word boundary. The padding is zero.

Protocol
In this document, the next higher level protocol identifier, an internet header field.

Rest
The local address portion of an Internet Address.

Source
The source address, an internet header field.

TCP
Transmission Control Protocol: A host-to-host protocol for reliable communication in internet environments.

TCP Segment
The unit of data exchanged between TCP modules (including the TCP header).

TFTP
Trivial File Transfer Protocol: A simple file transfer protocol built on UDP.

Time to Live
An internet header field which indicates the upper bound on how long this internet datagram may exist.

TOS
Type of Service

Total Length
The internet header field Total Length is the length of the datagram in octets including internet header and data.

TTL
Time to Live

Type of Service
: An internet header field which indicates the type (or quality) of service for this internet datagram.

UDP
: User Datagram Protocol: A user level protocol for transaction oriented applications.

User
: The user of the internet protocol. This may be a higher level protocol module, an application program, or a gateway program.

Version
: The Version field indicates the format of the internet header.

REFERENCES

[1] Cerf, V., "The Catenet Model for Internetworking," Information Processing Techniques Office, Defense Advanced Research Projects Agency, IEN 48, July 1978.

[2] Bolt Beranek and Newman, "Specification for the Interconnection of a Host and an IMP," BBN Technical Report 1822, Revised May 1978.

[3] Postel, J., "Internet Control Message Protocol - DARPA Internet Program Protocol Specification," RFC 792, USC/Information Sciences Institute, September 1981.

[4] Shoch, J., "Inter-Network Naming, Addressing, and Routing," COMPCON, IEEE Computer Society, Fall 1978.

[5] Postel, J., "Address Mappings," RFC 796, USC/Information Sciences Institute, September 1981.

[6] Shoch, J., "Packet Fragmentation in Inter-Network Protocols," Computer Networks, v. 3, n. 1, February 1979.

[7] Strazisar, V., "How to Build a Gateway", IEN 109, Bolt Beranek and Newman, August 1979.

[8] Postel, J., "Service Mappings," RFC 795, USC/Information Sciences Institute, September 1981.

[9] Postel, J., "Assigned Numbers," RFC 790, USC/Information Sciences Institute, September 1981.

Network Working Group
Request for Comments: 1034
Obsoletes: RFCs 882, 883, 973

P. Mockapetris
ISI

November 1987

DOMAIN NAMES - CONCEPTS AND FACILITIES

1. STATUS OF THIS MEMO

This RFC is an introduction to the Domain Name System (DNS), and omits many details which can be found in a companion RFC, "Domain Names - Implementation and Specification" [RFC-1035]. That RFC assumes that the reader is familiar with the concepts discussed in this memo.

A subset of DNS functions and data types constitute an official protocol. The official protocol includes standard queries and their responses and most of the Internet class data formats (e.g., host addresses).

However, the domain system is intentionally extensible. Researchers are continuously proposing, implementing and experimenting with new data types, query types, classes, functions, etc. Thus while the components of the official protocol are expected to stay essentially unchanged and operate as a production service, experimental behavior should always be expected in extensions beyond the official protocol. Experimental or obsolete features are clearly marked in these RFCs, and such information should be used with caution.

The reader is especially cautioned not to depend on the values which appear in examples to be current or complete, since their purpose is primarily pedagogical. Distribution of this memo is unlimited.

2. INTRODUCTION

This RFC introduces domain style names, their use for Internet mail and host address support, and the protocols and servers used to implement domain name facilities.

2.1. The history of domain names

The impetus for the development of the domain system was growth in the Internet:

- Host name to address mappings were maintained by the Network Information Center (NIC) in a single file (HOSTS.TXT) which was FTPed by all hosts [RFC-952, RFC-953]. The total network

bandwidth consumed in distributing a new version by this scheme is proportional to the square of the number of hosts in the network, and even when multiple levels of FTP are used, the outgoing FTP load on the NIC host is considerable. Explosive growth in the number of hosts didn't bode well for the future.

- The network population was also changing in character. The timeshared hosts that made up the original ARPANET were being replaced with local networks of workstations. Local organizations were administering their own names and addresses, but had to wait for the NIC to change HOSTS.TXT to make changes visible to the Internet at large. Organizations also wanted some local structure on the name space.

- The applications on the Internet were getting more sophisticated and creating a need for general purpose name service.

The result was several ideas about name spaces and their management [IEN-116, RFC-799, RFC-819, RFC-830]. The proposals varied, but a common thread was the idea of a hierarchical name space, with the hierarchy roughly corresponding to organizational structure, and names using "." as the character to mark the boundary between hierarchy levels. A design using a distributed database and generalized resources was described in [RFC-882, RFC-883]. Based on experience with several implementations, the system evolved into the scheme described in this memo.

The terms "domain" or "domain name" are used in many contexts beyond the DNS described here. Very often, the term domain name is used to refer to a name with structure indicated by dots, but no relation to the DNS. This is particularly true in mail addressing [Quarterman 86].

2.2. DNS design goals

The design goals of the DNS influence its structure. They are:

- The primary goal is a consistent name space which will be used for referring to resources. In order to avoid the problems caused by ad hoc encodings, names should not be required to contain network identifiers, addresses, routes, or similar information as part of the name.

- The sheer size of the database and frequency of updates suggest that it must be maintained in a distributed manner, with local caching to improve performance. Approaches that

attempt to collect a consistent copy of the entire database will become more and more expensive and difficult, and hence should be avoided. The same principle holds for the structure of the name space, and in particular mechanisms for creating and deleting names; these should also be distributed.

- Where there tradeoffs between the cost of acquiring data, the speed of updates, and the accuracy of caches, the source of the data should control the tradeoff.

- The costs of implementing such a facility dictate that it be generally useful, and not restricted to a single application. We should be able to use names to retrieve host addresses, mailbox data, and other as yet undetermined information. All data associated with a name is tagged with a type, and queries can be limited to a single type.

- Because we want the name space to be useful in dissimilar networks and applications, we provide the ability to use the same name space with different protocol families or management. For example, host address formats differ between protocols, though all protocols have the notion of address. The DNS tags all data with a class as well as the type, so that we can allow parallel use of different formats for data of type address.

- We want name server transactions to be independent of the communications system that carries them. Some systems may wish to use datagrams for queries and responses, and only establish virtual circuits for transactions that need the reliability (e.g., database updates, long transactions); other systems will use virtual circuits exclusively.

- The system should be useful across a wide spectrum of host capabilities. Both personal computers and large timeshared hosts should be able to use the system, though perhaps in different ways.

2.3. Assumptions about usage

The organization of the domain system derives from some assumptions about the needs and usage patterns of its user community and is designed to avoid many of the the complicated problems found in general purpose database systems.

The assumptions are:

- The size of the total database will initially be proportional

to the number of hosts using the system, but will eventually grow to be proportional to the number of users on those hosts as mailboxes and other information are added to the domain system.

- Most of the data in the system will change very slowly (e.g., mailbox bindings, host addresses), but that the system should be able to deal with subsets that change more rapidly (on the order of seconds or minutes).

- The administrative boundaries used to distribute responsibility for the database will usually correspond to organizations that have one or more hosts. Each organization that has responsibility for a particular set of domains will provide redundant name servers, either on the organization's own hosts or other hosts that the organization arranges to use.

- Clients of the domain system should be able to identify trusted name servers they prefer to use before accepting referrals to name servers outside of this "trusted" set.

- Access to information is more critical than instantaneous updates or guarantees of consistency. Hence the update process allows updates to percolate out through the users of the domain system rather than guaranteeing that all copies are simultaneously updated. When updates are unavailable due to network or host failure, the usual course is to believe old information while continuing efforts to update it. The general model is that copies are distributed with timeouts for refreshing. The distributor sets the timeout value and the recipient of the distribution is responsible for performing the refresh. In special situations, very short intervals can be specified, or the owner can prohibit copies.

- In any system that has a distributed database, a particular name server may be presented with a query that can only be answered by some other server. The two general approaches to dealing with this problem are "recursive", in which the first server pursues the query for the client at another server, and "iterative", in which the server refers the client to another server and lets the client pursue the query. Both approaches have advantages and disadvantages, but the iterative approach is preferred for the datagram style of access. The domain system requires implementation of the iterative approach, but allows the recursive approach as an option.

The domain system assumes that all data originates in master files scattered through the hosts that use the domain system. These master files are updated by local system administrators. Master files are text files that are read by a local name server, and hence become available through the name servers to users of the domain system. The user programs access name servers through standard programs called resolvers.

The standard format of master files allows them to be exchanged between hosts (via FTP, mail, or some other mechanism); this facility is useful when an organization wants a domain, but doesn't want to support a name server. The organization can maintain the master files locally using a text editor, transfer them to a foreign host which runs a name server, and then arrange with the system administrator of the name server to get the files loaded.

Each host's name servers and resolvers are configured by a local system administrator [RFC-1033]. For a name server, this configuration data includes the identity of local master files and instructions on which non-local master files are to be loaded from foreign servers. The name server uses the master files or copies to load its zones. For resolvers, the configuration data identifies the name servers which should be the primary sources of information.

The domain system defines procedures for accessing the data and for referrals to other name servers. The domain system also defines procedures for caching retrieved data and for periodic refreshing of data defined by the system administrator.

The system administrators provide:

- The definition of zone boundaries.

- Master files of data.

- Updates to master files.

- Statements of the refresh policies desired.

The domain system provides:

- Standard formats for resource data.

- Standard methods for querying the database.

- Standard methods for name servers to refresh local data from foreign name servers.

2.4. Elements of the DNS

The DNS has three major components:

- The DOMAIN NAME SPACE and RESOURCE RECORDS, which are specifications for a tree structured name space and data associated with the names. Conceptually, each node and leaf of the domain name space tree names a set of information, and query operations are attempts to extract specific types of information from a particular set. A query names the domain name of interest and describes the type of resource information that is desired. For example, the Internet uses some of its domain names to identify hosts; queries for address resources return Internet host addresses.

- NAME SERVERS are server programs which hold information about the domain tree's structure and set information. A name server may cache structure or set information about any part of the domain tree, but in general a particular name server has complete information about a subset of the domain space, and pointers to other name servers that can be used to lead to information from any part of the domain tree. Name servers know the parts of the domain tree for which they have complete information; a name server is said to be an AUTHORITY for these parts of the name space. Authoritative information is organized into units called ZONEs, and these zones can be automatically distributed to the name servers which provide redundant service for the data in a zone.

- RESOLVERS are programs that extract information from name servers in response to client requests. Resolvers must be able to access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers. A resolver will typically be a system routine that is directly accessible to user programs; hence no protocol is necessary between the resolver and the user program.

These three components roughly correspond to the three layers or views of the domain system:

- From the user's point of view, the domain system is accessed through a simple procedure or OS call to a local resolver. The domain space consists of a single tree and the user can request information from any section of the tree.

- From the resolver's point of view, the domain system is composed of an unknown number of name servers. Each name

server has one or more pieces of the whole domain tree's data, but the resolver views each of these databases as essentially static.

- From a name server's point of view, the domain system consists of separate sets of local information called zones. The name server has local copies of some of the zones. The name server must periodically refresh its zones from master copies in local files or foreign name servers. The name server must concurrently process queries that arrive from resolvers.

In the interests of performance, implementations may couple these functions. For example, a resolver on the same machine as a name server might share a database consisting of the the zones managed by the name server and the cache managed by the resolver.

3. DOMAIN NAME SPACE and RESOURCE RECORDS

3.1. Name space specifications and terminology

The domain name space is a tree structure. Each node and leaf on the tree corresponds to a resource set (which may be empty). The domain system makes no distinctions between the uses of the interior nodes and leaves, and this memo uses the term "node" to refer to both.

Each node has a label, which is zero to 63 octets in length. Brother nodes may not have the same label, although the same label can be used for nodes which are not brothers. One label is reserved, and that is the null (i.e., zero length) label used for the root.

The domain name of a node is the list of the labels on the path from the node to the root of the tree. By convention, the labels that compose a domain name are printed or read left to right, from the most specific (lowest, farthest from the root) to the least specific (highest, closest to the root).

Internally, programs that manipulate domain names should represent them as sequences of labels, where each label is a length octet followed by an octet string. Because all domain names end at the root, which has a null string for a label, these internal representations can use a length byte of zero to terminate a domain name.

By convention, domain names can be stored with arbitrary case, but domain name comparisons for all present domain functions are done in a case-insensitive manner, assuming an ASCII character set, and a high order zero bit. This means that you are free to create a node with label "A" or a node with label "a", but not both as brothers; you could refer to either using "a" or "A". When you receive a domain name or

label, you should preserve its case. The rationale for this choice is that we may someday need to add full binary domain names for new services; existing services would not be changed.

When a user needs to type a domain name, the length of each label is omitted and the labels are separated by dots ("."). Since a complete domain name ends with the root label, this leads to a printed form which ends in a dot. We use this property to distinguish between:

- a character string which represents a complete domain name (often called "absolute"). For example, "poneria.ISI.EDU."

- a character string that represents the starting labels of a domain name which is incomplete, and should be completed by local software using knowledge of the local domain (often called "relative"). For example, "poneria" used in the ISI.EDU domain.

Relative names are either taken relative to a well known origin, or to a list of domains used as a search list. Relative names appear mostly at the user interface, where their interpretation varies from implementation to implementation, and in master files, where they are relative to a single origin domain name. The most common interpretation uses the root "." as either the single origin or as one of the members of the search list, so a multi-label relative name is often one where the trailing dot has been omitted to save typing.

To simplify implementations, the total number of octets that represent a domain name (i.e., the sum of all label octets and label lengths) is limited to 255.

A domain is identified by a domain name, and consists of that part of the domain name space that is at or below the domain name which specifies the domain. A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name. For example, A.B.C.D is a subdomain of B.C.D, C.D, D, and " ".

3.2. Administrative guidelines on use

As a matter of policy, the DNS technical specifications do not mandate a particular tree structure or rules for selecting labels; its goal is to be as general as possible, so that it can be used to build arbitrary applications. In particular, the system was designed so that the name space did not have to be organized along the lines of network boundaries, name servers, etc. The rationale for this is not that the name space should have no implied semantics, but rather that the choice of implied semantics should be left open to be used for the problem at

hand, and that different parts of the tree can have different implied semantics. For example, the IN-ADDR.ARPA domain is organized and distributed by network and host address because its role is to translate from network or host numbers to names; NetBIOS domains [RFC-1001, RFC-1002] are flat because that is appropriate for that application.

However, there are some guidelines that apply to the "normal" parts of the name space used for hosts, mailboxes, etc., that will make the name space more uniform, provide for growth, and minimize problems as software is converted from the older host table. The political decisions about the top levels of the tree originated in RFC-920. Current policy for the top levels is discussed in [RFC-1032]. MILNET conversion issues are covered in [RFC-1031].

Lower domains which will eventually be broken into multiple zones should provide branching at the top of the domain so that the eventual decomposition can be done without renaming. Node labels which use special characters, leading digits, etc., are likely to break older software which depends on more restrictive choices.

3.3. Technical guidelines on use

Before the DNS can be used to hold naming information for some kind of object, two needs must be met:

- A convention for mapping between object names and domain names. This describes how information about an object is accessed.

- RR types and data formats for describing the object.

These rules can be quite simple or fairly complex. Very often, the designer must take into account existing formats and plan for upward compatibility for existing usage. Multiple mappings or levels of mapping may be required.

For hosts, the mapping depends on the existing syntax for host names which is a subset of the usual text representation for domain names, together with RR formats for describing host addresses, etc. Because we need a reliable inverse mapping from address to host name, a special mapping for addresses into the IN-ADDR.ARPA domain is also defined.

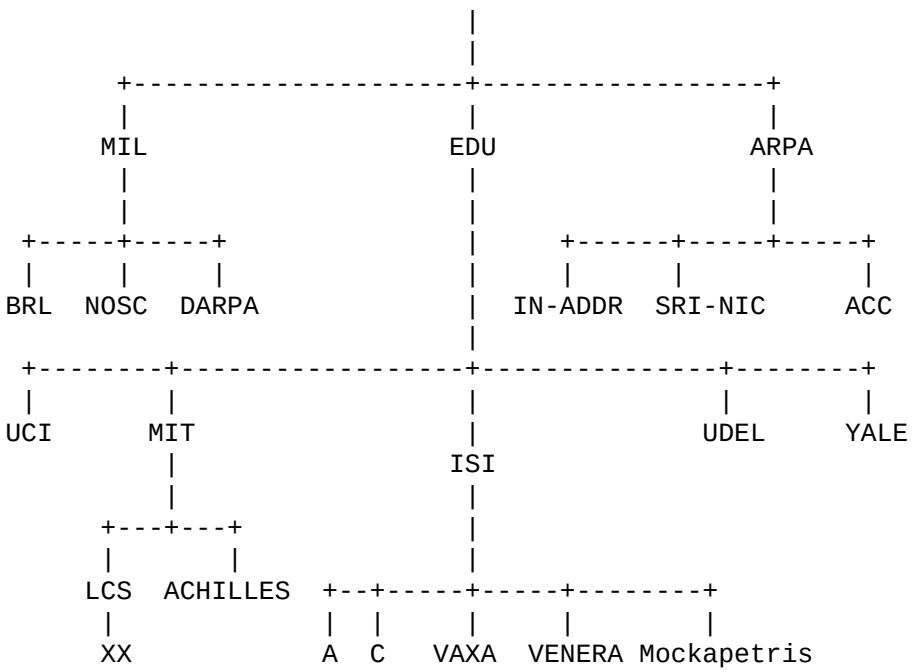
For mailboxes, the mapping is slightly more complex. The usual mail address <local-part>@<mail-domain> is mapped into a domain name by converting <local-part> into a single label (regardles of dots it contains), converting <mail-domain> into a domain name using the usual text format for domain names (dots denote label breaks), and concatenating the two to form a single domain name. Thus the mailbox

HOSTMASTER@SRI-NIC.ARPA is represented as a domain name by HOSTMASTER.SRI-NIC.ARPA. An appreciation for the reasons behind this design also must take into account the scheme for mail exchanges [RFC-974].

The typical user is not concerned with defining these rules, but should understand that they usually are the result of numerous compromises between desires for upward compatibility with old usage, interactions between different object definitions, and the inevitable urge to add new features when defining the rules. The way the DNS is used to support some object is often more crucial than the restrictions inherent in the DNS.

3.4. Example name space

The following figure shows a part of the current domain name space, and is used in many examples in this RFC. Note that the tree is a very small subset of the actual name space.



In this example, the root domain has three immediate subdomains: MIL, EDU, and ARPA. The LCS.MIT.EDU domain has one immediate subdomain named XX.LCS.MIT.EDU. All of the leaves are also domains.

3.5. Preferred name syntax

The DNS specifications attempt to be as general as possible in the rules

for constructing domain names. The idea is that the name of any existing object can be expressed as a domain name with minimal changes. However, when assigning a domain name for an object, the prudent user will select a name which satisfies both the rules of the domain system and any existing rules for the object, whether these rules are published or implied by existing programs.

For example, when naming a mail domain, the user should satisfy both the rules of this memo and those in RFC-822. When creating a new host name, the old rules for HOSTS.TXT should be followed. This avoids problems when old software is converted to use domain names.

The following syntax will result in fewer problems with many applications that use domain names (e.g., mail, TELNET).

<domain> ::= <subdomain> | " "

<subdomain> ::= <label> | <subdomain> "." <label>

<label> ::= <letter> [[<ldh-str>] <let-dig>]

<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>

<let-dig-hyp> ::= <let-dig> | "-"

<let-dig> ::= <letter> | <digit>

<letter> ::= any one of the 52 alphabetic characters A through Z in upper case and a through z in lower case

<digit> ::= any one of the ten digits 0 through 9

Note that while upper and lower case letters are allowed in domain names, no significance is attached to the case. That is, two names with the same spelling but different case are to be treated as if identical.

The labels must follow the rules for ARPANET host names. They must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. There are also some restrictions on the length. Labels must be 63 characters or less.

For example, the following strings identify hosts in the Internet:

A.ISI.EDU XX.LCS.MIT.EDU SRI-NIC.ARPA

3.6. Resource Records

A domain name identifies a node. Each node has a set of resource

information, which may be empty. The set of resource information associated with a particular name is composed of separate resource records (RRs). The order of RRs in a set is not significant, and need not be preserved by name servers, resolvers, or other parts of the DNS.

When we talk about a specific RR, we assume it has the following:

owner which is the domain name where the RR is found.

type which is an encoded 16 bit value that specifies the type of the resource in this resource record. Types refer to abstract resources.

This memo uses the following types:

A a host address

CNAME identifies the canonical name of an alias

HINFO identifies the CPU and OS used by a host

MX identifies a mail exchange for the domain. See [RFC-974 for details.

NS
the authoritative name server for the domain

PTR
a pointer to another part of the domain name space

SOA
identifies the start of a zone of authority]

class which is an encoded 16 bit value which identifies a protocol family or instance of a protocol.

This memo uses the following classes:

IN the Internet system

CH the Chaos system

TTL which is the time to live of the RR. This field is a 32 bit integer in units of seconds, an is primarily used by resolvers when they cache RRs. The TTL describes how long a RR can be cached before it should be discarded.

RDATA which is the type and sometimes class dependent data which describes the resource:

A	For the IN class, a 32 bit IP address For the CH class, a domain name followed by a 16 bit octal Chaos address.
CNAME	a domain name.
MX	a 16 bit preference value (lower is better) followed by a host name willing to act as a mail exchange for the owner domain.
NS	a host name.
PTR	a domain name.
SOA	several fields.

The owner name is often implicit, rather than forming an integral part of the RR. For example, many name servers internally form tree or hash structures for the name space, and chain RRs off nodes. The remaining RR parts are the fixed header (type, class, TTL) which is consistent for all RRs, and a variable part (RDATA) that fits the needs of the resource being described.

The meaning of the TTL field is a time limit on how long an RR can be kept in a cache. This limit does not apply to authoritative data in zones; it is also timed out, but by the refreshing policies for the zone. The TTL is assigned by the administrator for the zone where the data originates. While short TTLs can be used to minimize caching, and a zero TTL prohibits caching, the realities of Internet performance suggest that these times should be on the order of days for the typical host. If a change can be anticipated, the TTL can be reduced prior to the change to minimize inconsistency during the change, and then increased back to its former value following the change.

The data in the RDATA section of RRs is carried as a combination of binary strings and domain names. The domain names are frequently used as "pointers" to other data in the DNS.

3.6.1. Textual expression of RRs

RRs are represented in binary form in the packets of the DNS protocol, and are usually represented in highly encoded form when stored in a name server or resolver. In this memo, we adopt a style similar to that used

in master files in order to show the contents of RRs. In this format, most RRs are shown on a single line, although continuation lines are possible using parentheses.

The start of the line gives the owner of the RR. If a line begins with a blank, then the owner is assumed to be the same as that of the previous RR. Blank lines are often included for readability.

Following the owner, we list the TTL, type, and class of the RR. Class and type use the mnemonics defined above, and TTL is an integer before the type field. In order to avoid ambiguity in parsing, type and class mnemonics are disjoint, TTLs are integers, and the type mnemonic is always last. The IN class and TTL values are often omitted from examples in the interests of clarity.

The resource data or RDATA section of the RR are given using knowledge of the typical representation for the data.

For example, we might show the RRs carried in a message as:

```
    ISI.EDU.        MX      10 VENERA.ISI.EDU.
                    MX      10 VAXA.ISI.EDU.
    VENERA.ISI.EDU. A       128.9.0.32
                    A       10.1.0.52
    VAXA.ISI.EDU.   A       10.2.0.27
                    A       128.9.0.33
```

The MX RRs have an RDATA section which consists of a 16 bit number followed by a domain name. The address RRs use a standard IP address format to contain a 32 bit internet address.

This example shows six RRs, with two RRs at each of three domain names.

Similarly we might see:

```
    XX.LCS.MIT.EDU. IN      A       10.0.0.44
                    CH      A       MIT.EDU. 2420
```

This example shows two addresses for XX.LCS.MIT.EDU, each of a different class.

3.6.2. Aliases and canonical names

In existing systems, hosts and other resources often have several names that identify the same resource. For example, the names C.ISI.EDU and USC-ISIC.ARPA both identify the same host. Similarly, in the case of mailboxes, many organizations provide many names that actually go to the same mailbox; for example Mockapetris@C.ISI.EDU, Mockapetris@B.ISI.EDU,

and PVM@ISI.EDU all go to the same mailbox (although the mechanism behind this is somewhat complicated).

Most of these systems have a notion that one of the equivalent set of names is the canonical or primary name and all others are aliases.

The domain system provides such a feature using the canonical name (CNAME) RR. A CNAME RR identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR. If a CNAME RR is present at a node, no other data should be present; this ensures that the data for a canonical name and its aliases cannot be different. This rule also insures that a cached CNAME can be used without checking with an authoritative server for other RR types.

CNAME RRs cause special action in DNS software. When a name server fails to find a desired RR in the resource set associated with the domain name, it checks to see if the resource set consists of a CNAME record with a matching class. If so, the name server includes the CNAME record in the response and restarts the query at the domain name specified in the data field of the CNAME record. The one exception to this rule is that queries which match the CNAME type are not restarted.

For example, suppose a name server was processing a query with for USC-ISIC.ARPA, asking for type A information, and had the following resource records:

```
    USC-ISIC.ARPA   IN      CNAME   C.ISI.EDU

    C.ISI.EDU       IN      A       10.0.0.52
```

Both of these RRs would be returned in the response to the type A query, while a type CNAME or * query should return just the CNAME.

Domain names in RRs which point at another name should always point at the primary name and not the alias. This avoids extra indirections in accessing information. For example, the address to name RR for the above host should be:

```
    52.0.0.10.IN-ADDR.ARPA  IN      PTR     C.ISI.EDU
```

rather than pointing at USC-ISIC.ARPA. Of course, by the robustness principle, domain software should not fail when presented with CNAME chains or loops; CNAME chains should be followed and CNAME loops signalled as an error.

3.7. Queries

Queries are messages which may be sent to a name server to provoke a

response. In the Internet, queries are carried in UDP datagrams or over TCP connections. The response by the name server either answers the question posed in the query, refers the requester to another set of name servers, or signals some error condition.

In general, the user does not generate queries directly, but instead makes a request to a resolver which in turn sends one or more queries to name servers and deals with the error conditions and referrals that may result. Of course, the possible questions which can be asked in a query does shape the kind of service a resolver can provide.

DNS queries and responses are carried in a standard message format. The message format has a header containing a number of fixed fields which are always present, and four sections which carry query parameters and RRs.

The most important field in the header is a four bit field called an opcode which separates different queries. Of the possible 16 values, one (standard query) is part of the official protocol, two (inverse query and status query) are options, one (completion) is obsolete, and the rest are unassigned.

The four sections are:

Question	Carries the query name and other query parameters.
Answer	Carries RRs which directly answer the query.
Authority	Carries RRs which describe other authoritative servers. May optionally carry the SOA RR for the authoritative data in the answer section.
Additional	Carries RRs which may be helpful in using the RRs in the other sections.

Note that the content, but not the format, of these sections varies with header opcode.

3.7.1. Standard queries

A standard query specifies a target domain name (QNAME), query type (QTYPE), and query class (QCLASS) and asks for RRs which match. This type of query makes up such a vast majority of DNS queries that we use the term "query" to mean standard query unless otherwise specified. The QTYPE and QCLASS fields are each 16 bits long, and are a superset of defined types and classes.

The QTYPE field may contain:

<any type>	matches just that type. (e.g., A, PTR).
AXFR	special zone transfer QTYPE.
MAILB	matches all mail box related RRs (e.g. MB and MG).
*	matches all RR types.

The QCLASS field may contain:

<any class>	matches just that class (e.g., IN, CH).
*	matches aLL RR classes.

Using the query domain name, QTYPE, and QCLASS, the name server looks for matching RRs. In addition to relevant records, the name server may return RRs that point toward a name server that has the desired information or RRs that are expected to be useful in interpreting the relevant RRs. For example, a name server that doesn't have the requested information may know a name server that does; a name server that returns a domain name in a relevant RR may also return the RR that binds that domain name to an address.

For example, a mailer tying to send mail to Mockapetris@ISI.EDU might ask the resolver for mail information about ISI.EDU, resulting in a query for QNAME=ISI.EDU, QTYPE=MX, QCLASS=IN. The response's answer section would be:

```
    ISI.EDU.        MX      10 VENERA.ISI.EDU.
                    MX      10 VAXA.ISI.EDU.
```

while the additional section might be:

```
    VAXA.ISI.EDU.   A       10.2.0.27
                    A       128.9.0.33
    VENERA.ISI.EDU. A       10.1.0.52
                    A       128.9.0.32
```

Because the server assumes that if the requester wants mail exchange information, it will probably want the addresses of the mail exchanges soon afterward.

Note that the QCLASS=* construct requires special interpretation regarding authority. Since a particular name server may not know all of the classes available in the domain system, it can never know if it is authoritative for all classes. Hence responses to QCLASS=* queries can

never be authoritative.

3.7.2. Inverse queries (Optional)

Name servers may also support inverse queries that map a particular resource to a domain name or domain names that have that resource. For example, while a standard query might map a domain name to a SOA RR, the corresponding inverse query might map the SOA RR back to the domain name.

Implementation of this service is optional in a name server, but all name servers must at least be able to understand an inverse query message and return a not-implemented error response.

The domain system cannot guarantee the completeness or uniqueness of inverse queries because the domain system is organized by domain name rather than by host address or any other resource type. Inverse queries are primarily useful for debugging and database maintenance activities.

Inverse queries may not return the proper TTL, and do not indicate cases where the identified RR is one of a set (for example, one address for a host having multiple addresses). Therefore, the RRs returned in inverse queries should never be cached.

Inverse queries are NOT an acceptable method for mapping host addresses to host names; use the IN-ADDR.ARPA domain instead.

A detailed discussion of inverse queries is contained in [RFC-1035].

3.8. Status queries (Experimental)

To be defined.

3.9. Completion queries (Obsolete)

The optional completion services described in RFCs 882 and 883 have been deleted. Redesigned services may become available in the future, or the opcodes may be reclaimed for other use.

4. NAME SERVERS

4.1. Introduction

Name servers are the repositories of information that make up the domain database. The database is divided up into sections called zones, which are distributed among the name servers. While name servers can have several optional functions and sources of data, the essential task of a name server is to answer queries using data in its zones. By design,

name servers can answer queries in a simple manner; the response can always be generated using only local data, and either contains the answer to the question or a referral to other name servers "closer" to the desired information.

A given zone will be available from several name servers to insure its availability in spite of host or communication link failure. By administrative fiat, we require every zone to be available on at least two servers, and many zones have more redundancy than that.

A given name server will typically support one or more zones, but this gives it authoritative information about only a small section of the domain tree. It may also have some cached non-authoritative data about other parts of the tree. The name server marks its responses to queries so that the requester can tell whether the response comes from authoritative data or not.

4.2. How the database is divided into zones

The domain database is partitioned in two ways: by class, and by "cuts" made in the name space between nodes.

The class partition is simple. The database for any class is organized, delegated, and maintained separately from all other classes. Since, by convention, the name spaces are the same for all classes, the separate classes can be thought of as an array of parallel namespace trees. Note that the data attached to nodes will be different for these different parallel classes. The most common reasons for creating a new class are the necessity for a new data format for existing types or a desire for a separately managed version of the existing name space.

Within a class, "cuts" in the name space can be made between any two adjacent nodes. After all cuts are made, each group of connected name space is a separate zone. The zone is said to be authoritative for all names in the connected region. Note that the "cuts" in the name space may be in different places for different classes, the name servers may be different, etc.

These rules mean that every zone has at least one node, and hence domain name, for which it is authoritative, and all of the nodes in a particular zone are connected. Given, the tree structure, every zone has a highest node which is closer to the root than any other node in the zone. The name of this node is often used to identify the zone.

It would be possible, though not particularly useful, to partition the name space so that each domain name was in a separate zone or so that all nodes were in a single zone. Instead, the database is partitioned at points where a particular organization wants to take over control of

a subtree. Once an organization controls its own zone it can unilaterally change the data in the zone, grow new tree sections connected to the zone, delete existing nodes, or delegate new subzones under its zone.

If the organization has substructure, it may want to make further internal partitions to achieve nested delegations of name space control. In some cases, such divisions are made purely to make database maintenance more convenient.

4.2.1. Technical considerations

The data that describes a zone has four major parts:

- Authoritative data for all nodes within the zone.
- Data that defines the top node of the zone (can be thought of as part of the authoritative data).
- Data that describes delegated subzones, i.e., cuts around the bottom of the zone.
- Data that allows access to name servers for subzones (sometimes called "glue" data).

All of this data is expressed in the form of RRs, so a zone can be completely described in terms of a set of RRs. Whole zones can be transferred between name servers by transferring the RRs, either carried in a series of messages or by FTPing a master file which is a textual representation.

The authoritative data for a zone is simply all of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone.

Though logically part of the authoritative data, the RRs that describe the top node of the zone are especially important to the zone's management. These RRs are of two types: name server RRs that list, one per RR, all of the servers for the zone, and a single SOA RR that describes zone management parameters.

The RRs that describe cuts around the bottom of the zone are NS RRs that name the servers for the subzones. Since the cuts are between nodes, these RRs are NOT part of the authoritative data of the zone, and should be exactly the same as the corresponding RRs in the top node of the subzone. Since name servers are always associated with zone boundaries, NS RRs are only found at nodes which are the top node of some zone. In the data that makes up a zone, NS RRs are found at the top node of the

zone (and are authoritative) and at cuts around the bottom of the zone (where they are not authoritative), but never in between.

One of the goals of the zone structure is that any zone have all the data required to set up communications with the name servers for any subzones. That is, parent zones have all the information needed to access servers for their children zones. The NS RRs that name the servers for subzones are often not enough for this task since they name the servers, but do not give their addresses. In particular, if the name of the name server is itself in the subzone, we could be faced with the situation where the NS RRs tell us that in order to learn a name server's address, we should contact the server using the address we wish to learn. To fix this problem, a zone contains "glue" RRs which are not part of the authoritative data, and are address RRs for the servers. These RRs are only necessary if the name server's name is "below" the cut, and are only used as part of a referral response.

4.2.2. Administrative considerations

When some organization wants to control its own domain, the first step is to identify the proper parent zone, and get the parent zone's owners to agree to the delegation of control. While there are no particular technical constraints dealing with where in the tree this can be done, there are some administrative groupings discussed in [RFC-1032] which deal with top level organization, and middle level zones are free to create their own rules. For example, one university might choose to use a single zone, while another might choose to organize by subzones dedicated to individual departments or schools. [RFC-1033] catalogs available DNS software an discusses administration procedures.

Once the proper name for the new subzone is selected, the new owners should be required to demonstrate redundant name server support. Note that there is no requirement that the servers for a zone reside in a host which has a name in that domain. In many cases, a zone will be more accessible to the internet at large if its servers are widely distributed rather than being within the physical facilities controlled by the same organization that manages the zone. For example, in the current DNS, one of the name servers for the United Kingdom, or UK domain, is found in the US. This allows US hosts to get UK data without using limited transatlantic bandwidth.

As the last installation step, the delegation NS RRs and glue RRs necessary to make the delegation effective should be added to the parent zone. The administrators of both zones should insure that the NS and glue RRs which mark both sides of the cut are consistent and remain so.

4.3. Name server internals

4.3.1. Queries and responses

The principal activity of name servers is to answer standard queries. Both the query and its response are carried in a standard message format which is described in [RFC-1035]. The query contains a QTYPE, QCLASS, and QNAME, which describe the types and classes of desired information and the name of interest.

The way that the name server answers the query depends upon whether it is operating in recursive mode or not:

- The simplest mode for the server is non-recursive, since it can answer queries using only local information: the response contains an error, the answer, or a referral to some other server "closer" to the answer. All name servers must implement non-recursive queries.

- The simplest mode for the client is recursive, since in this mode the name server acts in the role of a resolver and returns either an error or the answer, but never referrals. This service is optional in a name server, and the name server may also choose to restrict the clients which can use recursive mode.

Recursive service is helpful in several situations:

- a relatively simple requester that lacks the ability to use anything other than a direct answer to the question.

- a request that needs to cross protocol or other boundaries and can be sent to a server which can act as intermediary.

- a network where we want to concentrate the cache rather than having a separate cache for each client.

Non-recursive service is appropriate if the requester is capable of pursuing referrals and interested in information which will aid future requests.

The use of recursive mode is limited to cases where both the client and the name server agree to its use. The agreement is negotiated through the use of two bits in query and response messages:

- The recursion available, or RA bit, is set or cleared by a name server in all responses. The bit is true if the name server is willing to provide recursive service for the client, regardless of whether the client requested recursive service. That is, RA signals availability rather than use.

- Queries contain a bit called recursion desired or RD. This bit specifies specifies whether the requester wants recursive service for this query. Clients may request recursive service from any name server, though they should depend upon receiving it only from servers which have previously sent an RA, or servers which have agreed to provide service through private agreement or some other means outside of the DNS protocol.

The recursive mode occurs when a query with RD set arrives at a server which is willing to provide recursive service; the client can verify that recursive mode was used by checking that both RA and RD are set in the reply. Note that the name server should never perform recursive service unless asked via RD, since this interferes with trouble shooting of name servers and their databases.

If recursive service is requested and available, the recursive response to a query will be one of the following:

- The answer to the query, possibly preface by one or more CNAME RRs that specify aliases encountered on the way to an answer.

- A name error indicating that the name does not exist. This may include CNAME RRs that indicate that the original query name was an alias for a name which does not exist.

- A temporary error indication.

If recursive service is not requested or is not available, the non-recursive response will be one of the following:

- An authoritative name error indicating that the name does not exist.

- A temporary error indication.

- Some combination of:

  RRs that answer the question, together with an indication whether the data comes from a zone or is cached.

  A referral to name servers which have zones which are closer ancestors to the name than the server sending the reply.

- RRs that the name server thinks will prove useful to the requester.

4.3.2. Algorithm

The actual algorithm used by the name server will depend on the local OS and data structures used to store RRs. The following algorithm assumes that the RRs are organized in several tree structures, one for each zone, and another for the cache:

1. Set or clear the value of recursion available in the response depending on whether the name server is willing to provide recursive service. If recursive service is available and requested via the RD bit in the query, go to step 5, otherwise step 2.

2. Search the available zones for the zone which is the nearest ancestor to QNAME. If such a zone is found, go to step 3, otherwise step 4.

3. Start matching down, label by label, in the zone. The matching process can terminate several ways:

   a. If the whole of QNAME is matched, we have found the node.

      If the data at the node is a CNAME, and QTYPE doesn't match CNAME, copy the CNAME RR into the answer section of the response, change QNAME to the canonical name in the CNAME RR, and go back to step 1.

      Otherwise, copy all RRs which match QTYPE into the answer section and go to step 6.

   b. If a match would take us out of the authoritative data, we have a referral. This happens when we encounter a node with NS RRs marking cuts along the bottom of a zone.

      Copy the NS RRs for the subzone into the authority section of the reply. Put whatever addresses are available into the additional section, using glue RRs if the addresses are not available from authoritative data or the cache. Go to step 4.

   c. If at some label, a match is impossible (i.e., the corresponding label does not exist), look to see if a the "*" label exists.

      If the "*" label does not exist, check whether the name we are looking for is the original QNAME in the query

or a name we have followed due to a CNAME. If the name is original, set an authoritative name error in the response and exit. Otherwise just exit.

If the "*" label does exist, match RRs at that node against QTYPE. If any match, copy them into the answer section, but set the owner of the RR to be QNAME, and not the node with the "*" label. Go to step 6.

4. Start matching down in the cache. If QNAME is found in the cache, copy all RRs attached to it that match QTYPE into the answer section. If there was no delegation from authoritative data, look for the best one from the cache, and put it in the authority section. Go to step 6.

5. Using the local resolver or a copy of its algorithm (see resolver section of this memo) to answer the query. Store the results, including any intermediate CNAMEs, in the answer section of the response.

6. Using local data only, attempt to add other RRs which may be useful to the additional section of the query. Exit.

4.3.3. Wildcards

In the previous algorithm, special treatment was given to RRs with owner names starting with the label "*". Such RRs are called wildcards. Wildcard RRs can be thought of as instructions for synthesizing RRs. When the appropriate conditions are met, the name server creates RRs with an owner name equal to the query name and contents taken from the wildcard RRs.

This facility is most often used to create a zone which will be used to forward mail from the Internet to some other mail system. The general idea is that any name in that zone which is presented to server in a query will be assumed to exist, with certain properties, unless explicit evidence exists to the contrary. Note that the use of the term zone here, instead of domain, is intentional; such defaults do not propagate across zone boundaries, although a subzone may choose to achieve that appearance by setting up similar defaults.

The contents of the wildcard RRs follows the usual rules and formats for RRs. The wildcards in the zone have an owner name that controls the query names they will match. The owner name of the wildcard RRs is of the form "*.<anydomain>", where <anydomain> is any domain name. <anydomain> should not contain other * labels, and should be in the authoritative data of the zone. The wildcards potentially apply to descendants of <anydomain>, but not to <anydomain> itself. Another way

to look at this is that the "*" label always matches at least one whole label and sometimes more, but always whole labels.

Wildcard RRs do not apply:

- When the query is in another zone. That is, delegation cancels the wildcard defaults.

- When the query name or a name between the wildcard domain and the query name is know to exist. For example, if a wildcard RR has an owner name of "*.X", and the zone also contains RRs attached to B.X, the wildcards would apply to queries for name Z.X (presuming there is no explicit information for Z.X), but not to B.X, A.B.X, or X.

A * label appearing in a query name has no special effect, but can be used to test for wildcards in an authoritative zone; such a query is the only way to get a response containing RRs with an owner name with * in it. The result of such a query should not be cached.

Note that the contents of the wildcard RRs are not modified when used to synthesize RRs.

To illustrate the use of wildcard RRs, suppose a large company with a large, non-IP/TCP, network wanted to create a mail gateway. If the company was called X.COM, and IP/TCP capable gateway machine was called A.X.COM, the following RRs might be entered into the COM zone:

```
X.COM           MX      10      A.X.COM

*.X.COM         MX      10      A.X.COM

A.X.COM         A       1.2.3.4
A.X.COM         MX      10      A.X.COM

*.A.X.COM       MX      10      A.X.COM
```

This would cause any MX query for any domain name ending in X.COM to return an MX RR pointing at A.X.COM. Two wildcard RRs are required since the effect of the wildcard at *.X.COM is inhibited in the A.X.COM subtree by the explicit data for A.X.COM. Note also that the explicit MX data at X.COM and A.X.COM is required, and that none of the RRs above would match a query name of XX.COM.

4.3.4. Negative response caching (Optional)

The DNS provides an optional service which allows name servers to distribute, and resolvers to cache, negative results with TTLs. For

example, a name server can distribute a TTL along with a name error indication, and a resolver receiving such information is allowed to assume that the name does not exist during the TTL period without consulting authoritative data. Similarly, a resolver can make a query with a QTYPE which matches multiple types, and cache the fact that some of the types are not present.

This feature can be particularly important in a system which implements naming shorthands that use search lists beacuse a popular shorthand, which happens to require a suffix toward the end of the search list, will generate multiple name errors whenever it is used.

The method is that a name server may add an SOA RR to the additional section of a response when that response is authoritative. The SOA must be that of the zone which was the source of the authoritative data in the answer section, or name error if applicable. The MINIMUM field of the SOA controls the length of time that the negative result may be cached.

Note that in some circumstances, the answer section may contain multiple owner names. In this case, the SOA mechanism should only be used for the data which matches QNAME, which is the only authoritative data in this section.

Name servers and resolvers should never attempt to add SOAs to the additional section of a non-authoritative response, or attempt to infer results which are not directly stated in an authoritative response. There are several reasons for this, including: cached information isn't usually enough to match up RRs and their zone names, SOA RRs may be cached due to direct SOA queries, and name servers are not required to output the SOAs in the authority section.

This feature is optional, although a refined version is expected to become part of the standard protocol in the future. Name servers are not required to add the SOA RRs in all authoritative responses, nor are resolvers required to cache negative results. Both are recommended. All resolvers and recursive name servers are required to at least be able to ignore the SOA RR when it is present in a response.

Some experiments have also been proposed which will use this feature. The idea is that if cached data is known to come from a particular zone, and if an authoritative copy of the zone's SOA is obtained, and if the zone's SERIAL has not changed since the data was cached, then the TTL of the cached data can be reset to the zone MINIMUM value if it is smaller. This usage is mentioned for planning purposes only, and is not recommended as yet.

4.3.5. Zone maintenance and transfers

Part of the job of a zone administrator is to maintain the zones at all of the name servers which are authoritative for the zone. When the inevitable changes are made, they must be distributed to all of the name servers. While this distribution can be accomplished using FTP or some other ad hoc procedure, the preferred method is the zone transfer part of the DNS protocol.

The general model of automatic zone transfer or refreshing is that one of the name servers is the master or primary for the zone. Changes are coordinated at the primary, typically by editing a master file for the zone. After editing, the administrator signals the master server to load the new zone. The other non-master or secondary servers for the zone periodically check for changes (at a selectable interval) and obtain new zone copies when changes have been made.

To detect changes, secondaries just check the SERIAL field of the SOA for the zone. In addition to whatever other changes are made, the SERIAL field in the SOA of the zone is always advanced whenever any change is made to the zone. The advancing can be a simple increment, or could be based on the write date and time of the master file, etc. The purpose is to make it possible to determine which of two copies of a zone is more recent by comparing serial numbers. Serial number advances and comparisons use sequence space arithmetic, so there is a theoretic limit on how fast a zone can be updated, basically that old copies must die out before the serial number covers half of its 32 bit range. In practice, the only concern is that the compare operation deals properly with comparisons around the boundary between the most positive and most negative 32 bit numbers.

The periodic polling of the secondary servers is controlled by parameters in the SOA RR for the zone, which set the minimum acceptable polling intervals. The parameters are called REFRESH, RETRY, and EXPIRE. Whenever a new zone is loaded in a secondary, the secondary waits REFRESH seconds before checking with the primary for a new serial. If this check cannot be completed, new checks are started every RETRY seconds. The check is a simple query to the primary for the SOA RR of the zone. If the serial field in the secondary's zone copy is equal to the serial returned by the primary, then no changes have occurred, and the REFRESH interval wait is restarted. If the secondary finds it impossible to perform a serial check for the EXPIRE interval, it must assume that its copy of the zone is obsolete an discard it.

When the poll shows that the zone has changed, then the secondary server must request a zone transfer via an AXFR request for the zone. The AXFR may cause an error, such as refused, but normally is answered by a sequence of response messages. The first and last messages must contain

the data for the top authoritative node of the zone. Intermediate messages carry all of the other RRs from the zone, including both authoritative and non-authoritative RRs. The stream of messages allows the secondary to construct a copy of the zone. Because accuracy is essential, TCP or some other reliable protocol must be used for AXFR requests.

Each secondary server is required to perform the following operations against the master, but may also optionally perform these operations against other secondary servers. This strategy can improve the transfer process when the primary is unavailable due to host downtime or network problems, or when a secondary server has better network access to an "intermediate" secondary than to the primary.

5. RESOLVERS

5.1. Introduction

Resolvers are programs that interface user programs to domain name servers. In the simplest case, a resolver receives a request from a user program (e.g., mail programs, TELNET, FTP) in the form of a subroutine call, system call etc., and returns the desired information in a form compatible with the local host's data formats.

The resolver is located on the same machine as the program that requests the resolver's services, but it may need to consult name servers on other hosts. Because a resolver may need to consult several name servers, or may have the requested information in a local cache, the amount of time that a resolver will take to complete can vary quite a bit, from milliseconds to several seconds.

A very important goal of the resolver is to eliminate network delay and name server load from most requests by answering them from its cache of prior results. It follows that caches which are shared by multiple processes, users, machines, etc., are more efficient than non-shared caches.

5.2. Client-resolver interface

5.2.1. Typical functions

The client interface to the resolver is influenced by the local host's conventions, but the typical resolver-client interface has three functions:

1. Host name to host address translation.

   This function is often defined to mimic a previous HOSTS.TXT

based function. Given a character string, the caller wants one or more 32 bit IP addresses. Under the DNS, it translates into a request for type A RRs. Since the DNS does not preserve the order of RRs, this function may choose to sort the returned addresses or select the "best" address if the service returns only one choice to the client. Note that a multiple address return is recommended, but a single address may be the only way to emulate prior HOSTS.TXT services.

2. Host address to host name translation

   This function will often follow the form of previous functions. Given a 32 bit IP address, the caller wants a character string. The octets of the IP address are reversed, used as name components, and suffixed with "IN-ADDR.ARPA". A type PTR query is used to get the RR with the primary name of the host. For example, a request for the host name corresponding to IP address 1.2.3.4 looks for PTR RRs for domain name "4.3.2.1.IN-ADDR.ARPA".

3. General lookup function

   This function retrieves arbitrary information from the DNS, and has no counterpart in previous systems. The caller supplies a QNAME, QTYPE, and QCLASS, and wants all of the matching RRs. This function will often use the DNS format for all RR data instead of the local host's, and returns all RR content (e.g., TTL) instead of a processed form with local quoting conventions.

When the resolver performs the indicated function, it usually has one of the following results to pass back to the client:

- One or more RRs giving the requested data.

  In this case the resolver returns the answer in the appropriate format.

- A name error (NE).

  This happens when the referenced name does not exist. For example, a user may have mistyped a host name.

- A data not found error.

  This happens when the referenced name exists, but data of the appropriate type does not. For example, a host address

function applied to a mailbox name would return this error since the name exists, but no address RR is present.

It is important to note that the functions for translating between host names and addresses may combine the "name error" and "data not found" error conditions into a single type of error return, but the general function should not. One reason for this is that applications may ask first for one type of information about a name followed by a second request to the same name for some other type of information; if the two errors are combined, then useless queries may slow the application.

5.2.2. Aliases

While attempting to resolve a particular request, the resolver may find that the name in question is an alias. For example, the resolver might find that the name given for host name to address translation is an alias when it finds the CNAME RR. If possible, the alias condition should be signalled back from the resolver to the client.

In most cases a resolver simply restarts the query at the new name when it encounters a CNAME. However, when performing the general function, the resolver should not pursue aliases when the CNAME RR matches the query type. This allows queries which ask whether an alias is present. For example, if the query type is CNAME, the user is interested in the CNAME RR itself, and not the RRs at the name it points to.

Several special conditions can occur with aliases. Multiple levels of aliases should be avoided due to their lack of efficiency, but should not be signalled as an error. Alias loops and aliases which point to non-existent names should be caught and an error condition passed back to the client.

5.2.3. Temporary failures

In a less than perfect world, all resolvers will occasionally be unable to resolve a particular request. This condition can be caused by a resolver which becomes separated from the rest of the network due to a link failure or gateway problem, or less often by coincident failure or unavailability of all servers for a particular domain.

It is essential that this sort of condition should not be signalled as a name or data not present error to applications. This sort of behavior is annoying to humans, and can wreak havoc when mail systems use the DNS.

While in some cases it is possible to deal with such a temporary problem by blocking the request indefinitely, this is usually not a good choice, particularly when the client is a server process that could move on to

other tasks. The recommended solution is to always have temporary failure as one of the possible results of a resolver function, even though this may make emulation of existing HOSTS.TXT functions more difficult.

5.3. Resolver internals

Every resolver implementation uses slightly different algorithms, and typically spends much more logic dealing with errors of various sorts than typical occurances. This section outlines a recommended basic strategy for resolver operation, but leaves details to [RFC-1035].

5.3.1. Stub resolvers

One option for implementing a resolver is to move the resolution function out of the local machine and into a name server which supports recursive queries. This can provide an easy method of providing domain service in a PC which lacks the resources to perform the resolver function, or can centralize the cache for a whole local network or organization.

All that the remaining stub needs is a list of name server addresses that will perform the recursive requests. This type of resolver presumably needs the information in a configuration file, since it probably lacks the sophistication to locate it in the domain database. The user also needs to verify that the listed servers will perform the recursive service; a name server is free to refuse to perform recursive services for any or all clients. The user should consult the local system administrator to find name servers willing to perform the service.

This type of service suffers from some drawbacks. Since the recursive requests may take an arbitrary amount of time to perform, the stub may have difficulty optimizing retransmission intervals to deal with both lost UDP packets and dead servers; the name server can be easily overloaded by too zealous a stub if it interprets retransmissions as new requests. Use of TCP may be an answer, but TCP may well place burdens on the host’s capabilities which are similar to those of a real resolver.

5.3.2. Resources

In addition to its own resources, the resolver may also have shared access to zones maintained by a local name server. This gives the resolver the advantage of more rapid access, but the resolver must be careful to never let cached information override zone data. In this discussion the term "local information" is meant to mean the union of the cache and such shared zones, with the understanding that

authoritative data is always used in preference to cached data when both are present.

The following resolver algorithm assumes that all functions have been converted to a general lookup function, and uses the following data structures to represent the state of a request in progress in the resolver:

SNAME — the domain name we are searching for.

STYPE — the QTYPE of the search request.

SCLASS — the QCLASS of the search request.

SLIST — a structure which describes the name servers and the zone which the resolver is currently trying to query. This structure keeps track of the resolver's current best guess about which name servers hold the desired information; it is updated when arriving information changes the guess. This structure includes the equivalent of a zone name, the known name servers for the zone, the known addresses for the name servers, and history information which can be used to suggest which server is likely to be the best one to try next. The zone name equivalent is a match count of the number of labels from the root down which SNAME has in common with the zone being queried; this is used as a measure of how "close" the resolver is to SNAME.

SBELT — a "safety belt" structure of the same form as SLIST, which is initialized from a configuration file, and lists servers which should be used when the resolver doesn't have any local information to guide name server selection. The match count will be -1 to indicate that no labels are known to match.

CACHE — A structure which stores the results from previous responses. Since resolvers are responsible for discarding old RRs whose TTL has expired, most implementations convert the interval specified in arriving RRs to some sort of absolute time when the RR is stored in the cache. Instead of counting the TTLs down individually, the resolver just ignores or discards old RRs when it runs across them in the course of a search, or discards them during periodic sweeps to reclaim the memory consumed by old RRs.

5.3.3. Algorithm

The top level algorithm has four steps:

1. See if the answer is in local information, and if so return it to the client.

2. Find the best servers to ask.

3. Send them queries until one returns a response.

4. Analyze the response, either:

    a. if the response answers the question or contains a name error, cache the data as well as returning it back to the client.

    b. if the response contains a better delegation to other servers, cache the delegation information, and go to step 2.

    c. if the response shows a CNAME and that is not the answer itself, cache the CNAME, change the SNAME to the canonical name in the CNAME RR and go to step 1.

    d. if the response shows a servers failure or other bizarre contents, delete the server from the SLIST and go back to step 3.

Step 1 searches the cache for the desired data. If the data is in the cache, it is assumed to be good enough for normal use. Some resolvers have an option at the user interface which will force the resolver to ignore the cached data and consult with an authoritative server. This is not recommended as the default. If the resolver has direct access to a name server's zones, it should check to see if the desired data is present in authoritative form, and if so, use the authoritative data in preference to cached data.

Step 2 looks for a name server to ask for the required data. The general strategy is to look for locally-available name server RRs, starting at SNAME, then the parent domain name of SNAME, the grandparent, and so on toward the root. Thus if SNAME were Mockapetris.ISI.EDU, this step would look for NS RRs for Mockapetris.ISI.EDU, then ISI.EDU, then EDU, and then . (the root). These NS RRs list the names of hosts for a zone at or above SNAME. Copy the names into SLIST. Set up their addresses using local data. It may be the case that the addresses are not available. The resolver has many choices here; the best is to start parallel resolver processes looking

for the addresses while continuing onward with the addresses which are available. Obviously, the design choices and options are complicated and a function of the local host's capabilities. The recommended priorities for the resolver designer are:

1. Bound the amount of work (packets sent, parallel processes started) so that a request can't get into an infinite loop or start off a chain reaction of requests or queries with other implementations EVEN IF SOMEONE HAS INCORRECTLY CONFIGURED SOME DATA.

2. Get back an answer if at all possible.

3. Avoid unnecessary transmissions.

4. Get the answer as quickly as possible.

If the search for NS RRs fails, then the resolver initializes SLIST from the safety belt SBELT. The basic idea is that when the resolver has no idea what servers to ask, it should use information from a configuration file that lists several servers which are expected to be helpful. Although there are special situations, the usual choice is two of the root servers and two of the servers for the host's domain. The reason for two of each is for redundancy. The root servers will provide eventual access to all of the domain space. The two local servers will allow the resolver to continue to resolve local names if the local network becomes isolated from the internet due to gateway or link failure.

In addition to the names and addresses of the servers, the SLIST data structure can be sorted to use the best servers first, and to insure that all addresses of all servers are used in a round-robin manner. The sorting can be a simple function of preferring addresses on the local network over others, or may involve statistics from past events, such as previous response times and batting averages.

Step 3 sends out queries until a response is received. The strategy is to cycle around all of the addresses for all of the servers with a timeout between each transmission. In practice it is important to use all addresses of a multihomed host, and too aggressive a retransmission policy actually slows response when used by multiple resolvers contending for the same name server and even occasionally for a single resolver. SLIST typically contains data values to control the timeouts and keep track of previous transmissions.

Step 4 involves analyzing responses. The resolver should be highly paranoid in its parsing of responses. It should also check that the response matches the query it sent using the ID field in the response.

The ideal answer is one from a server authoritative for the query which either gives the required data or a name error. The data is passed back to the user and entered in the cache for future use if its TTL is greater than zero.

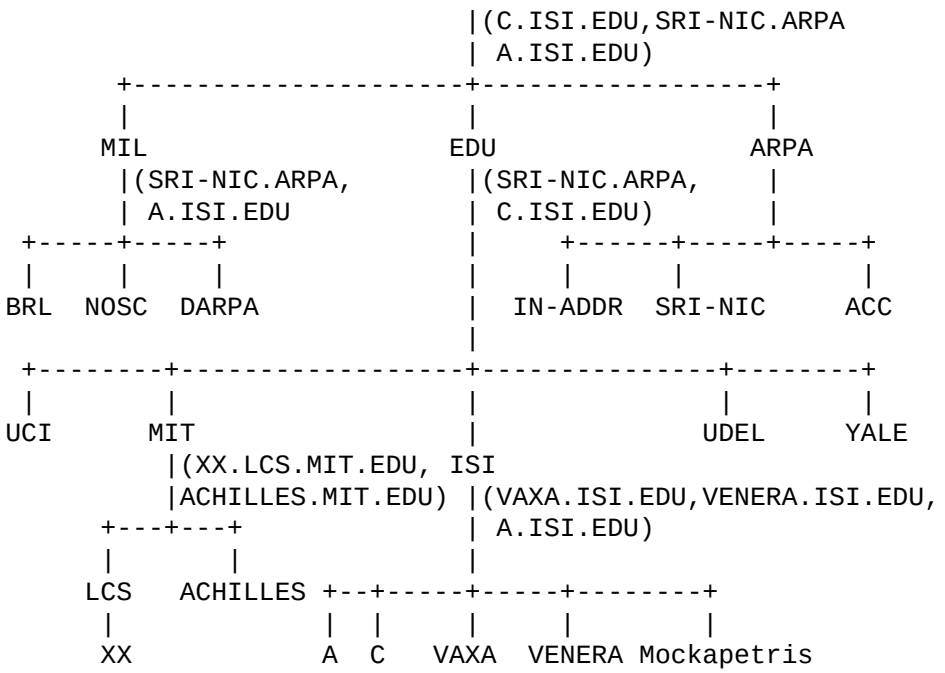
If the response shows a delegation, the resolver should check to see that the delegation is "closer" to the answer than the servers in SLIST are. This can be done by comparing the match count in SLIST with that computed from SNAME and the NS RRs in the delegation. If not, the reply is bogus and should be ignored. If the delegation is valid the NS delegation RRs and any address RRs for the servers should be cached. The name servers are entered in the SLIST, and the search is restarted.

If the response contains a CNAME, the search is restarted at the CNAME unless the response has the data for the canonical name or if the CNAME is the answer itself.

Details and implementation hints can be found in [RFC-1035].

6. A SCENARIO

In our sample domain space, suppose we wanted separate administrative control for the root, MIL, EDU, MIT.EDU and ISI.EDU zones. We might allocate name servers as follows:



In this example, the authoritative name server is shown in parentheses at the point in the domain tree at which is assumes control.

Thus the root name servers are on C.ISI.EDU, SRI-NIC.ARPA, and A.ISI.EDU. The MIL domain is served by SRI-NIC.ARPA and A.ISI.EDU. The EDU domain is served by SRI-NIC.ARPA. and C.ISI.EDU. Note that servers may have zones which are contiguous or disjoint. In this scenario, C.ISI.EDU has contiguous zones at the root and EDU domains. A.ISI.EDU has contiguous zones at the root and MIL domains, but also has a non-contiguous zone at ISI.EDU.

6.1. C.ISI.EDU name server

C.ISI.EDU is a name server for the root, MIL, and EDU domains of the IN class, and would have zones for these domains. The zone data for the root domain might be:

```
    .       IN      SOA     SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                            870611          ;serial
                            1800            ;refresh every 30 min
                            300             ;retry every 5 min
                            604800          ;expire after a week
                            86400)          ;minimum of a day
                    NS      A.ISI.EDU.
                    NS      C.ISI.EDU.
                    NS      SRI-NIC.ARPA.

    MIL.    86400   NS      SRI-NIC.ARPA.
            86400   NS      A.ISI.EDU.

    EDU.    86400   NS      SRI-NIC.ARPA.
            86400   NS      C.ISI.EDU.

    SRI-NIC.ARPA.   A       26.0.0.73
                    A       10.0.0.51
                    MX      0 SRI-NIC.ARPA.
                    HINFO   DEC-2060 TOPS20

    ACC.ARPA.       A       26.6.0.65
                    HINFO   PDP-11/70 UNIX
                    MX      10 ACC.ARPA.

    USC-ISIC.ARPA.  CNAME   C.ISI.EDU.

    73.0.0.26.IN-ADDR.ARPA.  PTR    SRI-NIC.ARPA.
    65.0.6.26.IN-ADDR.ARPA.  PTR    ACC.ARPA.
    51.0.0.10.IN-ADDR.ARPA.  PTR    SRI-NIC.ARPA.
    52.0.0.10.IN-ADDR.ARPA.  PTR    C.ISI.EDU.
```

```
    103.0.3.26.IN-ADDR.ARPA. PTR    A.ISI.EDU.

    A.ISI.EDU. 86400 A      26.3.0.103
    C.ISI.EDU. 86400 A      10.0.0.52
```

This data is represented as it would be in a master file. Most RRs are single line entries; the sole exception here is the SOA RR, which uses "(" to start a multi-line RR and ")" to show the end of a multi-line RR. Since the class of all RRs in a zone must be the same, only the first RR in a zone need specify the class. When a name server loads a zone, it forces the TTL of all authoritative RRs to be at least the MINIMUM field of the SOA, here 86400 seconds, or one day. The NS RRs marking delegation of the MIL and EDU domains, together with the glue RRs for the servers host addresses, are not part of the authoritative data in the zone, and hence have explicit TTLs.

Four RRs are attached to the root node: the SOA which describes the root zone and the 3 NS RRs which list the name servers for the root. The data in the SOA RR describes the management of the zone. The zone data is maintained on host SRI-NIC.ARPA, and the responsible party for the zone is HOSTMASTER@SRI-NIC.ARPA. A key item in the SOA is the 86400 second minimum TTL, which means that all authoritative data in the zone has at least that TTL, although higher values may be explicitly specified.

The NS RRs for the MIL and EDU domains mark the boundary between the root zone and the MIL and EDU zones. Note that in this example, the lower zones happen to be supported by name servers which also support the root zone.

The master file for the EDU zone might be stated relative to the origin EDU. The zone data for the EDU domain might be:

```
    EDU.  IN SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA. (
                            870729 ;serial
                            1800 ;refresh every 30 minutes
                            300 ;retry every 5 minutes
                            604800 ;expire after a week
                            86400 ;minimum of a day
                            )
                    NS SRI-NIC.ARPA.
                    NS C.ISI.EDU.

    UCI 172800 NS ICS.UCI
                    172800 NS ROME.UCI
    ICS.UCI 172800 A 192.5.19.1
    ROME.UCI 172800 A 192.5.19.31
```

```
    ISI 172800 NS VAXA.ISI
                   172800 NS A.ISI
                   172800 NS VENERA.ISI.EDU.
    VAXA.ISI 172800 A 10.2.0.27
                   172800 A 128.9.0.33
    VENERA.ISI.EDU. 172800 A 10.1.0.52
                   172800 A 128.9.0.32
    A.ISI 172800 A 26.3.0.103

    UDEL.EDU.  172800 NS LOUIE.UDEL.EDU.
                   172800 NS UMN-REI-UC.ARPA.
    LOUIE.UDEL.EDU. 172800 A 10.0.0.96
                   172800 A 192.5.39.3

    YALE.EDU.  172800 NS YALE.ARPA.
    YALE.EDU.  172800 NS YALE-BULLDOG.ARPA.

    MIT.EDU.  43200 NS XX.LCS.MIT.EDU.
                     43200 NS ACHILLES.MIT.EDU.
    XX.LCS.MIT.EDU.  43200 A 10.0.0.44
    ACHILLES.MIT.EDU. 43200 A 18.72.0.8
```

Note the use of relative names here. The owner name for the ISI.EDU. is stated using a relative name, as are two of the name server RR contents. Relative and absolute domain names may be freely intermixed in a master

6.2. Example standard queries

The following queries and responses illustrate name server behavior. Unless otherwise noted, the queries do not have recursion desired (RD) in the header. Note that the answers to non-recursive queries do depend on the server being asked, but do not depend on the identity of the requester.

6.2.1. QNAME=SRI-NIC.ARPA, QTYPE=A

The query would look like:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY                                   |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A         |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

The response from C.ISI.EDU would be:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA                     |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A         |
              +-------------------------------------------------+
    Answer    | SRI-NIC.ARPA. 86400 IN A 26.0.0.73              |
              |               86400 IN A 10.0.0.51              |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

The header of the response looks like the header of the query, except that the RESPONSE bit is set, indicating that this message is a response, not a query, and the Authoritative Answer (AA) bit is set indicating that the address RRs in the answer section are from authoritative data. The question section of the response matches the question section of the query.

If the same query was sent to some other server which was not authoritative for SRI-NIC.ARPA, the response might be:

```
            +-------------------------------------------------+
Header      | OPCODE=SQUERY,RESPONSE                          |
            +-------------------------------------------------+
Question    | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A         |
            +-------------------------------------------------+
Answer      | SRI-NIC.ARPA. 1777 IN A 10.0.0.51               |
            |               1777 IN A 26.0.0.73               |
            +-------------------------------------------------+
Authority   | <empty>                                         |
            +-------------------------------------------------+
Additional  | <empty>                                         |
            +-------------------------------------------------+
```

This response is different from the previous one in two ways: the header does not have AA set, and the TTLs are different. The inference is that the data did not come from a zone, but from a cache. The difference between the authoritative TTL and the TTL here is due to aging of the data in a cache. The difference in ordering of the RRs in the answer section is not significant.

6.2.2. QNAME=SRI-NIC.ARPA, QTYPE=*

A query similar to the previous one, but using a QTYPE of *, would receive the following response from C.ISI.EDU:

```
            +-------------------------------------------------+
Header      | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Question    | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
            +-------------------------------------------------+
Answer      | SRI-NIC.ARPA. 86400 IN  A     26.0.0.73         |
            |                         A     10.0.0.51         |
            |                         MX    0 SRI-NIC.ARPA.   |
            |                         HINFO DEC-2060 TOPS20   |
            +-------------------------------------------------+
Authority   | <empty>                                         |
            +-------------------------------------------------+
Additional  | <empty>                                         |
            +-------------------------------------------------+
```

If a similar query was directed to two name servers which are not authoritative for SRI-NIC.ARPA, the responses might be:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
              +-------------------------------------------------+
    Answer    | SRI-NIC.ARPA. 12345 IN     A       26.0.0.73    |
              |                            A       10.0.0.51    |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

and

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=*         |
              +-------------------------------------------------+
    Answer    | SRI-NIC.ARPA. 1290 IN HINFO  DEC-2060 TOPS20    |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

Neither of these answers have AA set, so neither response comes from authoritative data. The different contents and different TTLs suggest that the two servers cached data at different times, and that the first server cached the response to a QTYPE=A query and the second cached the response to a HINFO query.

6.2.3. QNAME=SRI-NIC.ARPA, QTYPE=MX

This type of query might be result from a mailer trying to look up routing information for the mail destination HOSTMASTER@SRI-NIC.ARPA. The response from C.ISI.EDU would be:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA                     |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=MX        |
              +-------------------------------------------------+
    Answer    | SRI-NIC.ARPA. 86400 IN     MX      0 SRI-NIC.ARPA.|
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| SRI-NIC.ARPA. 86400 IN     A       26.0.0.73     |
              |                            A       10.0.0.51     |
              +-------------------------------------------------+
```

This response contains the MX RR in the answer section of the response. The additional section contains the address RRs because the name server at C.ISI.EDU guesses that the requester will need the addresses in order to properly use the information carried by the MX.

6.2.4. QNAME=SRI-NIC.ARPA, QTYPE=NS

C.ISI.EDU would reply to this query with:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA                     |
              +-------------------------------------------------+
    Question  | QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=NS        |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

The only difference between the response and the query is the AA and RESPONSE bits in the header. The interpretation of this response is that the server is authoritative for the name, and the name exists, but no RRs of type NS are present there.

6.2.5. QNAME=SIR-NIC.ARPA, QTYPE=A

If a user mistyped a host name, we might see this type of query.

C.ISI.EDU would answer it with:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA, RCODE=NE           |
              +-------------------------------------------------+
    Question  | QNAME=SIR-NIC.ARPA., QCLASS=IN, QTYPE=A         |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | . SOA SRI-NIC.ARPA. HOSTMASTER.SRI-NIC.ARPA.    |
              |       870611 1800 300 604800 86400              |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

This response states that the name does not exist. This condition is signalled in the response code (RCODE) section of the header.

The SOA RR in the authority section is the optional negative caching information which allows the resolver using this response to assume that the name will not exist for the SOA MINIMUM (86400) seconds.

6.2.6. QNAME=BRL.MIL, QTYPE=A

If this query is sent to C.ISI.EDU, the reply would be:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
    Question  | QNAME=BRL.MIL, QCLASS=IN, QTYPE=A               |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | MIL.            86400 IN NS       SRI-NIC.ARPA. |
              |                 86400    NS       A.ISI.EDU.    |
              +-------------------------------------------------+
    Additional| A.ISI.EDU.              A         26.3.0.103    |
              | SRI-NIC.ARPA.           A         26.0.0.73     |
              |                         A         10.0.0.51     |
              +-------------------------------------------------+
```

This response has an empty answer section, but is not authoritative, so it is a referral. The name server on C.ISI.EDU, realizing that it is not authoritative for the MIL domain, has referred the requester to servers on A.ISI.EDU and SRI-NIC.ARPA, which it knows are authoritative for the MIL domain.

6.2.7. QNAME=USC-ISIC.ARPA, QTYPE=A

The response to this query from A.ISI.EDU would be:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA                     |
              +-------------------------------------------------+
    Question  | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A        |
              +-------------------------------------------------+
    Answer    | USC-ISIC.ARPA. 86400 IN CNAME     C.ISI.EDU.    |
              | C.ISI.EDU.     86400 IN A         10.0.0.52     |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

Note that the AA bit in the header guarantees that the data matching QNAME is authoritative, but does not say anything about whether the data for C.ISI.EDU is authoritative. This complete reply is possible because A.ISI.EDU happens to be authoritative for both the ARPA domain where USC-ISIC.ARPA is found and the ISI.EDU domain where C.ISI.EDU data is found.

If the same query was sent to C.ISI.EDU, its response might be the same as shown above if it had its own address in its cache, but might also be:

```
            +-------------------------------------------------+
Header      | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Question    | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A        |
            +-------------------------------------------------+
Answer      | USC-ISIC.ARPA.   86400 IN CNAME   C.ISI.EDU.    |
            +-------------------------------------------------+
Authority   | ISI.EDU.        172800 IN NS      VAXA.ISI.EDU. |
            |                        NS      A.ISI.EDU.      |
            |                        NS      VENERA.ISI.EDU. |
            +-------------------------------------------------+
Additional  | VAXA.ISI.EDU.   172800    A       10.2.0.27     |
            |                 172800    A       128.9.0.33    |
            | VENERA.ISI.EDU. 172800    A       10.1.0.52     |
            |                 172800    A       128.9.0.32    |
            | A.ISI.EDU.      172800    A       26.3.0.103    |
            +-------------------------------------------------+
```

This reply contains an authoritative reply for the alias USC-ISIC.ARPA, plus a referral to the name servers for ISI.EDU. This sort of reply isn't very likely given that the query is for the host name of the name server being asked, but would be common for other aliases.

6.2.8. QNAME=USC-ISIC.ARPA, QTYPE=CNAME

If this query is sent to either A.ISI.EDU or C.ISI.EDU, the reply would be:

```
            +-------------------------------------------------+
Header      | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Question    | QNAME=USC-ISIC.ARPA., QCLASS=IN, QTYPE=A        |
            +-------------------------------------------------+
Answer      | USC-ISIC.ARPA. 86400 IN CNAME      C.ISI.EDU.   |
            +-------------------------------------------------+
Authority   | <empty>                                         |
            +-------------------------------------------------+
Additional  | <empty>                                         |
            +-------------------------------------------------+
```

Because QTYPE=CNAME, the CNAME RR itself answers the query, and the name server doesn't attempt to look up anything for C.ISI.EDU. (Except possibly for the additional section.)

6.3. Example resolution

The following examples illustrate the operations a resolver must perform for its client. We assume that the resolver is starting without a

cache, as might be the case after system boot. We further assume that the system is not one of the hosts in the data and that the host is located somewhere on net 26, and that its safety belt (SBELT) data structure has the following information:

```
Match count = -1
SRI-NIC.ARPA.   26.0.0.73       10.0.0.51
A.ISI.EDU.      26.3.0.103
```

This information specifies servers to try, their addresses, and a match count of -1, which says that the servers aren't very close to the target. Note that the -1 isn't supposed to be an accurate closeness measure, just a value so that later stages of the algorithm will work.

The following examples illustrate the use of a cache, so each example assumes that previous requests have completed.

6.3.1. Resolve MX for ISI.EDU.

Suppose the first request to the resolver comes from the local mailer, which has mail for PVM@ISI.EDU. The mailer might then ask for type MX RRs for the domain name ISI.EDU.

The resolver would look in its cache for MX RRs at ISI.EDU, but the empty cache wouldn't be helpful. The resolver would recognize that it needed to query foreign servers and try to determine the best servers to query. This search would look for NS RRs for the domains ISI.EDU, EDU, and the root. These searches of the cache would also fail. As a last resort, the resolver would use the information from the SBELT, copying it into its SLIST structure.

At this point the resolver would need to pick one of the three available addresses to try. Given that the resolver is on net 26, it should choose either 26.0.0.73 or 26.3.0.103 as its first choice. It would then send off a query of the form:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY                                   |
              +-------------------------------------------------+
    Question  | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX             |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | <empty>                                         |
              +-------------------------------------------------+
    Additional| <empty>                                         |
              +-------------------------------------------------+
```

The resolver would then wait for a response to its query or a timeout. If the timeout occurs, it would try different servers, then different addresses of the same servers, lastly retrying addresses already tried. It might eventually receive a reply from SRI-NIC.ARPA:

```
              +-------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE                         |
              +-------------------------------------------------+
    Question  | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX             |
              +-------------------------------------------------+
    Answer    | <empty>                                         |
              +-------------------------------------------------+
    Authority | ISI.EDU.        172800 IN NS      VAXA.ISI.EDU. |
              |                           NS      A.ISI.EDU.    |
              |                           NS      VENERA.ISI.EDU.|
              +-------------------------------------------------+
    Additional| VAXA.ISI.EDU.   172800    A       10.2.0.27     |
              |                 172800    A       128.9.0.33    |
              | VENERA.ISI.EDU. 172800    A       10.1.0.52     |
              |                 172800    A       128.9.0.32    |
              | A.ISI.EDU.      172800    A       26.3.0.103    |
              +-------------------------------------------------+
```

The resolver would notice that the information in the response gave a closer delegation to ISI.EDU than its existing SLIST (since it matches three labels). The resolver would then cache the information in this response and use it to set up a new SLIST:

```
    Match count = 3
    A.ISI.EDU.      26.3.0.103
    VAXA.ISI.EDU.   10.2.0.27       128.9.0.33
    VENERA.ISI.EDU. 10.1.0.52       128.9.0.32
```

A.ISI.EDU appears on this list as well as the previous one, but that is purely coincidental. The resolver would again start transmitting and waiting for responses. Eventually it would get an answer:

```
            +-------------------------------------------------+
Header      | OPCODE=SQUERY, RESPONSE, AA                     |
            +-------------------------------------------------+
Question    | QNAME=ISI.EDU., QCLASS=IN, QTYPE=MX             |
            +-------------------------------------------------+
Answer      | ISI.EDU.               MX 10 VENERA.ISI.EDU.    |
            |                        MX 20 VAXA.ISI.EDU.      |
            +-------------------------------------------------+
Authority   | <empty>                                         |
            +-------------------------------------------------+
Additional  | VAXA.ISI.EDU.   172800  A  10.2.0.27            |
            |                 172800  A  128.9.0.33           |
            | VENERA.ISI.EDU. 172800  A  10.1.0.52            |
            |                 172800  A  128.9.0.32           |
            +-------------------------------------------------+
```

The resolver would add this information to its cache, and return the MX RRs to its client.

6.3.2. Get the host name for address 26.6.0.65

The resolver would translate this into a request for PTR RRs for 65.0.6.26.IN-ADDR.ARPA. This information is not in the cache, so the resolver would look for foreign servers to ask. No servers would match, so it would use SBELT again. (Note that the servers for the ISI.EDU domain are in the cache, but ISI.EDU is not an ancestor of 65.0.6.26.IN-ADDR.ARPA, so the SBELT is used.)

Since this request is within the authoritative data of both servers in SBELT, eventually one would return:

```
              +-----------------------------------------------------+
    Header    | OPCODE=SQUERY, RESPONSE, AA                         |
              +-----------------------------------------------------+
    Question  | QNAME=65.0.6.26.IN-ADDR.ARPA.,QCLASS=IN,QTYPE=PTR   |
              +-----------------------------------------------------+
    Answer    | 65.0.6.26.IN-ADDR.ARPA.    PTR     ACC.ARPA.        |
              +-----------------------------------------------------+
    Authority | <empty>                                             |
              +-----------------------------------------------------+
    Additional| <empty>                                             |
              +-----------------------------------------------------+
```

6.3.3. Get the host address of poneria.ISI.EDU

This request would translate into a type A request for poneria.ISI.EDU. The resolver would not find any cached data for this name, but would find the NS RRs in the cache for ISI.EDU when it looks for foreign servers to ask. Using this data, it would construct a SLIST of the form:

```
    Match count = 3

    A.ISI.EDU.      26.3.0.103
    VAXA.ISI.EDU.   10.2.0.27       128.9.0.33
    VENERA.ISI.EDU. 10.1.0.52
```

A.ISI.EDU is listed first on the assumption that the resolver orders its choices by preference, and A.ISI.EDU is on the same network.

One of these servers would answer the query.

7. REFERENCES and BIBLIOGRAPHY

[Dyer 87] Dyer, S., and F. Hsu, "Hesiod", Project Athena Technical Plan - Name Service, April 1987, version 1.9.

Describes the fundamentals of the Hesiod name service.

[IEN-116] J. Postel, "Internet Name Server", IEN-116, USC/Information Sciences Institute, August 1979.

A name service obsoleted by the Domain Name System, but still in use.

[Quarterman 86] Quarterman, J., and J. Hoskins, "Notable Computer Networks",Communications of the ACM, October 1986, volume 29, number 10.

[RFC-742] K. Harrenstien, "NAME/FINGER", RFC-742, Network Information Center, SRI International, December 1977.

[RFC-768] J. Postel, "User Datagram Protocol", RFC-768, USC/Information Sciences Institute, August 1980.

[RFC-793] J. Postel, "Transmission Control Protocol", RFC-793, USC/Information Sciences Institute, September 1981.

[RFC-799] D. Mills, "Internet Name Domains", RFC-799, COMSAT, September 1981.

Suggests introduction of a hierarchy in place of a flat name space for the Internet.

[RFC-805] J. Postel, "Computer Mail Meeting Notes", RFC-805, USC/Information Sciences Institute, February 1982.

[RFC-810] E. Feinler, K. Harrenstien, Z. Su, and V. White, "DOD Internet Host Table Specification", RFC-810, Network Information Center, SRI International, March 1982.

Obsolete. See RFC-952.

[RFC-811] K. Harrenstien, V. White, and E. Feinler, "Hostnames Server", RFC-811, Network Information Center, SRI International, March 1982.

Obsolete. See RFC-953.

[RFC-812] K. Harrenstien, and V. White, "NICNAME/WHOIS", RFC-812, Network Information Center, SRI International, March 1982.

[RFC-819] Z. Su, and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC-819, Network Information Center, SRI International, August 1982.

Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-821] J. Postel, "Simple Mail Transfer Protocol", RFC-821, USC/Information Sciences Institute, August 1980.

[RFC-830] Z. Su, "A Distributed System for Internet Name Service", RFC-830, Network Information Center, SRI International, October 1982.

Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC-882, USC/Information Sciences Institute, November 1983.

Superceeded by this memo.

[RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC-883, USC/Information Sciences Institute, November 1983.

Superceeded by this memo.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC-920, USC/Information Sciences Institute October 1984.

Explains the naming scheme for top level domains.

[RFC-952] K. Harrenstien, M. Stahl, E. Feinler, "DoD Internet Host Table Specification", RFC-952, SRI, October 1985.

Specifies the format of HOSTS.TXT, the host/address table replaced by the DNS.

[RFC-953] K. Harrenstien, M. Stahl, E. Feinler, "HOSTNAME Server", RFC-953, SRI, October 1985.

This RFC contains the official specification of the hostname server protocol, which is obsoleted by the DNS. This TCP based protocol accesses information stored in the RFC-952 format, and is used to obtain copies of the host table.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC-973, USC/Information Sciences Institute, January 1986.

Describes changes to RFC-882 and RFC-883 and reasons for them. Now obsolete.

[RFC-974] C. Partridge, "Mail routing and the domain system", RFC-974, CSNET CIC BBN Labs, January 1986.

Describes the transition from HOSTS.TXT based mail addressing to the more powerful MX system used with the domain system.

[RFC-1001] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and Methods", RFC-1001, March 1987.

This RFC and RFC-1002 are a preliminary design for NETBIOS on top of TCP/IP which proposes to base NetBIOS name service on top of the DNS.

[RFC-1002] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed Specifications", RFC-1002, March 1987.

[RFC-1010] J. Reynolds and J. Postel, "Assigned Numbers", RFC-1010, USC/Information Sciences Institute, May 1987

Contains socket numbers and mnemonics for host names, operating systems, etc.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition", RFC-1031, November 1987.

Describes a plan for converting the MILNET to the DNS.

[RFC-1032] M. K. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC-1032, November 1987.

Describes the registration policies used by the NIC to administer the top level domains and delegate subzones.

[RFC-1033] M. K. Lottor, "Domain Administrators Operations Guide", RFC-1033, November 1987.

A cookbook for domain administrators.

[Solomon 82] M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server", Computer Networks, vol 6, nr 3, July 1982.

Describes a name service for CSNET which is independent from the DNS and DNS use in the CSNET.

Index

Network Working Group
Request for Comments: 1035

P. Mockapetris
ISI

November 1987

Obsoletes: RFCs 882, 883, 973

DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION

1. STATUS OF THIS MEMO

This RFC describes the details of the domain system and protocol, and assumes that the reader is familiar with the concepts discussed in a companion RFC, "Domain Names - Concepts and Facilities" [RFC-1034].

The domain system is a mixture of functions and data types which are an official protocol and functions and data types which are still experimental. Since the domain system is intentionally extensible, new data types and experimental behavior should always be expected in parts of the system beyond the official protocol. The official protocol parts include standard queries, responses and the Internet class RR data formats (e.g., host addresses). Since the previous RFC set, several definitions have changed, so some previous definitions are obsolete.

Experimental or obsolete features are clearly marked in these RFCs, and such information should be used with caution.

The reader is especially cautioned not to depend on the values which appear in examples to be current or complete, since their purpose is primarily pedagogical. Distribution of this memo is unlimited.

Table of Contents

2. INTRODUCTION

2.1. Overview

The goal of domain names is to provide a mechanism for naming resources in such a way that the names are usable in different hosts, networks, protocol families, internets, and administrative organizations.

From the user's point of view, domain names are useful as arguments to a local agent, called a resolver, which retrieves information associated with the domain name. Thus a user might ask for the host address or mail information associated with a particular domain name. To enable the user to request a particular type of information, an appropriate query type is passed to the resolver with the domain name. To the user, the domain tree is a single information space; the resolver is responsible for hiding the distribution of data among name servers from the user.

From the resolver's point of view, the database that makes up the domain space is distributed among various name servers. Different parts of the domain space are stored in different name servers, although a particular data item will be stored redundantly in two or more name servers. The resolver starts with knowledge of at least one name server. When the resolver processes a user query it asks a known name server for the information; in return, the resolver either receives the desired information or a referral to another name server. Using these referrals, resolvers learn the identities and contents of other name servers. Resolvers are responsible for dealing with the distribution of the domain space and dealing with the effects of name server failure by consulting redundant databases in other servers.

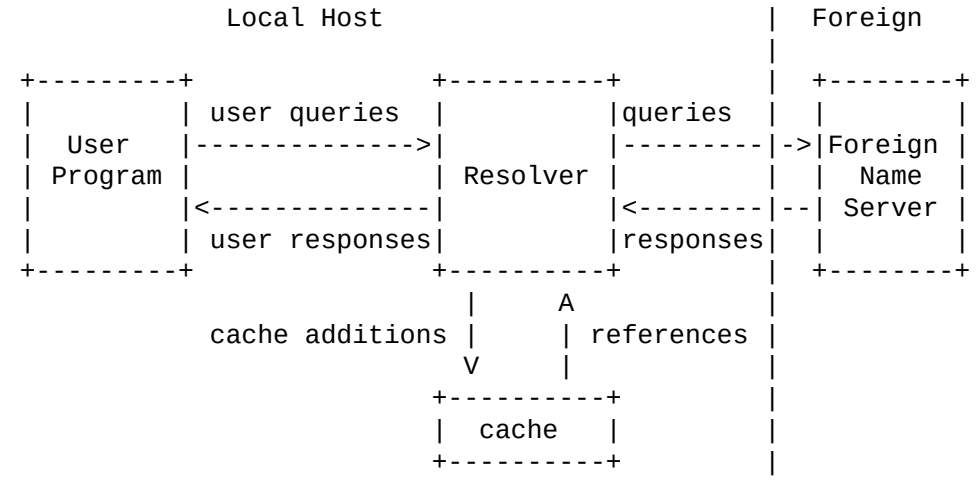
Name servers manage two kinds of data. The first kind of data held in sets called zones; each zone is the complete database for a particular "pruned" subtree of the domain space. This data is called authoritative. A name server periodically checks to make sure that its zones are up to date, and if not, obtains a new copy of updated zones

from master files stored locally or in another name server. The second kind of data is cached data which was acquired by a local resolver. This data may be incomplete, but improves the performance of the retrieval process when non-local data is repeatedly accessed. Cached data is eventually discarded by a timeout mechanism.

This functional structure isolates the problems of user interface, failure recovery, and distribution in the resolvers and isolates the database update and refresh problems in the name servers.

2.2. Common configurations

A host can participate in the domain name system in a number of ways, depending on whether the host runs programs that retrieve information from the domain system, name servers that answer queries from other hosts, or various combinations of both functions. The simplest, and perhaps most typical, configuration is shown below:

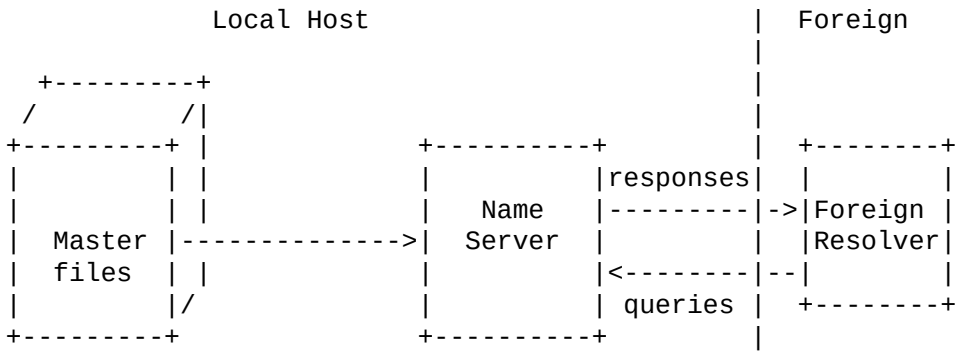


User programs interact with the domain name space through resolvers; the format of user queries and user responses is specific to the host and its operating system. User queries will typically be operating system calls, and the resolver and its cache will be part of the host operating system. Less capable hosts may choose to implement the resolver as a subroutine to be linked in with every program that needs its services. Resolvers answer user queries with information they acquire via queries to foreign name servers and the local cache.

Note that the resolver may have to make several queries to several different foreign name servers to answer a particular user query, and hence the resolution of a user query may involve several network accesses and an arbitrary amount of time. The queries to foreign name servers and the corresponding responses have a standard format described

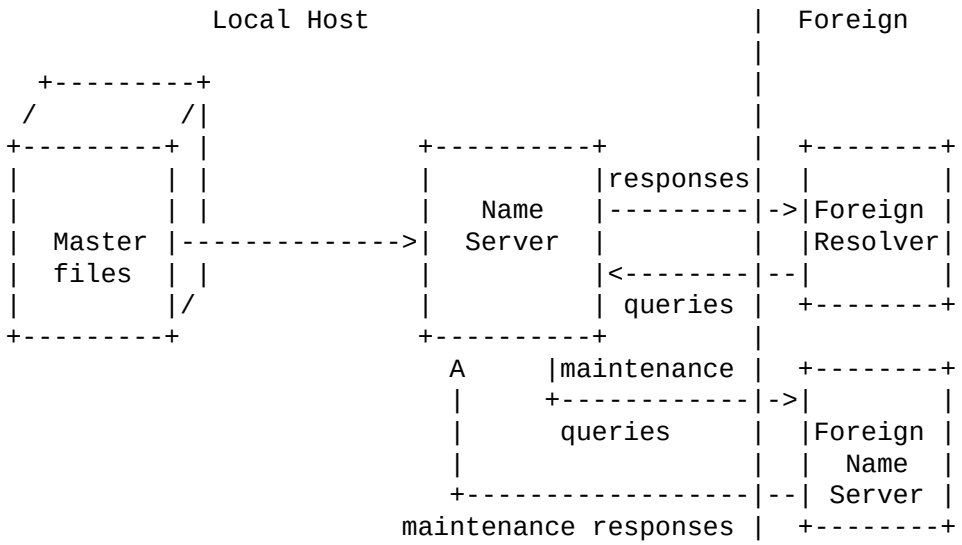
in this memo, and may be datagrams.

Depending on its capabilities, a name server could be a stand alone program on a dedicated machine or a process or processes on a large timeshared host. A simple configuration might be:



Here a primary name server acquires information about one or more zones by reading master files from its local file system, and answers queries about those zones that arrive from foreign resolvers.

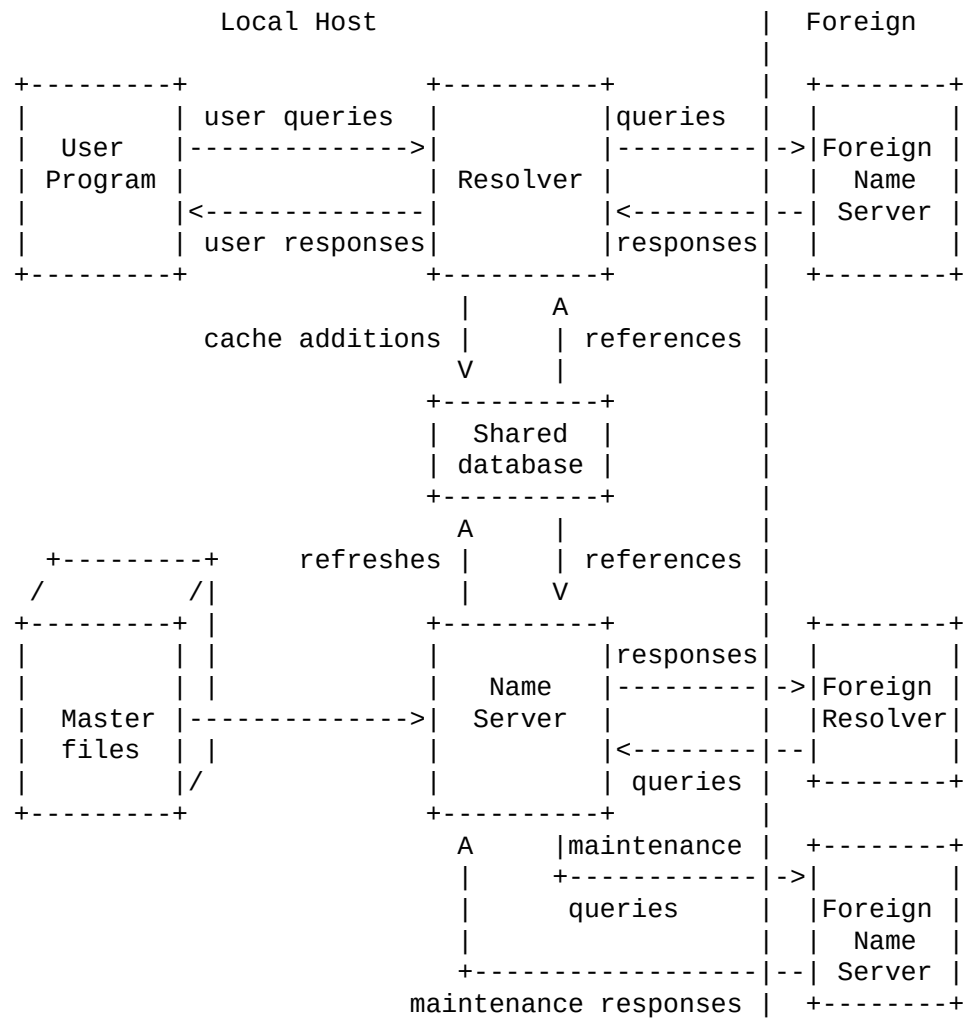
The DNS requires that all zones be redundantly supported by more than one name server. Designated secondary servers can acquire zones and check for updates from the primary server using the zone transfer protocol of the DNS. This configuration is shown below:



In this configuration, the name server periodically establishes a virtual circuit to a foreign name server to acquire a copy of a zone or to check that an existing copy has not changed. The messages sent for

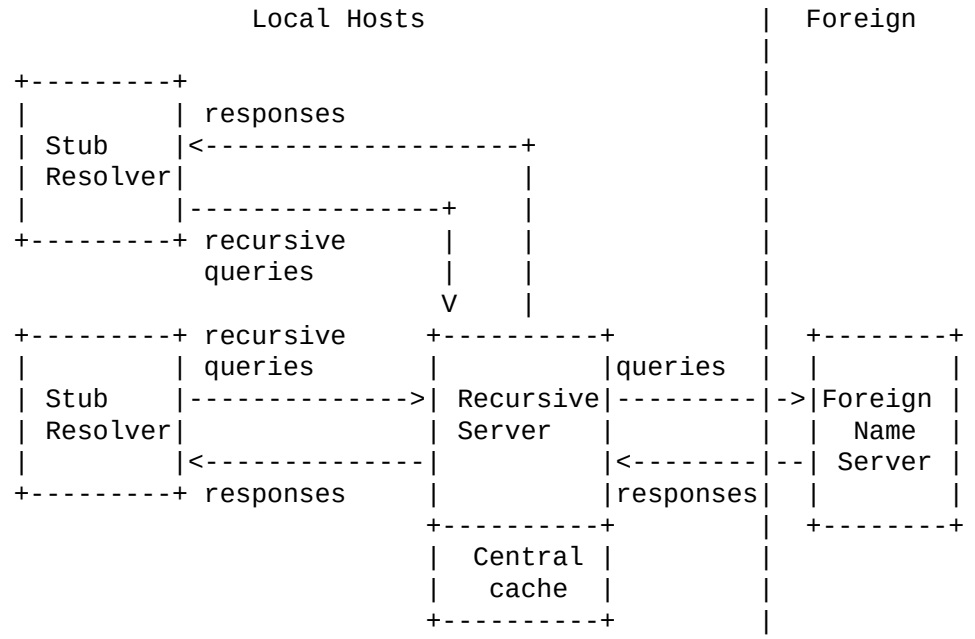
these maintenance activities follow the same form as queries and responses, but the message sequences are somewhat different.

The information flow in a host that supports all aspects of the domain name system is shown below:



The shared database holds domain space data for the local name server and resolver. The contents of the shared database will typically be a mixture of authoritative data maintained by the periodic refresh operations of the name server and cached data from previous resolver requests. The structure of the domain data and the necessity for synchronization between name servers and resolvers imply the general characteristics of this database, but the actual format is up to the local implementor.

Information flow can also be tailored so that a group of hosts act together to optimize activities. Sometimes this is done to offload less capable hosts so that they do not have to implement a full resolver. This can be appropriate for PCs or hosts which want to minimize the amount of new network code which is required. This scheme can also allow a group of hosts can share a small number of caches rather than maintaining a large number of separate caches, on the premise that the centralized caches will have a higher hit ratio. In either case, resolvers are replaced with stub resolvers which act as front ends to resolvers located in a recursive server in one or more name servers known to perform that service:



In any case, note that domain components are always replicated for reliability whenever possible.

2.3. Conventions

The domain system has several conventions dealing with low-level, but fundamental, issues. While the implementor is free to violate these conventions WITHIN HIS OWN SYSTEM, he must observe these conventions in ALL behavior observed from other hosts.

2.3.1. Preferred name syntax

The DNS specifications attempt to be as general as possible in the rules for constructing domain names. The idea is that the name of any existing object can be expressed as a domain name with minimal changes.

However, when assigning a domain name for an object, the prudent user will select a name which satisfies both the rules of the domain system and any existing rules for the object, whether these rules are published or implied by existing programs.

For example, when naming a mail domain, the user should satisfy both the rules of this memo and those in RFC-822. When creating a new host name, the old rules for HOSTS.TXT should be followed. This avoids problems when old software is converted to use domain names.

The following syntax will result in fewer problems with many

applications that use domain names (e.g., mail, TELNET).

<domain> ::= <subdomain> | " "

<subdomain> ::= <label> | <subdomain> "." <label>

<label> ::= <letter> [[<ldh-str>] <let-dig>]

<ldh-str> ::= <let-dig-hyp> | <let-dig-hyp> <ldh-str>

<let-dig-hyp> ::= <let-dig> | "-"

<let-dig> ::= <letter> | <digit>

<letter> ::= any one of the 52 alphabetic characters A through Z in upper case and a through z in lower case

<digit> ::= any one of the ten digits 0 through 9

Note that while upper and lower case letters are allowed in domain names, no significance is attached to the case. That is, two names with the same spelling but different case are to be treated as if identical.

The labels must follow the rules for ARPANET host names. They must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. There are also some restrictions on the length. Labels must be 63 characters or less.

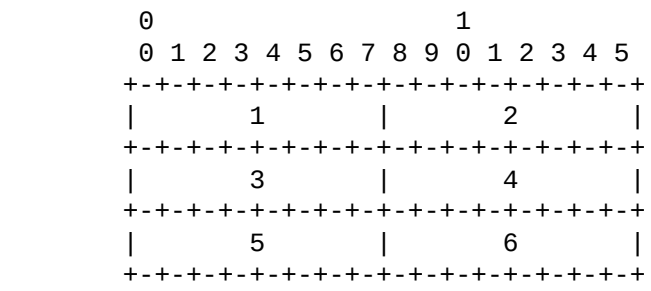
For example, the following strings identify hosts in the Internet:

A.ISI.EDU XX.LCS.MIT.EDU SRI-NIC.ARPA

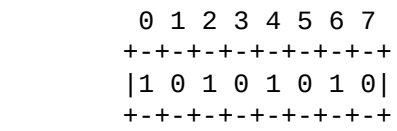
2.3.2. Data Transmission Order

The order of transmission of the header and data described in this document is resolved to the octet level. Whenever a diagram shows a

group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram, the octets are transmitted in the order they are numbered.



Whenever an octet represents a numeric quantity, the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).



Similarly, whenever a multi-octet field represents a numeric quantity the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

2.3.3. Character Case

For all parts of the DNS that are part of the official protocol, all comparisons between character strings (e.g., labels, domain names, etc.) are done in a case-insensitive manner. At present, this rule is in force throughout the domain system without exception. However, future additions beyond current usage may need to use the full binary octet capabilities in names, so attempts to store domain names in 7-bit ASCII or use of special bytes to terminate labels, etc., should be avoided.

When data enters the domain system, its original case should be preserved whenever possible. In certain circumstances this cannot be done. For example, if two RRs are stored in a database, one at x.y and one at X.Y, they are actually stored at the same place in the database, and hence only one casing would be preserved. The basic rule is that case can be discarded only when data is used to define structure in a database, and two names are identical when compared in a case insensitive manner.

Loss of case sensitive data must be minimized. Thus while data for x.y and X.Y may both be stored under a single location x.y or X.Y, data for a.x and B.X would never be stored under A.x, A.X, b.x, or b.X. In general, this preserves the case of the first label of a domain name, but forces standardization of interior node labels.

Systems administrators who enter data into the domain database should take care to represent the data they supply to the domain system in a case-consistent manner if their system is case-sensitive. The data distribution system in the domain system will ensure that consistent representations are preserved.

2.3.4. Size limits

Various objects and parameters in the DNS have size limits. They are listed below. Some could be easily changed, others are more fundamental.

labels	63 octets or less
names	255 octets or less
TTL	positive values of a signed 32 bit number.
UDP messages	512 octets or less

3. DOMAIN NAME SPACE AND RR DEFINITIONS

3.1. Name space definitions

Domain names in messages are expressed in terms of a sequence of labels. Each label is represented as a one octet length field followed by that number of octets. Since every domain name ends with the null label of the root, a domain name is terminated by a length byte of zero. The high order two bits of every length octet must be zero, and the remaining six bits of the length field limit the label to 63 octets or less.

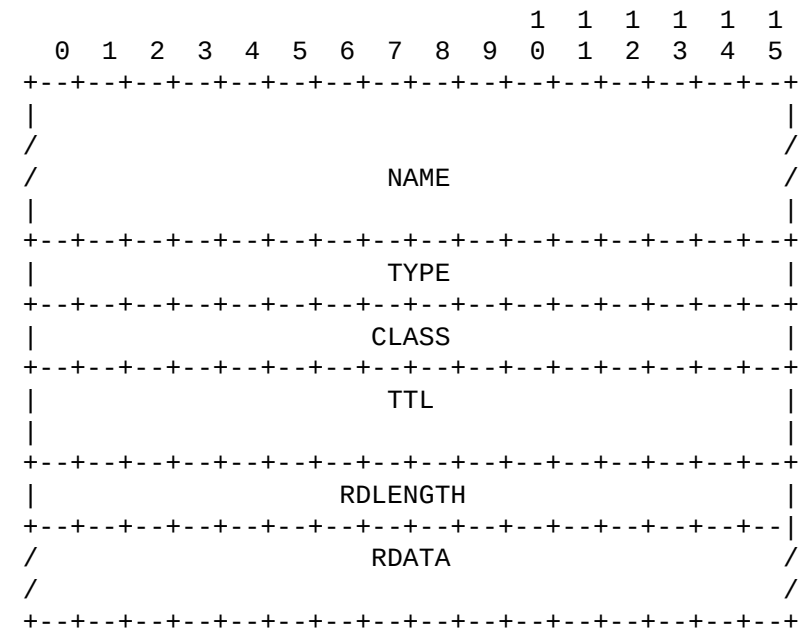
To simplify implementations, the total length of a domain name (i.e., label octets and label length octets) is restricted to 255 octets or less.

Although labels can contain any 8 bit values in octets that make up a label, it is strongly recommended that labels follow the preferred syntax described elsewhere in this memo, which is compatible with existing host naming conventions. Name servers and resolvers must compare labels in a case-insensitive manner (i.e., A=a), assuming ASCII with zero parity. Non-alphabetic codes must match exactly.

3.2. RR definitions

3.2.1. Format

All RRs have the same top level format shown below:



where:

NAME — an owner name, i.e., the name of the node to which this resource record pertains.

TYPE — two octets containing one of the RR TYPE codes.

CLASS — two octets containing one of the RR CLASS codes.

TTL — a 32 bit signed integer that specifies the time interval that the resource record may be cached before the source of the information should again be consulted. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached. For example, SOA records are always distributed with a zero TTL to prohibit caching. Zero values can also be used for extremely volatile data.

RDLENGTH — an unsigned 16 bit integer that specifies the length in octets of the RDATA field.

RDATA a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record.

3.2.2. TYPE values

TYPE fields are used in resource records. Note that these types are a subset of QTYPEs.

TYPE	value and meaning
A	1 a host address
NS	2 an authoritative name server
MD	3 a mail destination (Obsolete - use MX)
MF	4 a mail forwarder (Obsolete - use MX)
CNAME	5 the canonical name for an alias
SOA	6 marks the start of a zone of authority
MB	7 a mailbox domain name (EXPERIMENTAL)
MG	8 a mail group member (EXPERIMENTAL)
MR	9 a mail rename domain name (EXPERIMENTAL)
NULL	10 a null RR (EXPERIMENTAL)
WKS	11 a well known service description
PTR	12 a domain name pointer
HINFO	13 host information
MINFO	14 mailbox or mail list information
MX	15 mail exchange
TXT	16 text strings

3.2.3. QTYPE values

QTYPE fields appear in the question part of a query. QTYPES are a superset of TYPEs, hence all TYPEs are valid QTYPEs. In addition, the following QTYPEs are defined:

AXFR	252 A request for a transfer of an entire zone
MAILB	253 A request for mailbox-related records (MB, MG or MR)
MAILA	254 A request for mail agent RRs (Obsolete - see MX)
*	255 A request for all records

3.2.4. CLASS values

CLASS fields appear in resource records. The following CLASS mnemonics and values are defined:

IN	1 the Internet
CS	2 the CSNET class (Obsolete - used only for examples in some obsolete RFCs)
CH	3 the CHAOS class
HS	4 Hesiod [Dyer 87]

3.2.5. QCLASS values

QCLASS fields appear in the question section of a query. QCLASS values are a superset of CLASS values; every CLASS is a valid QCLASS. In addition to CLASS values, the following QCLASSes are defined:

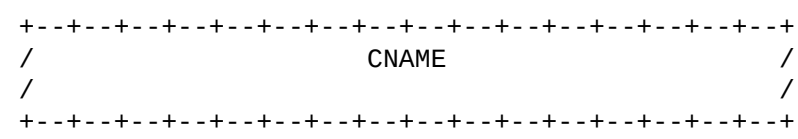
*	255 any class

3.3. Standard RRs

The following RR definitions are expected to occur, at least potentially, in all classes. In particular, NS, SOA, CNAME, and PTR will be used in all classes, and have the same format in all classes. Because their RDATA format is known, all domain names in the RDATA section of these RRs may be compressed.

<domain-name> is a domain name represented as a series of labels, and terminated by a label with zero length. <character-string> is a single length octet followed by that number of characters. <character-string> is treated as binary information, and can be up to 256 characters in length (including the length octet).

3.3.1. CNAME RDATA format

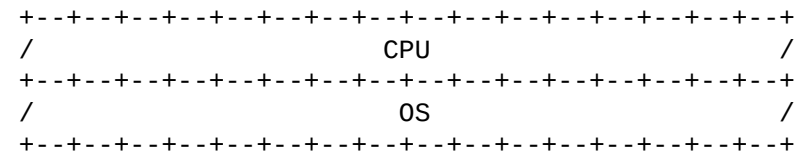


where:

CNAME A <domain-name> which specifies the canonical or primary name for the owner. The owner name is an alias.

CNAME RRs cause no additional section processing, but name servers may choose to restart the query at the canonical name in certain cases. See the description of name server logic in [RFC-1034] for details.

3.3.2. HINFO RDATA format



where:

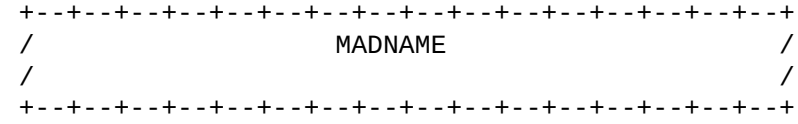
CPU A <character-string> which specifies the CPU type.

OS A <character-string> which specifies the operating system type.

Standard values for CPU and OS can be found in [RFC-1010].

HINFO records are used to acquire general information about a host. The main use is for protocols such as FTP that can use special procedures when talking between machines or operating systems of the same type.

3.3.3. MB RDATA format (EXPERIMENTAL)

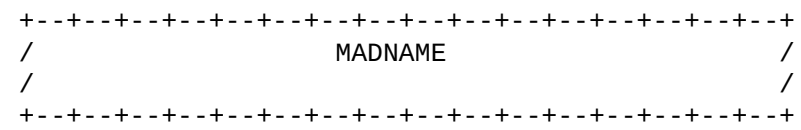


where:

MADNAME A <domain-name> which specifies a host which has the specified mailbox.

MB records cause additional section processing which looks up an A type RRs corresponding to MADNAME.

3.3.4. MD RDATA format (Obsolete)



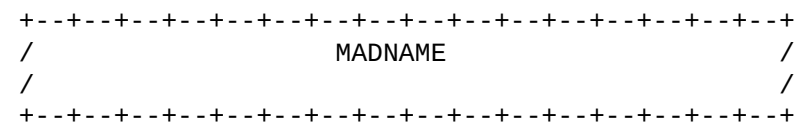
where:

MADNAME A <domain-name> which specifies a host which has a mail agent for the domain which should be able to deliver mail for the domain.

MD records cause additional section processing which looks up an A type record corresponding to MADNAME.

MD is obsolete. See the definition of MX and [RFC-974] for details of the new scheme. The recommended policy for dealing with MD RRs found in a master file is to reject them, or to convert them to MX RRs with a preference of 0.

3.3.5. MF RDATA format (Obsolete)



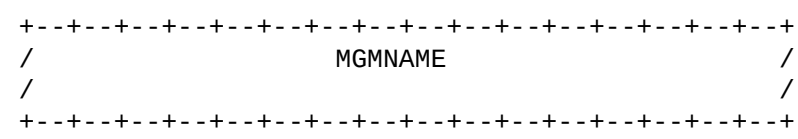
where:

MADNAME A <domain-name> which specifies a host which has a mail agent for the domain which will accept mail for forwarding to the domain.

MF records cause additional section processing which looks up an A type record corresponding to MADNAME.

MF is obsolete. See the definition of MX and [RFC-974] for details ofw the new scheme. The recommended policy for dealing with MD RRs found in a master file is to reject them, or to convert them to MX RRs with a preference of 10.

3.3.6. MG RDATA format (EXPERIMENTAL)

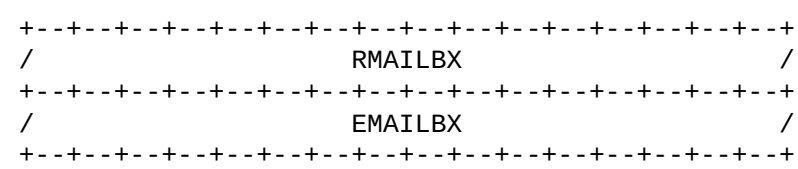


where:

MGMNAME A <domain-name> which specifies a mailbox which is a member of the mail group specified by the domain name.

MG records cause no additional section processing.

3.3.7. MINFO RDATA format (EXPERIMENTAL)



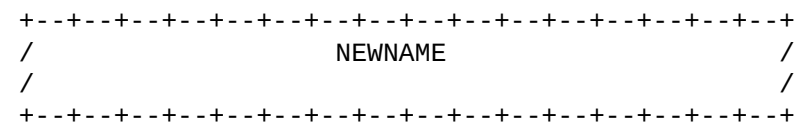
where:

RMAILBX A <domain-name> which specifies a mailbox which is responsible for the mailing list or mailbox. If this domain name names the root, the owner of the MINFO RR is responsible for itself. Note that many existing mailing lists use a mailbox X-request for the RMAILBX field of mailing list X, e.g., Msgroup-request for Msgroup. This field provides a more general mechanism.

EMAILBX A <domain-name> which specifies a mailbox which is to receive error messages related to the mailing list or mailbox specified by the owner of the MINFO RR (similar to the ERRORS-TO: field which has been proposed). If this domain name names the root, errors should be returned to the sender of the message.

MINFO records cause no additional section processing. Although these records can be associated with a simple mailbox, they are usually used with a mailing list.

3.3.8. MR RDATA format (EXPERIMENTAL)

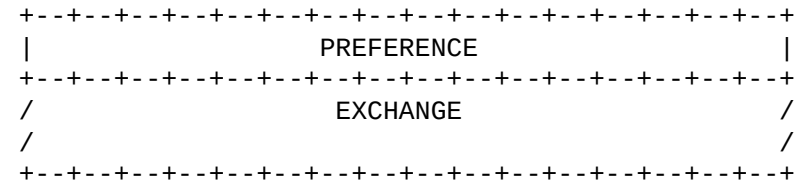


where:

NEWNAME A <domain-name> which specifies a mailbox which is the proper rename of the specified mailbox.

MR records cause no additional section processing. The main use for MR is as a forwarding entry for a user who has moved to a different mailbox.

3.3.9. MX RDATA format



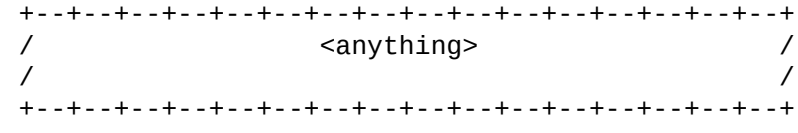
where:

PREFERENCE A 16 bit integer which specifies the preference given to this RR among others at the same owner. Lower values are preferred.

EXCHANGE A <domain-name> which specifies a host willing to act as a mail exchange for the owner name.

MX records cause type A additional section processing for the host specified by EXCHANGE. The use of MX RRs is explained in detail in [RFC-974].

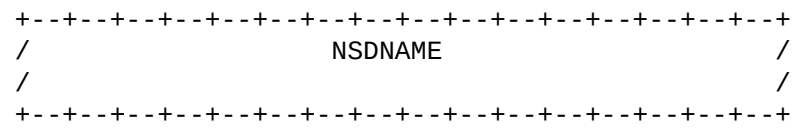
3.3.10. NULL RDATA format (EXPERIMENTAL)



Anything at all may be in the RDATA field so long as it is 65535 octets or less.

NULL records cause no additional section processing. NULL RRs are not allowed in master files. NULLs are used as placeholders in some experimental extensions of the DNS.

3.3.11. NS RDATA format



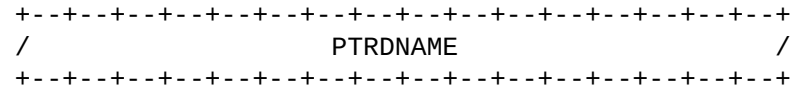
where:

NSDNAME A <domain-name> which specifies a host which should be authoritative for the specified class and domain.

NS records cause both the usual additional section processing to locate a type A record, and, when used in a referral, a special search of the zone in which they reside for glue information.

The NS RR states that the named host should be expected to have a zone starting at owner name of the specified class. Note that the class may not indicate the protocol family which should be used to communicate with the host, although it is typically a strong hint. For example, hosts which are name servers for either Internet (IN) or Hesiod (HS) class information are normally queried using IN class protocols.

3.3.12. PTR RDATA format

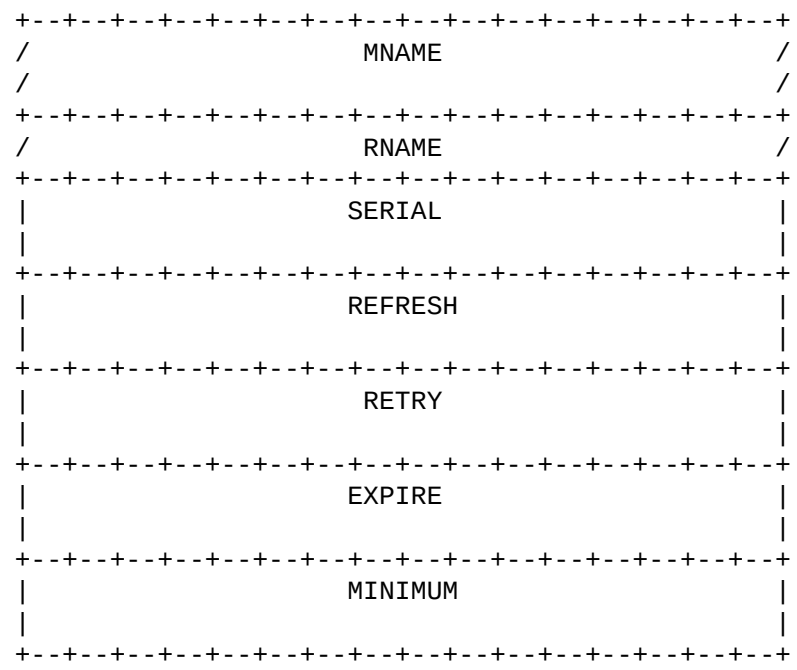


where:

PTRDNAME A <domain-name> which points to some location in the domain name space.

PTR records cause no additional section processing. These RRs are used in special domains to point to some other location in the domain space. These records are simple data, and don’t imply any special processing similar to that performed by CNAME, which identifies aliases. See the description of the IN-ADDR.ARPA domain for an example.

3.3.13. SOA RDATA format



where:

MNAME The <domain-name> of the name server that was the original or primary source of data for this zone.

RNAME A <domain-name> which specifies the mailbox of the person responsible for this zone.

SERIAL The unsigned 32 bit version number of the original copy of the zone. Zone transfers preserve this value. This value wraps and should be compared using sequence space arithmetic.

REFRESH A 32 bit time interval before the zone should be refreshed.

RETRY A 32 bit time interval that should elapse before a failed refresh should be retried.

EXPIRE A 32 bit time value that specifies the upper limit on the time interval that can elapse before the zone is no longer authoritative.

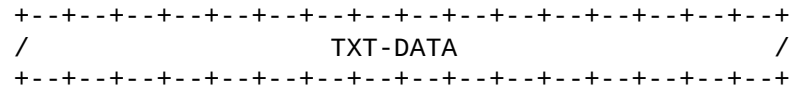
MINIMUM The unsigned 32 bit minimum TTL field that should be exported with any RR from this zone.

SOA records cause no additional section processing.

All times are in units of seconds.

Most of these fields are pertinent only for name server maintenance operations. However, MINIMUM is used in all query operations that retrieve RRs from a zone. Whenever a RR is sent in a response to a query, the TTL field is set to the maximum of the TTL field from the RR and the MINIMUM field in the appropriate SOA. Thus MINIMUM is a lower bound on the TTL field for all RRs in a zone. Note that this use of MINIMUM should occur when the RRs are copied into the response and not when the zone is loaded from a master file or via a zone transfer. The reason for this provison is to allow future dynamic update facilities to change the SOA RR with known semantics.

3.3.14. TXT RDATA format



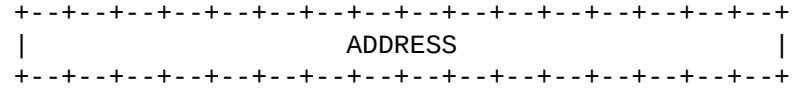
where:

TXT-DATA One or more <character-string>s.

TXT RRs are used to hold descriptive text. The semantics of the text depends on the domain where it is found.

3.4. Internet specific RRs

3.4.1. A RDATA format



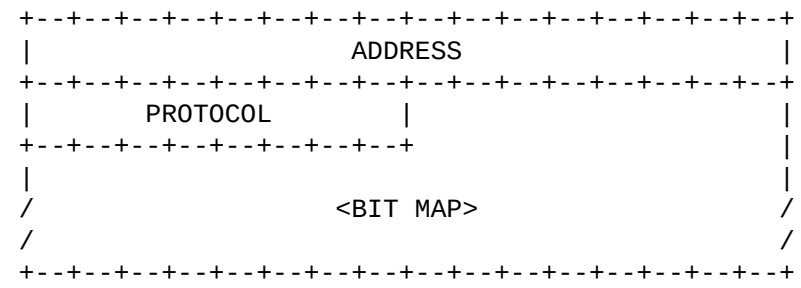
where:

ADDRESS A 32 bit Internet address.

Hosts that have multiple Internet addresses will have multiple A records.

A records cause no additional section processing. The RDATA section of an A line in a master file is an Internet address expressed as four decimal numbers separated by dots without any imbedded spaces (e.g., "10.2.0.52" or "192.0.5.6").

3.4.2. WKS RDATA format



where:

ADDRESS	An 32 bit Internet address
PROTOCOL	An 8 bit IP protocol number
<BIT MAP>	A variable length bit map. The bit map must be a multiple of 8 bits long.

The WKS record is used to describe the well known services supported by a particular protocol on a particular internet address. The PROTOCOL field specifies an IP protocol number, and the bit map has one bit per port of the specified protocol. The first bit corresponds to port 0, the second to port 1, etc. If the bit map does not include a bit for a protocol of interest, that bit is assumed zero. The appropriate values and mnemonics for ports and protocols are specified in [RFC-1010].

For example, if PROTOCOL=TCP (6), the 26th bit corresponds to TCP port 25 (SMTP). If this bit is set, a SMTP server should be listening on TCP port 25; if zero, SMTP service is not supported on the specified address.

The purpose of WKS RRs is to provide availability information for servers for TCP and UDP. If a server supports both TCP and UDP, or has multiple Internet addresses, then multiple WKS RRs are used.

WKS RRs cause no additional section processing.

In master files, both ports and protocols are expressed using mnemonics or decimal numbers.

3.5. IN-ADDR.ARPA domain

The Internet uses a special domain to support gateway location and Internet address to host mapping. Other classes may employ a similar strategy in other domains. The intent of this domain is to provide a guaranteed method to perform host address to host name mapping, and to facilitate queries to locate all gateways on a particular network in the Internet.

Note that both of these services are similar to functions that could be performed by inverse queries; the difference is that this part of the domain name space is structured according to address, and hence can guarantee that the appropriate data can be located without an exhaustive search of the domain space.

The domain begins at IN-ADDR.ARPA and has a substructure which follows the Internet addressing structure.

Domain names in the IN-ADDR.ARPA domain are defined to have up to four labels in addition to the IN-ADDR.ARPA suffix. Each label represents one octet of an Internet address, and is expressed as a character string for a decimal value in the range 0-255 (with leading zeros omitted except in the case of a zero octet which is represented by a single zero).

Host addresses are represented by domain names that have all four labels specified. Thus data for Internet address 10.2.0.52 is located at domain name 52.0.2.10.IN-ADDR.ARPA. The reversal, though awkward to read, allows zones to be delegated which are exactly one network of address space. For example, 10.IN-ADDR.ARPA can be a zone containing data for the ARPANET, while 26.IN-ADDR.ARPA can be a separate zone for MILNET. Address nodes are used to hold pointers to primary host names in the normal domain space.

Network numbers correspond to some non-terminal nodes at various depths in the IN-ADDR.ARPA domain, since Internet network numbers are either 1, 2, or 3 octets. Network nodes are used to hold pointers to the primary host names of gateways attached to that network. Since a gateway is, by definition, on more than one network, it will typically have two or more network nodes which point at it. Gateways will also have host level pointers at their fully qualified addresses.

Both the gateway pointers at network nodes and the normal host pointers at full address nodes use the PTR RR to point back to the primary domain names of the corresponding hosts.

For example, the IN-ADDR.ARPA domain will contain information about the ISI gateway between net 10 and 26, an MIT gateway from net 10 to MIT's

net 18, and hosts A.ISI.EDU and MULTICS.MIT.EDU. Assuming that ISI gateway has addresses 10.2.0.22 and 26.0.0.103, and a name MILNET-GW.ISI.EDU, and the MIT gateway has addresses 10.0.0.77 and 18.10.0.4 and a name GW.LCS.MIT.EDU, the domain database would contain:

```
    10.IN-ADDR.ARPA.           PTR MILNET-GW.ISI.EDU.
    10.IN-ADDR.ARPA.           PTR GW.LCS.MIT.EDU.
    18.IN-ADDR.ARPA.           PTR GW.LCS.MIT.EDU.
    26.IN-ADDR.ARPA.           PTR MILNET-GW.ISI.EDU.
    22.0.2.10.IN-ADDR.ARPA.    PTR MILNET-GW.ISI.EDU.
    103.0.0.26.IN-ADDR.ARPA.   PTR MILNET-GW.ISI.EDU.
    77.0.0.10.IN-ADDR.ARPA.    PTR GW.LCS.MIT.EDU.
    4.0.10.18.IN-ADDR.ARPA.    PTR GW.LCS.MIT.EDU.
    103.0.3.26.IN-ADDR.ARPA.   PTR A.ISI.EDU.
    6.0.0.10.IN-ADDR.ARPA.     PTR MULTICS.MIT.EDU.
```

Thus a program which wanted to locate gateways on net 10 would originate a query of the form QTYPE=PTR, QCLASS=IN, QNAME=10.IN-ADDR.ARPA. It would receive two RRs in response:

```
    10.IN-ADDR.ARPA.           PTR MILNET-GW.ISI.EDU.
    10.IN-ADDR.ARPA.           PTR GW.LCS.MIT.EDU.
```

The program could then originate QTYPE=A, QCLASS=IN queries for MILNET-GW.ISI.EDU. and GW.LCS.MIT.EDU. to discover the Internet addresses of these gateways.

A resolver which wanted to find the host name corresponding to Internet host address 10.0.0.6 would pursue a query of the form QTYPE=PTR, QCLASS=IN, QNAME=6.0.0.10.IN-ADDR.ARPA, and would receive:

```
    6.0.0.10.IN-ADDR.ARPA.     PTR MULTICS.MIT.EDU.
```

Several cautions apply to the use of these services:

- Since the IN-ADDR.ARPA special domain and the normal domain for a particular host or gateway will be in different zones, the possibility exists that that the data may be inconsistent.

- Gateways will often have two names in separate domains, only one of which can be primary.

- Systems that use the domain database to initialize their routing tables must start with enough gateway information to guarantee that they can access the appropriate name server.

- The gateway data only reflects the existence of a gateway in a manner equivalent to the current HOSTS.TXT file. It doesn't replace the dynamic availability information from GGP or EGP.

3.6. Defining new types, classes, and special namespaces

The previously defined types and classes are the ones in use as of the date of this memo. New definitions should be expected. This section makes some recommendations to designers considering additions to the existing facilities. The mailing list NAMEDROPPERS@SRI-NIC.ARPA is the forum where general discussion of design issues takes place.

In general, a new type is appropriate when new information is to be added to the database about an existing object, or we need new data formats for some totally new object. Designers should attempt to define types and their RDATA formats that are generally applicable to all classes, and which avoid duplication of information. New classes are appropriate when the DNS is to be used for a new protocol, etc which requires new class-specific data formats, or when a copy of the existing name space is desired, but a separate management domain is necessary.

New types and classes need mnemonics for master files; the format of the master files requires that the mnemonics for type and class be disjoint.

TYPE and CLASS values must be a proper subset of QTYPEs and QCLASSes respectively.

The present system uses multiple RRs to represent multiple values of a type rather than storing multiple values in the RDATA section of a single RR. This is less efficient for most applications, but does keep RRs shorter. The multiple RRs assumption is incorporated in some experimental work on dynamic update methods.

The present system attempts to minimize the duplication of data in the database in order to insure consistency. Thus, in order to find the address of the host for a mail exchange, you map the mail domain name to a host name, then the host name to addresses, rather than a direct mapping to host address. This approach is preferred because it avoids the opportunity for inconsistency.

In defining a new type of data, multiple RR types should not be used to create an ordering between entries or express different formats for equivalent bindings, instead this information should be carried in the body of the RR and a single type used. This policy avoids problems with caching multiple types and defining QTYPEs to match multiple types.

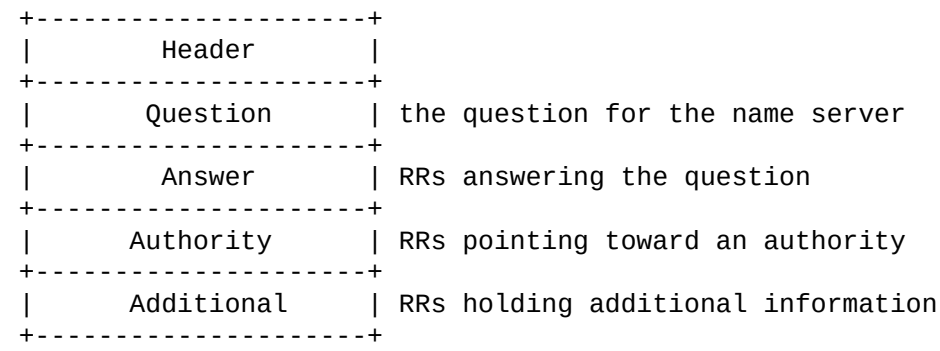
For example, the original form of mail exchange binding used two RR types one to represent a "closer" exchange (MD) and one to represent a "less close" exchange (MF). The difficulty is that the presence of one RR type in a cache doesn’t convey any information about the other because the query which acquired the cached information might have used a QTYPE of MF, MD, or MAILA (which matched both). The redesigned

service used a single type (MX) with a "preference" value in the RDATA section which can order different RRs. However, if any MX RRs are found in the cache, then all should be there.

4. MESSAGES

4.1. Format

All communications inside of the domain protocol are carried in a single format called a message. The top level format of message is divided into 5 sections (some of which are empty in certain cases) shown below:

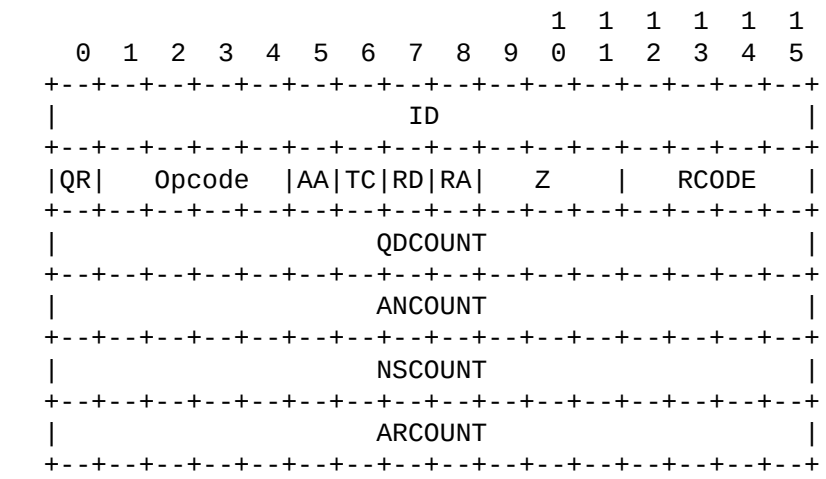


The header section is always present. The header includes fields that specify which of the remaining sections are present, and also specify whether the message is a query or a response, a standard query or some other opcode, etc.

The names of the sections after the header are derived from their use in standard queries. The question section contains fields that describe a question to a name server. These fields are a query type (QTYPE), a query class (QCLASS), and a query domain name (QNAME). The last three sections have the same format: a possibly empty list of concatenated resource records (RRs). The answer section contains RRs that answer the question; the authority section contains RRs that point toward an authoritative name server; the additional records section contains RRs which relate to the query, but are not strictly answers for the question.

4.1.1. Header section format

The header contains the following fields:



where:

ID A 16 bit identifier assigned by the program that generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.

QR A one bit field that specifies whether this message is a query (0), or a response (1).

OPCODE A four bit field that specifies kind of query in this message. This value is set by the originator of a query and copied into the response. The values are:

0 a standard query (QUERY)

1 an inverse query (IQUERY)

2 a server status request (STATUS)

3-15 reserved for future use

AA Authoritative Answer - this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in question section.

Note that the contents of the answer section may have multiple owner names because of aliases. The AA bit

corresponds to the name which matches the query name, or the first owner name in the answer section.

TC — TrunCation - specifies that this message was truncated due to length greater than that permitted on the transmission channel.

RD — Recursion Desired - this bit may be set in a query and is copied into the response. If RD is set, it directs the name server to pursue the query recursively. Recursive query support is optional.

RA — Recursion Available - this be is set or cleared in a response, and denotes whether recursive query support is available in the name server.

Z — Reserved for future use. Must be zero in all queries and responses.

RCODE — Response code - this 4 bit field is set as part of responses. The values have the following interpretation:

0	No error condition
1	Format error - The name server was unable to interpret the query.
2	Server failure - The name server was unable to process this query due to a problem with the name server.
3	Name Error - Meaningful only for responses from an authoritative name server, this code signifies that the domain name referenced in the query does not exist.
4	Not Implemented - The name server does not support the requested kind of query.
5	Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone

transfer) for particular data.

6-15 Reserved for future use.

QDCOUNT an unsigned 16 bit integer specifying the number of entries in the question section.

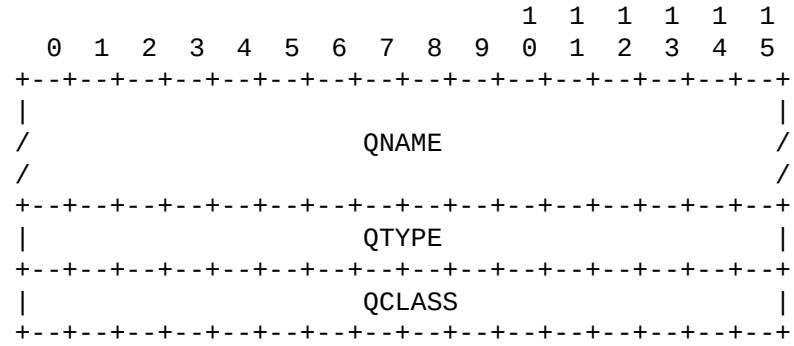
ANCOUNT an unsigned 16 bit integer specifying the number of resource records in the answer section.

NSCOUNT an unsigned 16 bit integer specifying the number of name server resource records in the authority records section.

ARCOUNT an unsigned 16 bit integer specifying the number of resource records in the additional records section.

4.1.2. Question section format

The question section is used to carry the "question" in most queries, i.e., the parameters that define what is being asked. The section contains QDCOUNT (usually 1) entries, each of the following format:



where:

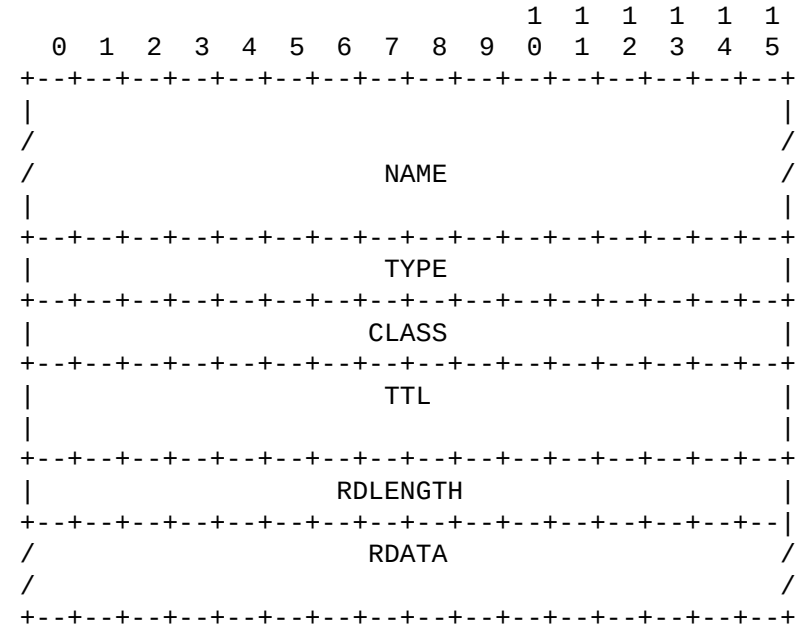
QNAME a domain name represented as a sequence of labels, where each label consists of a length octet followed by that number of octets. The domain name terminates with the zero length octet for the null label of the root. Note that this field may be an odd number of octets; no padding is used.

QTYPE a two octet code which specifies the type of the query. The values for this field include all codes valid for a TYPE field, together with some more general codes which can match more than one type of RR.

QCLASS a two octet code that specifies the class of the query. For example, the QCLASS field is IN for the Internet.

4.1.3. Resource record format

The answer, authority, and additional sections all share the same format: a variable number of resource records, where the number of records is specified in the corresponding count field in the header. Each resource record has the following format:



where:

NAME a domain name to which this resource record pertains.

TYPE two octets containing one of the RR type codes. This field specifies the meaning of the data in the RDATA field.

CLASS two octets which specify the class of the data in the RDATA field.

TTL a 32 bit unsigned integer that specifies the time interval (in seconds) that the resource record may be cached before it should be discarded. Zero values are interpreted to mean that the RR can only be used for the transaction in progress, and should not be cached.

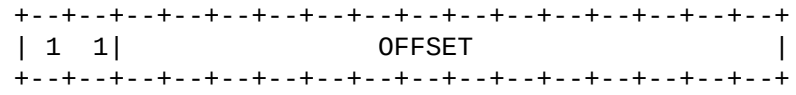
RDLENGTH an unsigned 16 bit integer that specifies the length in octets of the RDATA field.

RDATA a variable length string of octets that describes the resource. The format of this information varies according to the TYPE and CLASS of the resource record. For example, the if the TYPE is A and the CLASS is IN, the RDATA field is a 4 octet ARPA Internet address.

4.1.4. Message compression

In order to reduce the size of messages, the domain system utilizes a compression scheme which eliminates the repetition of domain names in a message. In this scheme, an entire domain name or a list of labels at the end of a domain name is replaced with a pointer to a prior occurance of the same name.

The pointer takes the form of a two octet sequence:



The first two bits are ones. This allows a pointer to be distinguished from a label, since the label must begin with two zero bits because labels are restricted to 63 octets or less. (The 10 and 01 combinations are reserved for future use.) The OFFSET field specifies an offset from the start of the message (i.e., the first octet of the ID field in the domain header). A zero offset specifies the first byte of the ID field, etc.

The compression scheme allows a domain name in a message to be represented as either:

- a sequence of labels ending in a zero octet

- a pointer

- a sequence of labels ending with a pointer

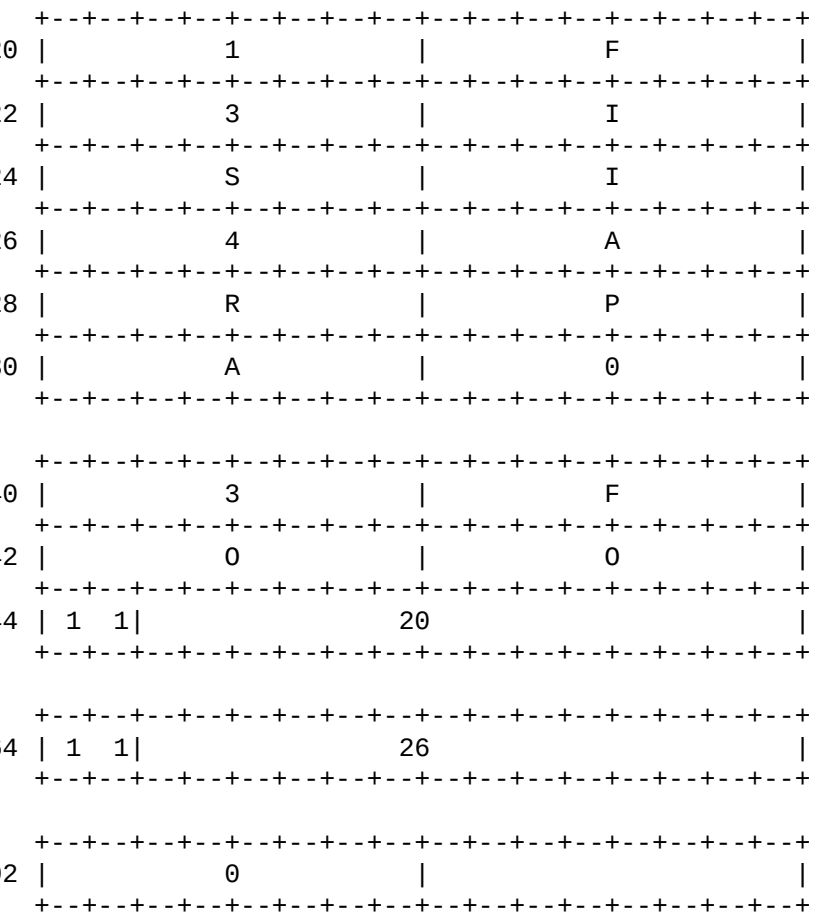
Pointers can only be used for occurances of a domain name where the format is not class specific. If this were not the case, a name server or resolver would be required to know the format of all RRs it handled. As yet, there are no such cases, but they may occur in future RDATA formats.

If a domain name is contained in a part of the message subject to a length field (such as the RDATA section of an RR), and compression is

used, the length of the compressed name is used in the length calculation, rather than the length of the expanded name.

Programs are free to avoid using pointers in messages they generate, although this will reduce datagram capacity, and may cause truncation. However all programs are required to understand arriving messages that contain pointers.

For example, a datagram might need to use the domain names F.ISI.ARPA, FOO.F.ISI.ARPA, ARPA, and the root. Ignoring the other fields of the message, these domain names might be represented as:



The domain name for F.ISI.ARPA is shown at offset 20. The domain name FOO.F.ISI.ARPA is shown at offset 40; this definition uses a pointer to concatenate a label for FOO to the previously defined F.ISI.ARPA. The domain name ARPA is defined at offset 64 using a pointer to the ARPA component of the name F.ISI.ARPA at 20; note that this pointer relies on ARPA being the last label in the string at 20. The root domain name is

defined by a single octet of zeros at 92; the root domain name has no labels.

4.2. Transport

The DNS assumes that messages will be transmitted as datagrams or in a byte stream carried by a virtual circuit. While virtual circuits can be used for any DNS activity, datagrams are preferred for queries due to their lower overhead and better performance. Zone refresh activities must use virtual circuits because of the need for reliable transfer.

The Internet supports name server access using TCP [RFC-793] on server port 53 (decimal) as well as datagram access using UDP [RFC-768] on UDP port 53 (decimal).

4.2.1. UDP usage

Messages sent using UDP user server port 53 (decimal).

Messages carried by UDP are restricted to 512 bytes (not counting the IP or UDP headers). Longer messages are truncated and the TC bit is set in the header.

UDP is not acceptable for zone transfers, but is the recommended method for standard queries in the Internet. Queries sent using UDP may be lost, and hence a retransmission strategy is required. Queries or their responses may be reordered by the network, or by processing in name servers, so resolvers should not depend on them being returned in order.

The optimal UDP retransmission policy will vary with performance of the Internet and the needs of the client, but the following are recommended:

- The client should try other servers and server addresses before repeating a query to a specific address of a server.

- The retransmission interval should be based on prior statistics if possible. Too aggressive retransmission can easily slow responses for the community at large. Depending on how well connected the client is to its expected servers, the minimum retransmission interval should be 2-5 seconds.

More suggestions on server selection and retransmission policy can be found in the resolver section of this memo.

4.2.2. TCP usage

Messages sent over TCP connections use server port 53 (decimal). The message is prefixed with a two byte length field which gives the message

length, excluding the two byte length field. This length field allows the low-level processing to assemble a complete message before beginning to parse it.

Several connection management policies are recommended:

- The server should not block other activities waiting for TCP data.

- The server should support multiple connections.

- The server should assume that the client will initiate connection closing, and should delay closing its end of the connection until all outstanding client requests have been satisfied.

- If the server needs to close a dormant connection to reclaim resources, it should wait until the connection has been idle for a period on the order of two minutes. In particular, the server should allow the SOA and AXFR request sequence (which begins a refresh operation) to be made on a single connection. Since the server would be unable to answer queries anyway, a unilateral close or reset may be used instead of a graceful close.

5. MASTER FILES

Master files are text files that contain RRs in text form. Since the contents of a zone can be expressed in the form of a list of RRs a master file is most often used to define a zone, though it can be used to list a cache's contents. Hence, this section first discusses the format of RRs in a master file, and then the special considerations when a master file is used to create a zone in some name server.

5.1. Format

The format of these files is a sequence of entries. Entries are predominantly line-oriented, though parentheses can be used to continue a list of items across a line boundary, and text literals can contain CRLF within the text. Any combination of tabs and spaces act as a delimiter between the separate items that make up an entry. The end of any line in the master file can end with a comment. The comment starts with a ";" (semicolon).

The following entries are defined:

    <blank>[<comment>]

```
    $ORIGIN <domain-name> [<comment>]

    $INCLUDE <file-name> [<domain-name>] [<comment>]

    <domain-name><rr> [<comment>]

    <blank><rr> [<comment>]
```

Blank lines, with or without comments, are allowed anywhere in the file.

Two control entries are defined: $ORIGIN and $INCLUDE. $ORIGIN is followed by a domain name, and resets the current origin for relative domain names to the stated name. $INCLUDE inserts the named file into the current file, and may optionally specify a domain name that sets the relative domain name origin for the included file. $INCLUDE may also have a comment. Note that a $INCLUDE entry never changes the relative origin of the parent file, regardless of changes to the relative origin made within the included file.

The last two forms represent RRs. If an entry for an RR begins with a blank, then the RR is assumed to be owned by the last stated owner. If an RR entry begins with a <domain-name>, then the owner name is reset.

<rr> contents take one of the following forms:

```
    [<TTL>] [<class>] <type> <RDATA>

    [<class>] [<TTL>] <type> <RDATA>
```

The RR begins with optional TTL and class fields, followed by a type and RDATA field appropriate to the type and class. Class and type use the standard mnemonics, TTL is a decimal integer. Omitted class and TTL values are default to the last explicitly stated values. Since type and class mnemonics are disjoint, the parse is unique. (Note that this order is different from the order used in examples and the order used in the actual RRs; the given order allows easier parsing and defaulting.)

<domain-name>s make up a large share of the data in the master file. The labels in the domain name are expressed as character strings and separated by dots. Quoting conventions allow arbitrary characters to be stored in domain names. Domain names that end in a dot are called absolute, and are taken as complete. Domain names which do not end in a dot are called relative; the actual domain name is the concatenation of the relative part with an origin specified in a $ORIGIN, $INCLUDE, or as an argument to the master file loading routine. A relative name is an error when no origin is available.

<character-string> is expressed in one or two ways: as a contiguous set of characters without interior spaces, or as a string beginning with a " and ending with a ". Inside a " delimited string any character can occur, except for a " itself, which must be quoted using \ (back slash).

Because these files are text files several special encodings are necessary to allow arbitrary data to be loaded. In particular:

of the root.

@ A free standing @ is used to denote the current origin.

\X where X is any character other than a digit (0-9), is used to quote that character so that its special meaning does not apply. For example, "\." can be used to place a dot character in a label.

\DDD where each D is a digit is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses are used to group data that crosses a line boundary. In effect, line terminations are not recognized within parentheses.

; Semicolon is used to start a comment; the remainder of the line is ignored.

5.2. Use of master files to define zones

When a master file is used to load a zone, the operation should be suppressed if any errors are encountered in the master file. The rationale for this is that a single error can have widespread consequences. For example, suppose that the RRs defining a delegation have syntax errors; then the server will return authoritative name errors for all names in the subzone (except in the case where the subzone is also present on the server).

Several other validity checks that should be performed in addition to insuring that the file is syntactically correct:

1. All RRs in the file should have the same class.

2. Exactly one SOA RR should be present at the top of the zone.

3. If delegations are present and glue information is required, it should be present.

4. Information present outside of the authoritative nodes in the zone should be glue information, rather than the result of an origin or similar error.

5.3. Master file example

The following is an example file which might be used to define the ISI.EDU zone.and is loaded with an origin of ISI.EDU:

```
@   IN  SOA     VENERA      Action\.domains (
                                 20     ; SERIAL
                                 7200   ; REFRESH
                                 600    ; RETRY
                                 3600000; EXPIRE
                                 60)    ; MINIMUM

        NS      A.ISI.EDU.
        NS      VENERA
        NS      VAXA
        MX      10      VENERA
        MX      20      VAXA

A       A       26.3.0.103

VENERA  A       10.1.0.52
        A       128.9.0.32

VAXA    A       10.2.0.27
        A       128.9.0.33


$INCLUDE <SUBSYS>ISI-MAILBOXES.TXT
```

Where the file <SUBSYS>ISI-MAILBOXES.TXT is:

```
    MOE     MB      A.ISI.EDU.
    LARRY   MB      A.ISI.EDU.
    CURLEY  MB      A.ISI.EDU.
    STOOGES MG      MOE
            MG      LARRY
            MG      CURLEY
```

Note the use of the \ character in the SOA RR to specify the responsible person mailbox "Action.domains@E.ISI.EDU".

6. NAME SERVER IMPLEMENTATION

6.1. Architecture

The optimal structure for the name server will depend on the host operating system and whether the name server is integrated with resolver operations, either by supporting recursive service, or by sharing its database with a resolver. This section discusses implementation considerations for a name server which shares a database with a resolver, but most of these concerns are present in any name server.

6.1.1. Control

A name server must employ multiple concurrent activities, whether they are implemented as separate tasks in the host's OS or multiplexing inside a single name server program. It is simply not acceptable for a name server to block the service of UDP requests while it waits for TCP data for refreshing or query activities. Similarly, a name server should not attempt to provide recursive service without processing such requests in parallel, though it may choose to serialize requests from a single client, or to regard identical requests from the same client as duplicates. A name server should not substantially delay requests while it reloads a zone from master files or while it incorporates a newly refreshed zone into its database.

6.1.2. Database

While name server implementations are free to use any internal data structures they choose, the suggested structure consists of three major parts:

- A "catalog" data structure which lists the zones available to this server, and a "pointer" to the zone data structure. The main purpose of this structure is to find the nearest ancestor zone, if any, for arriving standard queries.

- Separate data structures for each of the zones held by the name server.

- A data structure for cached data. (or perhaps separate caches for different classes)

All of these data structures can be implemented an identical tree structure format, with different data chained off the nodes in different parts: in the catalog the data is pointers to zones, while in the zone and cache data structures, the data will be RRs. In designing the tree framework the designer should recognize that query processing will need to traverse the tree using case-insensitive label comparisons; and that

in real data, a few nodes have a very high branching factor (100-1000 or more), but the vast majority have a very low branching factor (0-1).

One way to solve the case problem is to store the labels for each node in two pieces: a standardized-case representation of the label where all ASCII characters are in a single case, together with a bit mask that denotes which characters are actually of a different case. The branching factor diversity can be handled using a simple linked list for a node until the branching factor exceeds some threshold, and transitioning to a hash structure after the threshold is exceeded. In any case, hash structures used to store tree sections must insure that hash functions and procedures preserve the casing conventions of the DNS.

The use of separate structures for the different parts of the database is motivated by several factors:

- The catalog structure can be an almost static structure that need change only when the system administrator changes the zones supported by the server. This structure can also be used to store parameters used to control refreshing activities.

- The individual data structures for zones allow a zone to be replaced simply by changing a pointer in the catalog. Zone refresh operations can build a new structure and, when complete, splice it into the database via a simple pointer replacement. It is very important that when a zone is refreshed, queries should not use old and new data simultaneously.

- With the proper search procedures, authoritative data in zones will always "hide", and hence take precedence over, cached data.

- Errors in zone definitions that cause overlapping zones, etc., may cause erroneous responses to queries, but problem determination is simplified, and the contents of one "bad" zone can’t corrupt another.

- Since the cache is most frequently updated, it is most vulnerable to corruption during system restarts. It can also become full of expired RR data. In either case, it can easily be discarded without disturbing zone data.

A major aspect of database design is selecting a structure which allows the name server to deal with crashes of the name server’s host. State information which a name server should save across system crashes

includes the catalog structure (including the state of refreshing for each zone) and the zone data itself.

6.1.3. Time

Both the TTL data for RRs and the timing data for refreshing activities depends on 32 bit timers in units of seconds. Inside the database, refresh timers and TTLs for cached data conceptually "count down", while data in the zone stays with constant TTLs.

A recommended implementation strategy is to store time in two ways: as a relative increment and as an absolute time. One way to do this is to use positive 32 bit numbers for one type and negative numbers for the other. The RRs in zones use relative times; the refresh timers and cache data use absolute times. Absolute numbers are taken with respect to some known origin and converted to relative values when placed in the response to a query. When an absolute TTL is negative after conversion to relative, then the data is expired and should be ignored.

6.2. Standard query processing

The major algorithm for standard query processing is presented in [RFC-1034].

When processing queries with QCLASS=*, or some other QCLASS which matches multiple classes, the response should never be authoritative unless the server can guarantee that the response covers all classes.

When composing a response, RRs which are to be inserted in the additional section, but duplicate RRs in the answer or authority sections, may be omitted from the additional section.

When a response is so long that truncation is required, the truncation should start at the end of the response and work forward in the datagram. Thus if there is any data for the authority section, the answer section is guaranteed to be unique.

The MINIMUM value in the SOA should be used to set a floor on the TTL of data distributed from a zone. This floor function should be done when the data is copied into a response. This will allow future dynamic update protocols to change the SOA MINIMUM field without ambiguous semantics.

6.3. Zone refresh and reload processing

In spite of a server’s best efforts, it may be unable to load zone data from a master file due to syntax errors, etc., or be unable to refresh a zone within the its expiration parameter. In this case, the name server

should answer queries as if it were not supposed to possess the zone.

If a master is sending a zone out via AXFR, and a new version is created during the transfer, the master should continue to send the old version if possible. In any case, it should never send part of one version and part of another. If completion is not possible, the master should reset the connection on which the zone transfer is taking place.

6.4. Inverse queries (Optional)

Inverse queries are an optional part of the DNS. Name servers are not required to support any form of inverse queries. If a name server receives an inverse query that it does not support, it returns an error response with the "Not Implemented" error set in the header. While inverse query support is optional, all name servers must be at least able to return the error response.

6.4.1. The contents of inverse queries and responses

Inverse queries reverse the mappings performed by standard query operations; while a standard query maps a domain name to a resource, an inverse query maps a resource to a domain name. For example, a standard query might bind a domain name to a host address; the corresponding inverse query binds the host address to a domain name.

Inverse queries take the form of a single RR in the answer section of the message, with an empty question section. The owner name of the query RR and its TTL are not significant. The response carries questions in the question section which identify all names possessing the query RR WHICH THE NAME SERVER KNOWS. Since no name server knows about all of the domain name space, the response can never be assumed to be complete. Thus inverse queries are primarily useful for database management and debugging activities. Inverse queries are NOT an acceptable method of mapping host addresses to host names; use the IN-ADDR.ARPA domain instead.

Where possible, name servers should provide case-insensitive comparisons for inverse queries. Thus an inverse query asking for an MX RR of "Venera.isi.edu" should get the same response as a query for "VENERA.ISI.EDU"; an inverse query for HINFO RR "IBM-PC UNIX" should produce the same result as an inverse query for "IBM-pc unix". However, this cannot be guaranteed because name servers may possess RRs that contain character strings but the name server does not know that the data is character.

When a name server processes an inverse query, it either returns:

1. zero, one, or multiple domain names for the specified resource as QNAMEs in the question section

2. an error code indicating that the name server doesn't support inverse mapping of the specified resource type.

When the response to an inverse query contains one or more QNAMEs, the owner name and TTL of the RR in the answer section which defines the inverse query is modified to exactly match an RR found at the first QNAME.

RRs returned in the inverse queries cannot be cached using the same mechanism as is used for the replies to standard queries. One reason for this is that a name might have multiple RRs of the same type, and only one would appear. For example, an inverse query for a single address of a multiply homed host might create the impression that only one address existed.

6.4.2. Inverse query and response example The overall structure of an inverse query for retrieving the domain name that corresponds to Internet address 10.1.0.52 is shown below:

```
                  +-----------------------------------------+
   Header         |          OPCODE=IQUERY, ID=997          |
                  +-----------------------------------------+
   Question       |                 <empty>                 |
                  +-----------------------------------------+
   Answer         |        <anyname> A IN 10.1.0.52         |
                  +-----------------------------------------+
   Authority      |                 <empty>                 |
                  +-----------------------------------------+
   Additional     |                 <empty>                 |
                  +-----------------------------------------+
```

This query asks for a question whose answer is the Internet style address 10.1.0.52. Since the owner name is not known, any domain name can be used as a placeholder (and is ignored). A single octet of zero, signifying the root, is usually used because it minimizes the length of the message. The TTL of the RR is not significant. The response to this query might be:

```
                  +-----------------------------------------+
    Header        |         OPCODE=RESPONSE, ID=997         |
                  +-----------------------------------------+
    Question      |QTYPE=A, QCLASS=IN, QNAME=VENERA.ISI.EDU |
                  +-----------------------------------------+
    Answer        |  VENERA.ISI.EDU  A IN 10.1.0.52         |
                  +-----------------------------------------+
    Authority     |                 <empty>                 |
                  +-----------------------------------------+
    Additional    |                 <empty>                 |
                  +-----------------------------------------+
```

Note that the QTYPE in a response to an inverse query is the same as the TYPE field in the answer section of the inverse query. Responses to inverse queries may contain multiple questions when the inverse is not unique. If the question section in the response is not empty, then the RR in the answer section is modified to correspond to be an exact copy of an RR at the first QNAME.

6.4.3. Inverse query processing

Name servers that support inverse queries can support these operations through exhaustive searches of their databases, but this becomes impractical as the size of the database increases. An alternative approach is to invert the database according to the search key.

For name servers that support multiple zones and a large amount of data, the recommended approach is separate inversions for each zone. When a particular zone is changed during a refresh, only its inversions need to be redone.

Support for transfer of this type of inversion may be included in future versions of the domain system, but is not supported in this version.

6.5. Completion queries and responses

The optional completion services described in RFC-882 and RFC-883 have been deleted. Redesigned services may become available in the future.

7. RESOLVER IMPLEMENTATION

The top levels of the recommended resolver algorithm are discussed in [RFC-1034]. This section discusses implementation details assuming the database structure suggested in the name server implementation section of this memo.

7.1. Transforming a user request into a query

The first step a resolver takes is to transform the client's request, stated in a format suitable to the local OS, into a search specification for RRs at a specific name which match a specific QTYPE and QCLASS. Where possible, the QTYPE and QCLASS should correspond to a single type and a single class, because this makes the use of cached data much simpler. The reason for this is that the presence of data of one type in a cache doesn't confirm the existence or non-existence of data of other types, hence the only way to be sure is to consult an authoritative source. If QCLASS=* is used, then authoritative answers won't be available.

Since a resolver must be able to multiplex multiple requests if it is to perform its function efficiently, each pending request is usually represented in some block of state information. This state block will typically contain:

- A timestamp indicating the time the request began.
  The timestamp is used to decide whether RRs in the database can be used or are out of date. This timestamp uses the absolute time format previously discussed for RR storage in zones and caches. Note that when an RRs TTL indicates a relative time, the RR must be timely, since it is part of a zone. When the RR has an absolute time, it is part of a cache, and the TTL of the RR is compared against the timestamp for the start of the request.

  Note that using the timestamp is superior to using a current time, since it allows RRs with TTLs of zero to be entered in the cache in the usual manner, but still used by the current request, even after intervals of many seconds due to system load, query retransmission timeouts, etc.

- Some sort of parameters to limit the amount of work which will be performed for this request.

  The amount of work which a resolver will do in response to a client request must be limited to guard against errors in the database, such as circular CNAME references, and operational problems, such as network partition which prevents the

resolver from accessing the name servers it needs. While local limits on the number of times a resolver will retransmit a particular query to a particular name server address are essential, the resolver should have a global per-request counter to limit work on a single request. The counter should be set to some initial value and decremented whenever the resolver performs any action (retransmission timeout, retransmission, etc.) If the counter passes zero, the request is terminated with a temporary error.

Note that if the resolver structure allows one request to start others in parallel, such as when the need to access a name server for one request causes a parallel resolve for the name server's addresses, the spawned request should be started with a lower counter. This prevents circular references in the database from starting a chain reaction of resolver activity.

- The SLIST data structure discussed in [RFC-1034].

  This structure keeps track of the state of a request if it must wait for answers from foreign name servers.

7.2. Sending the queries

As described in [RFC-1034], the basic task of the resolver is to formulate a query which will answer the client's request and direct that query to name servers which can provide the information. The resolver will usually only have very strong hints about which servers to ask, in the form of NS RRs, and may have to revise the query, in response to CNAMEs, or revise the set of name servers the resolver is asking, in response to delegation responses which point the resolver to name servers closer to the desired information. In addition to the information requested by the client, the resolver may have to call upon its own services to determine the address of name servers it wishes to contact.

In any case, the model used in this memo assumes that the resolver is multiplexing attention between multiple requests, some from the client, and some internally generated. Each request is represented by some state information, and the desired behavior is that the resolver transmit queries to name servers in a way that maximizes the probability that the request is answered, minimizes the time that the request takes, and avoids excessive transmissions. The key algorithm uses the state information of the request to select the next name server address to query, and also computes a timeout which will cause the next action should a response not arrive. The next action will usually be a transmission to some other server, but may be a temporary error to the

client.

The resolver always starts with a list of server names to query (SLIST). This list will be all NS RRs which correspond to the nearest ancestor zone that the resolver knows about. To avoid startup problems, the resolver should have a set of default servers which it will ask should it have no current NS RRs which are appropriate. The resolver then adds to SLIST all of the known addresses for the name servers, and may start parallel requests to acquire the addresses of the servers when the resolver has the name, but no addresses, for the name servers.

To complete initialization of SLIST, the resolver attaches whatever history information it has to the each address in SLIST. This will usually consist of some sort of weighted averages for the response time of the address, and the batting average of the address (i.e., how often the address responded at all to the request). Note that this information should be kept on a per address basis, rather than on a per name server basis, because the response time and batting average of a particular server may vary considerably from address to address. Note also that this information is actually specific to a resolver address / server address pair, so a resolver with multiple addresses may wish to keep separate histories for each of its addresses. Part of this step must deal with addresses which have no such history; in this case an expected round trip time of 5-10 seconds should be the worst case, with lower estimates for the same local network, etc.

Note that whenever a delegation is followed, the resolver algorithm reinitializes SLIST.

The information establishes a partial ranking of the available name server addresses. Each time an address is chosen and the state should be altered to prevent its selection again until all other addresses have been tried. The timeout for each transmission should be 50-100% greater than the average predicted value to allow for variance in response.

Some fine points:

- The resolver may encounter a situation where no addresses are available for any of the name servers named in SLIST, and where the servers in the list are precisely those which would normally be used to look up their own addresses. This situation typically occurs when the glue address RRs have a smaller TTL than the NS RRs marking delegation, or when the resolver caches the result of a NS search. The resolver should detect this condition and restart the search at the next ancestor zone, or alternatively at the root.

- If a resolver gets a server error or other bizarre response from a name server, it should remove it from SLIST, and may wish to schedule an immediate transmission to the next candidate server address.

7.3. Processing responses

The first step in processing arriving response datagrams is to parse the response. This procedure should include:

- Check the header for reasonableness. Discard datagrams which are queries when responses are expected.

- Parse the sections of the message, and insure that all RRs are correctly formatted.

- As an optional step, check the TTLs of arriving data looking for RRs with excessively long TTLs. If a RR has an excessively long TTL, say greater than 1 week, either discard the whole response, or limit all TTLs in the response to 1 week.

The next step is to match the response to a current resolver request. The recommended strategy is to do a preliminary matching using the ID field in the domain header, and then to verify that the question section corresponds to the information currently desired. This requires that the transmission algorithm devote several bits of the domain ID field to a request identifier of some sort. This step has several fine points:

- Some name servers send their responses from different addresses than the one used to receive the query. That is, a resolver cannot rely that a response will come from the same address which it sent the corresponding query to. This name server bug is typically encountered in UNIX systems.

- If the resolver retransmits a particular request to a name server it should be able to use a response from any of the transmissions. However, if it is using the response to sample the round trip time to access the name server, it must be able to determine which transmission matches the response (and keep transmission times for each outgoing message), or only calculate round trip times based on initial transmissions.

- A name server will occasionally not have a current copy of a zone which it should have according to some NS RRs. The resolver should simply remove the name server from the current SLIST, and continue.

7.4. Using the cache

In general, we expect a resolver to cache all data which it receives in responses since it may be useful in answering future client requests. However, there are several types of data which should not be cached:

- When several RRs of the same type are available for a particular owner name, the resolver should either cache them all or none at all. When a response is truncated, and a resolver doesn't know whether it has a complete set, it should not cache a possibly partial set of RRs.

- Cached data should never be used in preference to authoritative data, so if caching would cause this to happen the data should not be cached.

- The results of an inverse query should not be cached.

- The results of standard queries where the QNAME contains "*" labels if the data might be used to construct wildcards. The reason is that the cache does not necessarily contain existing RRs or zone boundary information which is necessary to restrict the application of the wildcard RRs.

- RR data in responses of dubious reliability. When a resolver receives unsolicited responses or RR data other than that requested, it should discard it without caching it. The basic implication is that all sanity checks on a packet should be performed before any of it is cached.

In a similar vein, when a resolver has a set of RRs for some name in a response, and wants to cache the RRs, it should check its cache for already existing RRs. Depending on the circumstances, either the data in the response or the cache is preferred, but the two should never be combined. If the data in the response is from authoritative data in the answer section, it is always preferred.

8. MAIL SUPPORT

The domain system defines a standard for mapping mailboxes into domain names, and two methods for using the mailbox information to derive mail routing information. The first method is called mail exchange binding and the other method is mailbox binding. The mailbox encoding standard and mail exchange binding are part of the DNS official protocol, and are the recommended method for mail routing in the Internet. Mailbox binding is an experimental feature which is still under development and subject to change.

The mailbox encoding standard assumes a mailbox name of the form "<local-part>@<mail-domain>". While the syntax allowed in each of these sections varies substantially between the various mail internets, the preferred syntax for the ARPA Internet is given in [RFC-822].

The DNS encodes the <local-part> as a single label, and encodes the <mail-domain> as a domain name. The single label from the <local-part> is prefaced to the domain name from <mail-domain> to form the domain name corresponding to the mailbox. Thus the mailbox HOSTMASTER@SRI-NIC.ARPA is mapped into the domain name HOSTMASTER.SRI-NIC.ARPA. If the <local-part> contains dots or other special characters, its representation in a master file will require the use of backslash quoting to ensure that the domain name is properly encoded. For example, the mailbox Action.domains@ISI.EDU would be represented as Action\.domains.ISI.EDU.

8.1. Mail exchange binding

Mail exchange binding uses the <mail-domain> part of a mailbox specification to determine where mail should be sent. The <local-part> is not even consulted. [RFC-974] specifies this method in detail, and should be consulted before attempting to use mail exchange support.

One of the advantages of this method is that it decouples mail destination naming from the hosts used to support mail service, at the cost of another layer of indirection in the lookup function. However, the addition layer should eliminate the need for complicated "%", "!", etc encodings in <local-part>.

The essence of the method is that the <mail-domain> is used as a domain name to locate type MX RRs which list hosts willing to accept mail for <mail-domain>, together with preference values which rank the hosts according to an order specified by the administrators for <mail-domain>.

In this memo, the <mail-domain> ISI.EDU is used in examples, together with the hosts VENERA.ISI.EDU and VAXA.ISI.EDU as mail exchanges for ISI.EDU. If a mailer had a message for Mockapetris@ISI.EDU, it would route it by looking up MX RRs for ISI.EDU. The MX RRs at ISI.EDU name VENERA.ISI.EDU and VAXA.ISI.EDU, and type A queries can find the host addresses.

8.2. Mailbox binding (Experimental)

In mailbox binding, the mailer uses the entire mail destination specification to construct a domain name. The encoded domain name for the mailbox is used as the QNAME field in a QTYPE=MAILB query.

Several outcomes are possible for this query:

1. The query can return a name error indicating that the mailbox does not exist as a domain name.

   In the long term, this would indicate that the specified mailbox doesn't exist. However, until the use of mailbox binding is universal, this error condition should be interpreted to mean that the organization identified by the global part does not support mailbox binding. The appropriate procedure is to revert to exchange binding at this point.

2. The query can return a Mail Rename (MR) RR.

   The MR RR carries new mailbox specification in its RDATA field. The mailer should replace the old mailbox with the new one and retry the operation.

3. The query can return a MB RR.

   The MB RR carries a domain name for a host in its RDATA field. The mailer should deliver the message to that host via whatever protocol is applicable, e.g., b,SMTP.

4. The query can return one or more Mail Group (MG) RRs.

   This condition means that the mailbox was actually a mailing list or mail group, rather than a single mailbox. Each MG RR has a RDATA field that identifies a mailbox that is a member of the group. The mailer should deliver a copy of the message to each member.

5. The query can return a MB RR as well as one or more MG RRs.

   This condition means the the mailbox was actually a mailing list. The mailer can either deliver the message to the host specified by the MB RR, which will in turn do the delivery to all members, or the mailer can use the MG RRs to do the expansion itself.

In any of these cases, the response may include a Mail Information (MINFO) RR. This RR is usually associated with a mail group, but is legal with a MB. The MINFO RR identifies two mailboxes. One of these identifies a responsible person for the original mailbox name. This mailbox should be used for requests to be added to a mail group, etc. The second mailbox name in the MINFO RR identifies a mailbox that should receive error messages for mail failures. This is particularly appropriate for mailing lists when errors in member names should be reported to a person other than the one who sends a message to the list.

New fields may be added to this RR in the future.

9. REFERENCES and BIBLIOGRAPHY

[Dyer 87] S. Dyer, F. Hsu, "Hesiod", Project Athena Technical Plan - Name Service, April 1987, version 1.9.

Describes the fundamentals of the Hesiod name service.

[IEN-116] J. Postel, "Internet Name Server", IEN-116, USC/Information Sciences Institute, August 1979.

A name service obsoleted by the Domain Name System, but still in use.

[Quarterman 86] J. Quarterman, and J. Hoskins, "Notable Computer Networks", Communications of the ACM, October 1986, volume 29, number 10.

[RFC-742] K. Harrenstien, "NAME/FINGER", RFC-742, Network Information Center, SRI International, December 1977.

[RFC-768] J. Postel, "User Datagram Protocol", RFC-768, USC/Information Sciences Institute, August 1980.

[RFC-793] J. Postel, "Transmission Control Protocol", RFC-793, USC/Information Sciences Institute, September 1981.

[RFC-799] D. Mills, "Internet Name Domains", RFC-799, COMSAT, September 1981.

Suggests introduction of a hierarchy in place of a flat name space for the Internet.

[RFC-805] J. Postel, "Computer Mail Meeting Notes", RFC-805, USC/Information Sciences Institute, February 1982.

[RFC-810] E. Feinler, K. Harrenstien, Z. Su, and V. White, "DOD Internet Host Table Specification", RFC-810, Network Information Center, SRI International, March 1982.

Obsolete. See RFC-952.

[RFC-811] K. Harrenstien, V. White, and E. Feinler, "Hostnames Server", RFC-811, Network Information Center, SRI International, March 1982.

Obsolete. See RFC-953.

[RFC-812] K. Harrenstien, and V. White, "NICNAME/WHOIS", RFC-812, Network Information Center, SRI International, March 1982.

[RFC-819] Z. Su, and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC-819, Network Information Center, SRI International, August 1982.

Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-821] J. Postel, "Simple Mail Transfer Protocol", RFC-821, USC/Information Sciences Institute, August 1980.

[RFC-830] Z. Su, "A Distributed System for Internet Name Service", RFC-830, Network Information Center, SRI International, October 1982.

Early thoughts on the design of the domain system. Current implementation is completely different.

[RFC-882] P. Mockapetris, "Domain names - Concepts and Facilities," RFC-882, USC/Information Sciences Institute, November 1983.

Superceeded by this memo.

[RFC-883] P. Mockapetris, "Domain names - Implementation and Specification," RFC-883, USC/Information Sciences Institute, November 1983.

Superceeded by this memo.

[RFC-920] J. Postel and J. Reynolds, "Domain Requirements", RFC-920, USC/Information Sciences Institute, October 1984.

Explains the naming scheme for top level domains.

[RFC-952] K. Harrenstien, M. Stahl, E. Feinler, "DoD Internet Host Table Specification", RFC-952, SRI, October 1985.

Specifies the format of HOSTS.TXT, the host/address table replaced by the DNS.

[RFC-953] K. Harrenstien, M. Stahl, E. Feinler, "HOSTNAME Server", RFC-953, SRI, October 1985.

This RFC contains the official specification of the hostname server protocol, which is obsoleted by the DNS. This TCP based protocol accesses information stored in the RFC-952 format, and is used to obtain copies of the host table.

[RFC-973] P. Mockapetris, "Domain System Changes and Observations", RFC-973, USC/Information Sciences Institute, January 1986.

Describes changes to RFC-882 and RFC-883 and reasons for them.

[RFC-974] C. Partridge, "Mail routing and the domain system", RFC-974, CSNET CIC BBN Labs, January 1986.

Describes the transition from HOSTS.TXT based mail addressing to the more powerful MX system used with the domain system.

[RFC-1001] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and Methods", RFC-1001, March 1987.

This RFC and RFC-1002 are a preliminary design for NETBIOS on top of TCP/IP which proposes to base NetBIOS name service on top of the DNS.

[RFC-1002] NetBIOS Working Group, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed Specifications", RFC-1002, March 1987.

[RFC-1010] J. Reynolds, and J. Postel, "Assigned Numbers", RFC-1010, USC/Information Sciences Institute, May 1987.

Contains socket numbers and mnemonics for host names, operating systems, etc.

[RFC-1031] W. Lazear, "MILNET Name Domain Transition", RFC-1031, November 1987.

Describes a plan for converting the MILNET to the DNS.

[RFC-1032] M. Stahl, "Establishing a Domain - Guidelines for Administrators", RFC-1032, November 1987.

Describes the registration policies used by the NIC to administer the top level domains and delegate subzones.

[RFC-1033] M. Lottor, "Domain Administrators Operations Guide", RFC-1033, November 1987.

A cookbook for domain administrators.

[Solomon 82] M. Solomon, L. Landweber, and D. Neuhengen, "The CSNET Name Server", Computer Networks, vol 6, nr 3, July 1982.

Describes a name service for CSNET which is independent from the DNS and DNS use in the CSNET.

Index

```
QCLASS   13
QTYPE   12

RDATA   12
RDLENGTH  11

Secondary server   5
SOA   12
Stub resolvers   7

TCP   32
TXT   12
TYPE   11

UDP   32

WKS   12
```

Network Working Group
Request for Comments: 3761
Obsoletes: 2916
Category: Standards Track

P. Faltstrom
Cisco Systems, Inc.
M. Mealling
VeriSign

April 2004

The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM)

Status of this Memo

Copyright Notice

Abstract

This document discusses the use of the Domain Name System (DNS) for storage of E.164 numbers. More specifically, how DNS can be used for identifying available services connected to one E.164 number. It specifically obsoletes RFC 2916 to bring it in line with the Dynamic Delegation Discovery System (DDDS) Application specification found in the document series specified in RFC 3401. It is very important to note that it is impossible to read and understand this document without reading the documents discussed in RFC 3401.

Table of Contents

1. Introduction

This document discusses the use of the Domain Name System (DNS) for storage of E.164 numbers. More specifically, how DNS can be used for identifying available services connected to one E.164 number. It specifically obsoletes RFC 2916 to bring it in line with the Dynamic Delegation Discovery System (DDDS) Application specification found in the document series specified in RFC 3401 [6]. It is very important to note that it is impossible to read and understand this document without reading the documents discussed in RFC 3401 [6].

Through transformation of International Public Telecommunication Numbers in the international format [5], called within this document E.164 numbers, into DNS names and the use of existing DNS services like delegation through NS records and NAPTR records, one can look up what services are available for a specific E.164 in a decentralized way with distributed management of the different levels in the lookup process.

The domain "e164.arpa" is being populated in order to provide the infrastructure in DNS for storage of E.164 numbers. In order to facilitate distributed operations, this domain is divided into subdomains. Holders of E.164 numbers which want to be listed in DNS should contact the appropriate zone administrator according to the

policy which is attached to the zone. One should start looking for this information by examining the SOA resource record associated with the zone, just like in normal DNS operations.

Of course, as with other domains, policies for such listings will be controlled on a subdomain basis and may differ in different parts of the world.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [1].

All other capitalized terms are taken from the vocabulary found in the DDDS algorithm specification found in RFC 3403 [2].

1.2. Use for these mechanisms for private dialing plans

This document describes the operation of these mechanisms in the context of numbers allocated according to the ITU-T recommendation E.164. The same mechanisms might be used for private dialing plans. If these mechanisms are re-used, the suffix used for the private dialing plan MUST NOT be e164.arpa, to avoid conflict with this specification. Parties to the private dialing plan will need to know the suffix used by their private dialing plan for correct operation of these mechanisms. Further, the application unique string used SHOULD be the full number as specified, but without the leading '+', and such private use MUST NOT be called "ENUM".

1.3. Application of local policy

The Order field in the NAPTR record specifies in what order the DNS records are to be interpreted. This is because DNS does not guarantee the order of records returned in the answer section of a DNS packet. In most ENUM cases this isn't an issue because the typical regular expression will be '!^.*$!' since the first query often results in a terminal Rule.

But there are other cases (non-terminal Rules) where two different Rules both match the given Application Unique String. As each Rule is evaluated within the algorithm, one may match a more significant piece of the AUS than the other. For example, by using a non-terminal NAPTR a given set of numbers is sent to some private-dialing-plan-specific zone. Within that zone there are two Rules that state that if a match is for the entire exchange and the service is SIP related then the first, SIP-specific rule is used. But the other Rule matches a longer piece of the AUS, specifying that for

some other service (instant messaging) that the Rule denotes a departmental level service. If the shorter matching Rule comes before the longer match, it can 'mask' the other rules. Thus, the order in which each Rule is tested against the AUS is an important corner case that many DDDS applications take advantage of.

In the case where the zone authority wishes to state that two Rules have the same effect or are identical in usage, then the Order for those records is set to the same value. In that case, the Preference is used to specify a locally over-ridable suggestion by the zone authority that one Rule might simply be better than another for some reason.

For ENUM this specifies where a client is allowed to apply local policy and where it is not. The Order field in the NAPTR is a request from the holder of the E.164 number that the records be handled in a specific way. The Preference field is merely a suggestion from that E.164 holder that one record might be better than another. A client implementing ENUM MUST adhere to the Order field but can simply take the Preference value "on advisement" as part of a client context specific selection method.

2. The ENUM Application Specifications

This template defines the ENUM DDDS Application according to the rules and requirements found in [7]. The DDDS database used by this Application is found in [2] which is the document that defines the NAPTR DNS Resource Record type.

ENUM is only applicable for E.164 numbers. ENUM compliant applications MUST only query DNS for what it believes is an E.164 number. Since there are numerous dialing plans which can change over time, it is probably impossible for a client application to have perfect knowledge about every valid and dialable E.164 number. Therefore a client application, doing everything within its power, can end up with what it thinks is a syntactically correct E.164 number which in reality is not actually valid or dialable. This implies that applications MAY send DNS queries when, for example, a user mistypes a number in a user interface. Because of this, there is the risk that collisions between E.164 numbers and non-E.164 numbers can occur. To mitigate this risk, the E2U portion of the service field MUST NOT be used for non-E.164 numbers.

2.1. Application Unique String

The Application Unique String is a fully qualified E.164 number minus any non-digit characters except for the '+' character which appears at the beginning of the number. The "+" is kept to provide a well understood anchor for the AUS in order to distinguish it from other telephone numbers that are not part of the E.164 namespace.

For example, the E.164 number could start out as "+44-116-496-0348". To ensure that no syntactic sugar is allowed into the AUS, all non-digits except for "+" are removed, yielding "+441164960348".

2.2. First Well Known Rule

The First Well Known Rule for this Application is the identity rule. The output of this rule is the same as the input. This is because the E.164 namespace and this Applications databases are organized in such a way that it is possible to go directly from the name to the smallest granularity of the namespace directly from the name itself.

Take the previous example, the AUS is "+441164960348". Applying the First Well Known Rule produces the exact same string, "+441164960348".

2.3. Expected Output

The output of the last DDDS loop is a Uniform Resource Identifier in its absolute form according to the 'absoluteURI' production in the Collected ABNF found in RFC2396 [4].

2.4. Valid Databases

At present only one DDDS Database is specified for this Application. "Dynamic Delegation Discovery System (DDDS) Part Three: The DNS Database" (RFC 3403) [2] specifies a DDDS Database that uses the NAPTR DNS resource record to contain the rewrite rules. The Keys for this database are encoded as domain-names.

The output of the First Well Known Rule for the ENUM Application is the E.164 number minus all non-digit characters except for the +. In order to convert this to a unique key in this Database the string is converted into a domain-name according to this algorithm:

1. Remove all characters with the exception of the digits. For example, the First Well Known Rule produced the Key "+442079460148". This step would simply remove the leading "+", producing "442079460148".

2. Put dots (".") between each digit. Example:
   4.4.2.0.7.9.4.6.0.1.4.8

3. Reverse the order of the digits. Example:
   8.4.1.0.6.4.9.7.0.2.4.4

4. Append the string ".e164.arpa" to the end. Example:
   8.4.1.0.6.4.9.7.0.2.4.4.e164.arpa

This domain-name is used to request NAPTR records which may contain the end result or, if the flags field is blank, produces new keys in the form of domain-names from the DNS.

Some nameserver implementations attempt to be intelligent about items that are inserted into the additional information section of a given DNS response. For example, BIND will attempt to determine if it is authoritative for a domain whenever it encodes one into a packet. If it is, then it will insert any A records it finds for that domain into the additional information section of the answer until the packet reaches the maximum length allowed. It is therefore potentially useful for a client to check for this additional information. It is also easy to contemplate an ENUM enhanced nameserver that understand the actual contents of the NAPTR records it is serving and inserts more appropriate information into the additional information section of the response. Thus, DNS servers MAY interpret Flag values and use that information to include appropriate resource records in the Additional Information portion of the DNS packet. Clients are encouraged to check for additional information but are not required to do so. See the Additional Information Processing section of RFC 3403 [2], Section 4.2 for more information on NAPTR records and the Additional Information section of a DNS response packet.

The character set used to encode the substitution expression is UTF-8. The allowed input characters are all those characters that are allowed anywhere in an E.164 number. The characters allowed to be in a Key are those that are currently defined for DNS domain-names.

2.4.1. Flags

This Database contains a field that contains flags that signal when the DDDS algorithm has finished. At this time only one flag, "U", is defined. This means that this Rule is the last one and that the output of the Rule is a URI [4]. See RFC 3404 [3].

If a client encounters a record with an unknown flag, it MUST ignore it and move to the next Rule. This test takes precedence over any ordering since flags can control the interpretation placed on fields.

A novel flag might change the interpretation of the regexp and/or replacement fields such that it is impossible to determine if a record matched a given target.

If this flag is not present then this rule is non-terminal. If a Rule is non-terminal then clients MUST use the Key produced by this Rewrite Rule as the new Key in the DDDS loop (i.e., causing the client to query for new NAPTR records at the domain-name that is the result of this Rule).

2.4.2. Services Parameters

Service Parameters for this Application take the following form and are found in the Service field of the NAPTR record.

```
service-field = "E2U" 1*(servicespec)
servicespec   = "+" enumservice
enumservice   = type 0*(subtypespec)
subtypespec   = ":" subtype
type          = 1*32(ALPHA / DIGIT)
subtype       = 1*32(ALPHA / DIGIT)
```

In other words, a non-optional "E2U" (used to denote ENUM only Rewrite Rules in order to mitigate record collisions) followed by 1 or more or more Enumservices which indicate what class of functionality a given end point offers. Each Enumservice is indicated by an initial '+' character.

2.4.2.1. ENUM Services

Enumservice specifications contain the functional specification (i.e., what it can be used for), the valid protocols, and the URI schemes that may be returned. Note that there is no implicit mapping between the textual string "type" or "subtype" in the grammar for the Enumservice and URI schemes or protocols. The mapping, if any, must be made explicit in the specification for the Enumservice itself. A registration of a specific Type also has to specify the Subtypes allowed.

The only exception to the registration rule is for Types and Subtypes used for experimental purposes, and those are to start with the facet "X-". These elements are unregistered, experimental, and should be used only with the active agreement of the parties exchanging them.

The registration mechanism is specified in Section 3.

2.5. What constitutes an 'Enum Resolver'?

There has been some confusion over what exactly an ENUM Resolver returns and what relation that has to the 'Note 1' section in RFC 3402. On first reading it seems as though it might be possible for an ENUM Resolver to return two Rules.

The ENUM algorithm always returns a single rule. Specific applications may have application-specific knowledge or facilities that allow them to present multiple results or speed selection, but these should never change the operation of the algorithm.

3. Registration mechanism for Enumservices

As specified in the ABNF found in Section 2.4.2, an 'enumservice' is made up of 'types' and 'subtypes'. For any given 'type', the allowable 'subtypes' must be specified in the registration. There is currently no concept of a registered 'subtype' outside the scope of a given 'type'. Thus the registration process uses the 'type' as its main key within the IANA Registry. While the combination of each type and all of its subtypes constitutes the allowed values for the 'enumservice' field, it is not sufficient to simply document those values. A complete registration will also include the allowed URI schemes, a functional specification, security considerations, intended usage, and any other information needed to allow for interoperability within ENUM. In order to be a registered ENUM Service, the entire specification, including the template, requires approval by the IESG and publication of the Enumservice registration specification as an RFC.

3.1. Registration Requirements

Service registration proposals are all expected to conform to various requirements laid out in the following sections.

3.1.1. Functionality Requirement

A registered Enumservice must be able to function as a selection mechanism when choosing one NAPTR resource record from another. That means that the registration MUST specify what is expected when using that very NAPTR record, and the URI which is the outcome of the use of it.

Specifically, a registered Enumservice MUST specify the URI scheme(s) that may be used for the Enumservice, and, when needed, other information which will have to be transferred into the URI resolution process itself (LDAP Distinguished Names, transferring of the AUS into the resulting URI, etc).

3.1.2. Naming requirement

An Enumservice MUST be unique in order to be useful as a selection criteria. Since an Enumservice is made up of a type and a type-dependent subtype, it is sufficient to require that the 'type' itself be unique. The 'type' MUST be unique, conform to the ABNF specified in Section 2.4.2, and MUST NOT start with the facet "X-" which is reserved for experimental, private use.

The subtype, being dependent on the type, MUST be unique within a given 'type'. It must conform to the ABNF specified in Section 2.4.2, and MUST NOT start with the facet "X-" which is reserved for experimental, private use. The subtype for one type MAY be the same as a subtype for a different registered type but it is not sufficient to simply reference another type's subtype. The function of each subtype must be specified in the context of the type being registered.

3.1.3. Security requirement

An analysis of security issues is required for all registered Enumservices. (This is in accordance with the basic requirements for all IETF protocols.)

All descriptions of security issues must be as accurate as possible regardless of registration tree. In particular, a statement that there are "no security issues associated with this Enumservice" must not be confused with "the security issues associated with this Enumservice have not been assessed".

There is no requirement that an Enumservice must be secure or completely free from risks. Nevertheless, all known security risks must be identified in the registration of an Enumservice.

The security considerations section of all registrations is subject to continuing evaluation and modification.

Some of the issues that should be looked at in a security analysis of an Enumservice are:

1. Complex Enumservices may include provisions for directives that institute actions on a user's resources. In many cases provision can be made to specify arbitrary actions in an unrestricted fashion which may then have devastating results. Especially if there is a risk for a new ENUM lookup, and because of that an infinite loop in the overall resolution process of the E.164.

2. Complex Enumservices may include provisions for directives that institute actions which, while not directly harmful, may result in disclosure of information that either facilitates a subsequent attack or else violates the users privacy in some way.

3. An Enumservice might be targeted for applications that require some sort of security assurance but do not provide the necessary security mechanisms themselves. For example, an Enumservice could be defined for storage of confidential security services information such as alarm systems or message service passcodes, which in turn require an external confidentiality service.

3.1.4. Publication Requirements

Proposals for Enumservices registrations MUST be published as one of the following documents; RFC on the Standards Track, Experimental RFC, or as a BCP.

IANA will retain copies of all Enumservice registration proposals and "publish" them as part of the Enumservice Registration tree itself.

3.2. Registration procedure

3.2.1. IANA Registration

Provided that the Enumservice has obtained the necessary approval, and the RFC is published, IANA will register the Enumservice and make the Enumservice registration available to the community in addition to the RFC publication itself.

3.2.1.1. Location of Enumservice Registrations

Enumservice registrations will be published in the IANA repository and made available via anonymous FTP at the following URI: "ftp://ftp.iana.org/assignments/enum-services/".

3.2.1.2. Change Control

Change control of Enumservices stay with the IETF via the RFC publication process. Especially, Enumservice registrations may not be deleted; Enumservices which are no longer believed appropriate for use can be declared OBSOLETE by publication of a new RFC and a change to their "intended use" field; such Enumservice will be clearly marked in the lists published by IANA.

3.2.2. Registration Template

Enumservice Type:

Enumservice Subtype(s):

URI Scheme(s):

Functional Specification:

Security considerations:

Intended usage: (One of COMMON, LIMITED USE or OBSOLETE)

Author:

Any other information that the author deems interesting:

Note: In the case where a particular field has no value, that field is left completely blank, especially in the case where a given type has no subtypes.

4. Examples

The examples below use theoretical services that contain Enumservices which might not make sense, but that are still used for educational purposes. For example, the protocol used is in some cases exactly the same string as the URI scheme. That was the specification in RFC 2916, but this 'default' specification of an Enumservice is no longer allowed. All Enumservices need to be registered explicitly by the procedure specified in section Section 3.

4.1. Example

```
$ORIGIN 3.8.0.0.6.9.2.3.6.1.4.4.e164.arpa.
   NAPTR 10 100 "u" "E2U+sip" "!^.*$!sip:info@example.com!" .
   NAPTR 10 101 "u" "E2U+h323" "!^.*$!h323:info@example.com!" .
   NAPTR 10 102 "u" "E2U+msg" "!^.*$!mailto:info@example.com!" .
```

This describes that the domain 3.8.0.0.6.9.2.3.6.1.4.4.e164.arpa. is preferably contacted by SIP, secondly via H.323 for voice, and thirdly by SMTP for messaging. Note that the tokens "sip", "h323", and "msg" are Types registered with IANA, and they have no implicit connection with the protocols or URI schemes with the same names.

In all cases, the next step in the resolution process is to use the resolution mechanism for each of the protocols, (specified by the URI schemes sip, h323 and mailto) to know what node to contact for each.

5. IANA Considerations

RFC 2916 (which this document replaces) requested IANA to delegate the E164.ARPA domain following instructions to be provided by the IAB. The domain was delegated according to those instructions. Names within this zone are to be delegated to parties according to the ITU-T Recommendation E.164. The names allocated should be hierarchic in accordance with ITU-T Recommendation E.164, and the codes should be assigned in accordance with that Recommendation.

IAB is to coordinate with ITU-T TSB if the technical contact for the domain e164.arpa is to change, as ITU-T TSB has an operational working relationship with this technical contact which needs to be reestablished.

Delegations in the zone e164.arpa (not delegations in delegated domains of e164.arpa) should be done after Expert Review, and the IESG will appoint a designated expert.

IANA has created a registry for Enumservices as specified in Section 3. Whenever a new Enumservice is registered by the RFC process in the IETF, IANA is at the time of publication of the RFC to register the Enumservice and add a pointer to the RFC itself.

6. Security Considerations

6.1. DNS Security

As ENUM uses DNS, which in its current form is an insecure protocol, there is no mechanism for ensuring that the data one gets back is authentic. As ENUM is deployed on the global Internet, it is expected to be a popular target for various kind of attacks, and attacking the underlying DNS infrastructure is one way of attacking the ENUM service itself.

There are multiple types of attacks that can happen against DNS that ENUM implementations should be aware of. The following threats are taken from Threat Analysis Of The Domain Name System [10]:

Packet Interception
: Some of the simplest threats against DNS are various forms of packet interception: monkey-in-the-middle attacks, eavesdropping on requests combined with spoofed responses that beat the real response back to the resolver, and so forth. In any of these

scenarios, the attacker can simply tell either party (usually the resolver) whatever it wants that party to believe. While packet interception attacks are far from unique to DNS, DNS's usual behavior of sending an entire query or response in a single unsigned, unencrypted UDP packet makes these attacks particularly easy for any bad guy with the ability to intercept packets on a shared or transit network.

ID Guessing and Query Prediction
Since the ID field in the DNS header is only a 16-bit field and the server UDP port associated with DNS is a well-known value, there are only 2**32 possible combinations of ID and client UDP port for a given client and server. Thus it is possible for a reasonable brute force attack to allow an attacker to masquerade as a trusted server. In most respects, this attack is similar to a packet interception attack except that it does not require the attacker to be on a transit or shared network.

Name-based Attacks
Name-based attacks use the actual DNS caching behavior as a tool to insert bad data into a victim's cache, thus potentially subverting subsequent decisions based on DNS names. Most examples occur with CNAME, NS and DNAME Resource Records as they redirect a victim's query to another location. The common thread in all of these attacks is that response messages allow the attacker to introduce arbitrary DNS names of the attacker's choosing and provide further information that the attacker claims is associated with those names; unless the victim has better knowledge of the data associated with those names, the victim is going to have a hard time defending against this class of attacks.

Betrayal By A Trusted Server
Another variation on the packet interception attack is the trusted server that turns out not to be so trustworthy, whether by accident or by intent. Many client machines are only configured with stub resolvers, and use trusted servers to perform all of their DNS queries on their behalf. In many cases the trusted server is furnished by the user's ISP and advertised to the client via DHCP or PPP options. Besides accidental betrayal of this trust relationship (via server bugs, successful server break-ins, etc), the server itself may be configured to give back answers that are not what the user would expect (whether in an honest attempt to help the user or to further some other goal such as furthering a business partnership between the ISP and some third party).

Denial of Service
As with any network service (or, indeed, almost any service of any kind in any domain of discourse), DNS is vulnerable to denial of service attacks. DNS servers are also at risk of being used as denial of service amplifiers, since DNS response packets tend to be significantly longer than DNS query packets.

Authenticated Denial of Domain Names
The existence of RR types whose absence causes an action other than immediate failure (such as missing MX and SRV RRs, which fail over to A RRs) constitutes a real threat. In the specific case of ENUM, even the immediate failure of a missing RR can be considered a problem as a method for changing call routing policy.

Because of these threats, a deployed ENUM service SHOULD include mechanisms which ameliorate these threats. Most of these threats can be solved by verifying the authenticity of the data via mechanisms such as DNSSEC [8] once it is deployed. Others, such and Denial Of Service attacks, cannot be solved by data authentication. It is important to remember that these threats include not only the NAPTR lookups themselves, but also the various records needed for the services to be useful (for example NS, MX, SRV and A records).

Even if DNSSEC is deployed, a service that uses ENUM for address translation should not blindly trust that the peer is the intended party as all kind of attacks against DNS can not be protected against with DNSSEC. A service should always authenticate the peers as part of the setup process for the service itself and never blindly trust any kind of addressing mechanism.

Finally, as an ENUM service will be implementing some type of security mechanism, software which implements ENUM MUST be prepared to receive DNSSEC and other standardized DNS security responses, including large responses, EDNS0 signaling, unknown RRs, etc.

6.2. Caching Security

The caching in DNS can make the propagation time for a change take the same amount of time as the time to live for the NAPTR records in the zone that is changed. The use of this in an environment where IP-addresses are for hire (for example, when using DHCP [9]) must therefore be done very carefully.

6.3. Call Routing Security

There are a number of countries (and other numbering environments) in which there are multiple providers of call routing and number/name-translation services. In these areas, any system that permits users,

or putative agents for users, to change routing or supplier information may provide incentives for changes that are actually unauthorized (and, in some cases, for denial of legitimate change requests). Such environments should be designed with adequate mechanisms for identification and authentication of those requesting changes and for authorization of those changes.

6.4. URI Resolution Security

A large amount of Security Issues have to do with the resolution process itself, and use of the URIs produced by the DDDS mechanism. Those have to be specified in the registration of the Enumservice used, as specified in Section 3.1.3.

7. Acknowledgements

Support and ideas leading to RFC 2916 have come from people at Ericsson, Bjorn Larsson and the group which implemented this scheme in their lab to see that it worked. Input has also arrived from ITU-T SG2, Working Party 1/2 (Numbering, Routing, Global Mobility and Enumservice Definition), the ENUM working group in the IETF, John Klensin and Leif Sunnegardh.

This update of RFC 2916 is created with specific input from: Randy Bush, David Conrad, Richard Hill, Jon Peterson, Jim Reid, Joakim Stralmark, Robert Walter and James Yu.

8. Changes since RFC 2916

Part from clarifications in the text in this document, the major changes are two:

The document uses an explicit DDDS algorithm, and not only NAPTR resource records in an "ad-hoc" mode. In reality this doesn’t imply any changes in deployed base of applications, as the algorithm used for ENUM resolution is exactly the same.

The format of the service field has changed. The old format was of the form "example+E2U", while the new format is "E2U+example". Reason for this change have to with the added subtypes in the enumservice, the ability to support more than one enumservice per NAPTR RR, and a general agreement in the IETF that the main selector between different NAPTR with the same owner (E2U in this case) should be first.

9. References

9.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

[3] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Four: The Uniform Resource Identifiers (URI) Resolution Application", RFC 3404, October 2002.

[4] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[5] ITU-T, "The International Public Telecommunication Number Plan", Recommendation E.164, May 1997.

[6] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS", RFC 3401, October 2002.

[7] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Two: The Algorithm", RFC 3402, October 2002.

9.2. Informative References

[8] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[9] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[10] Atkins, D. and R. Austein, "Threat Analysis Of The Domain Name System", Work in Progress, April 2004.

10. Authors' Addresses

Patrik Faltstrom
Cisco Systems Inc
Ledasa
273 71 Lovestad
Sweden

EMail: paf@cisco.com
URI: http://www.cisco.com

Michael Mealling
VeriSign
21345 Ridgetop Circle
Sterling, VA 20166
US

Email: michael@verisignlabs.com
URI: http://www.verisignlabs.com

11. Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

(12) **United States Patent**
Kelly

(10) **Patent No.: US 6,594,254 B1**
(45) **Date of Patent: *Jul. 15, 2003**

(54) **DOMAIN NAME SERVER ARCHITECTURE FOR TRANSLATING TELEPHONE NUMBER DOMAIN NAMES INTO NETWORK PROTOCOL ADDRESSES**

(75) Inventor: **Keith C. Kelly**, Deerfield Beach, FL (US)

(73) Assignee: **Netspeak Corporation**, Newark, NJ (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/911,519**

(22) Filed: **Aug. 14, 1997**

Related U.S. Application Data

(60) Provisional application No. 60/023,891, filed on Aug. 16, 1996.

(51) **Int. Cl.**[7] **H04L 12/66**

(52) **U.S. Cl.** **370/352**; 370/389; 370/401

(58) **Field of Search** 370/352, 354, 370/356, 389, 392, 475, 237, 401, 230; 379/88.17, 201, 210, 219, 220, 224; 709/200, 201, 203, 218, 219, 227, 202, 230; 455/415, 461; 340/5.8

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,608,786	A	* 3/1997	Gordon	370/352
5,742,668	A	* 4/1998	Pepe et al.	455/415
5,742,762	A	* 4/1998	Scholl et al.	709/200
5,742,905	A	* 4/1998	Pepe et al.	455/461
5,799,063	A	* 8/1998	Krane	379/88.17
6,014,379	A	* 1/2000	White et al.	370/389
6,021,126	A	* 2/2000	White et al.	370/352
6,069,890	A	* 5/2000	White et al.	370/352
6,122,255	A	* 9/2000	Bartholomew et al.	370/237
6,154,445	A	* 11/2000	Farris et al.	370/237
6,226,678	B1	* 5/2001	Mattaway et al.	709/230
6,300,863	B1	* 10/2001	Cotichini et al.	340/5.8
6,347,085	B2	* 2/2002	Kelly	370/352

OTHER PUBLICATIONS

C. Yang, "INETPhone: Telephone Services and Servers on Internet", Network Working Group RFC: 1789, pp. 1–6, Apr. 1995.*

" *The History of TPC.INT,* " *http://www.tpc.int/faq/history.html.*

C. Malamud & M. Rose, " *Principles of Operation for the TPC.INT Subdomain: Remote Printing –Technical Procedures,*" Network Working Group, Request for Comments: 1528 (Oct. 1993).

C. Malamud & M. Rose, " *Principles of Operation for the TPC.INT Subdomain*: Remote Printing –Administrative Policies," Network Working Group, Request for Comments: 1529 (Oct. 1993).

C. Malamud & M. Rose, " *Principles of Operation for the TPC.INT Subdomain: General Principles and Policy,*" Network Working Group, Request for Comments: 1530 (Oct. 1993).

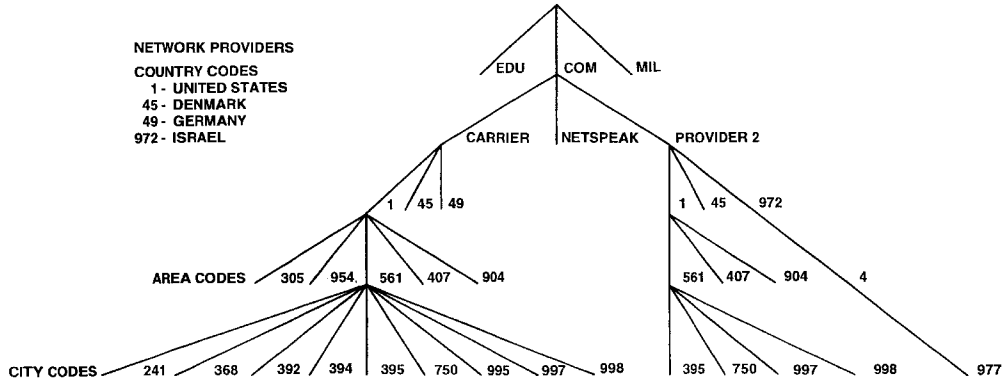
* cited by examiner

Primary Examiner—Hassan Kizou
Assistant Examiner—John Pezzlo
(74) *Attorney, Agent, or Firm*—Kenyon & Kenyon

(57) **ABSTRACT**

A method and apparatus for translating a domain name representing a telephone number into a network protocol address includes a domain name server architecture containing logic responsive to a telephone number domain name, the telephone number domain name representing the country code, area code, exchange, or subscriber number of a subscriber apparatus telephone number. The logic resolves the telephone number domain name into a network protocol address usable in ultimately initiating a communication with the subscriber apparatus on a circuit-switched network. In one embodiment, a hierarchical tree of domain names and subdomain names representing the country codes, area codes and exchange codes of telephone numbers is constructed to assist in the process of resolving domain names to network protocol addresses.

23 Claims, 7 Drawing Sheets



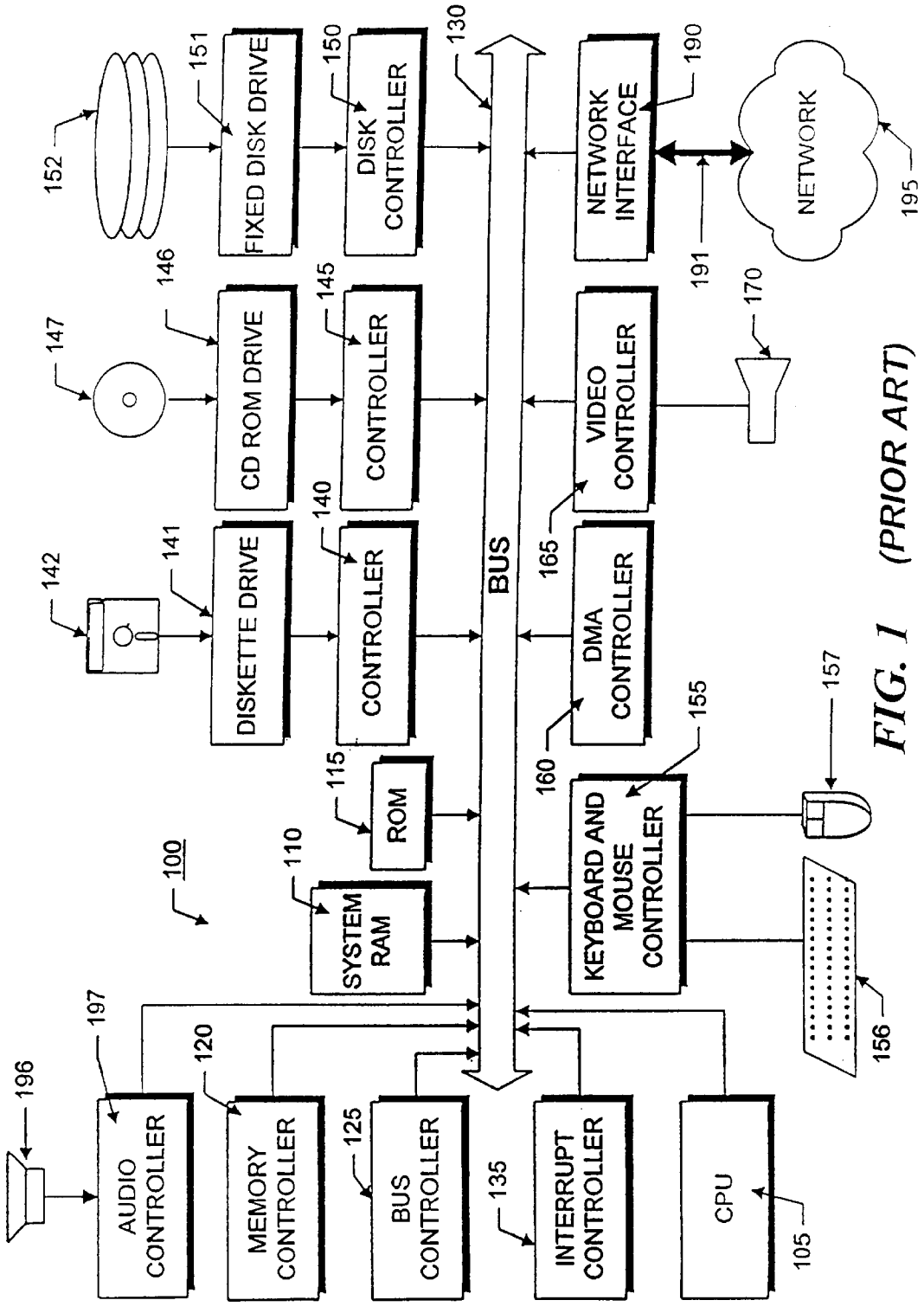


FIG. 1 (PRIOR ART)

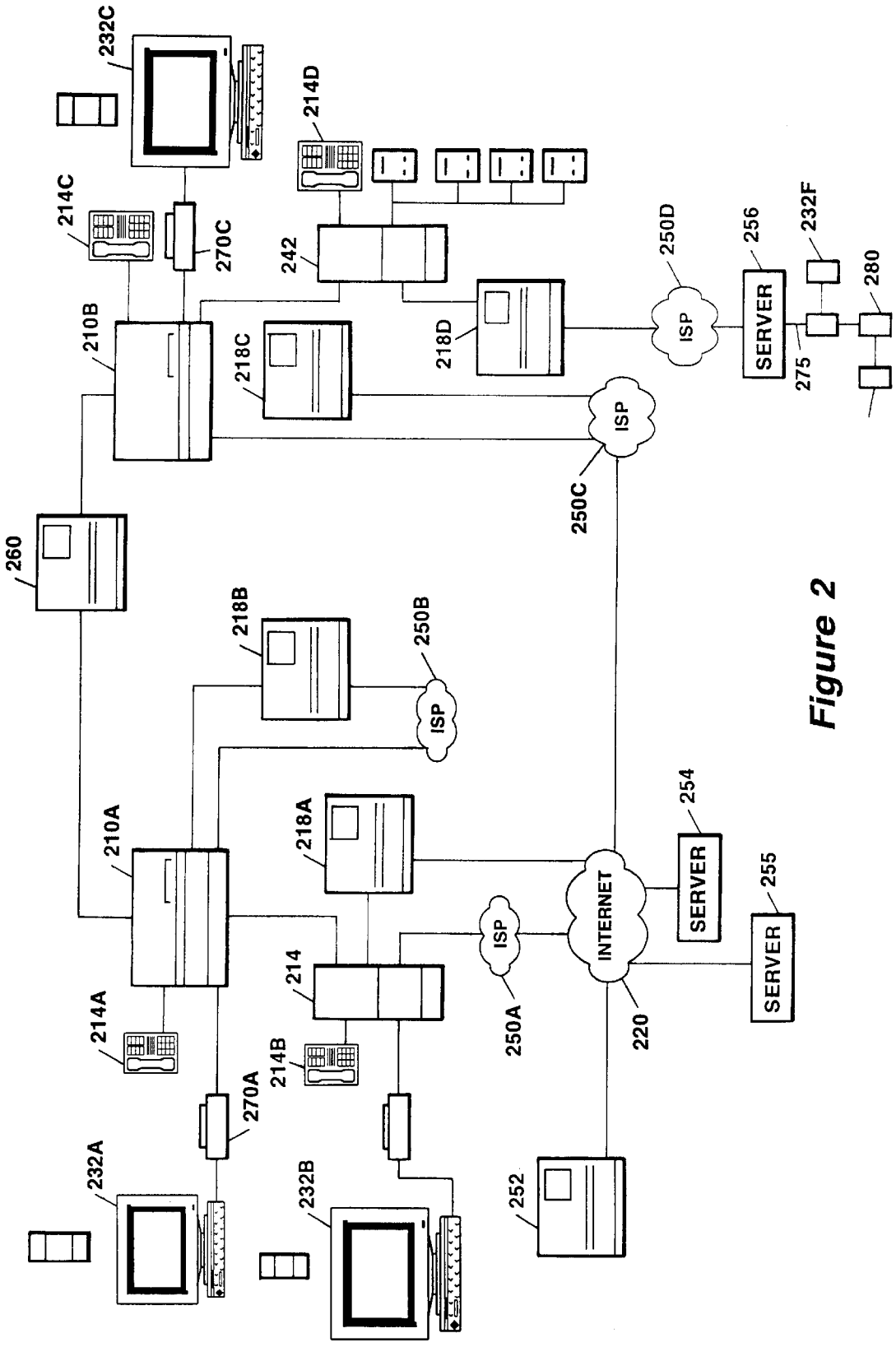

232C
214D
214C
270C
242
250D
256
232F
280
210B
218C
218D
ISP
SERVER
275
260
250C
218B
250B
ISP
210A
218A
254
INTERNET
SERVER
255
214
ISP
214A
214B
250A
220
270A
232A
232B
252

Figure 2

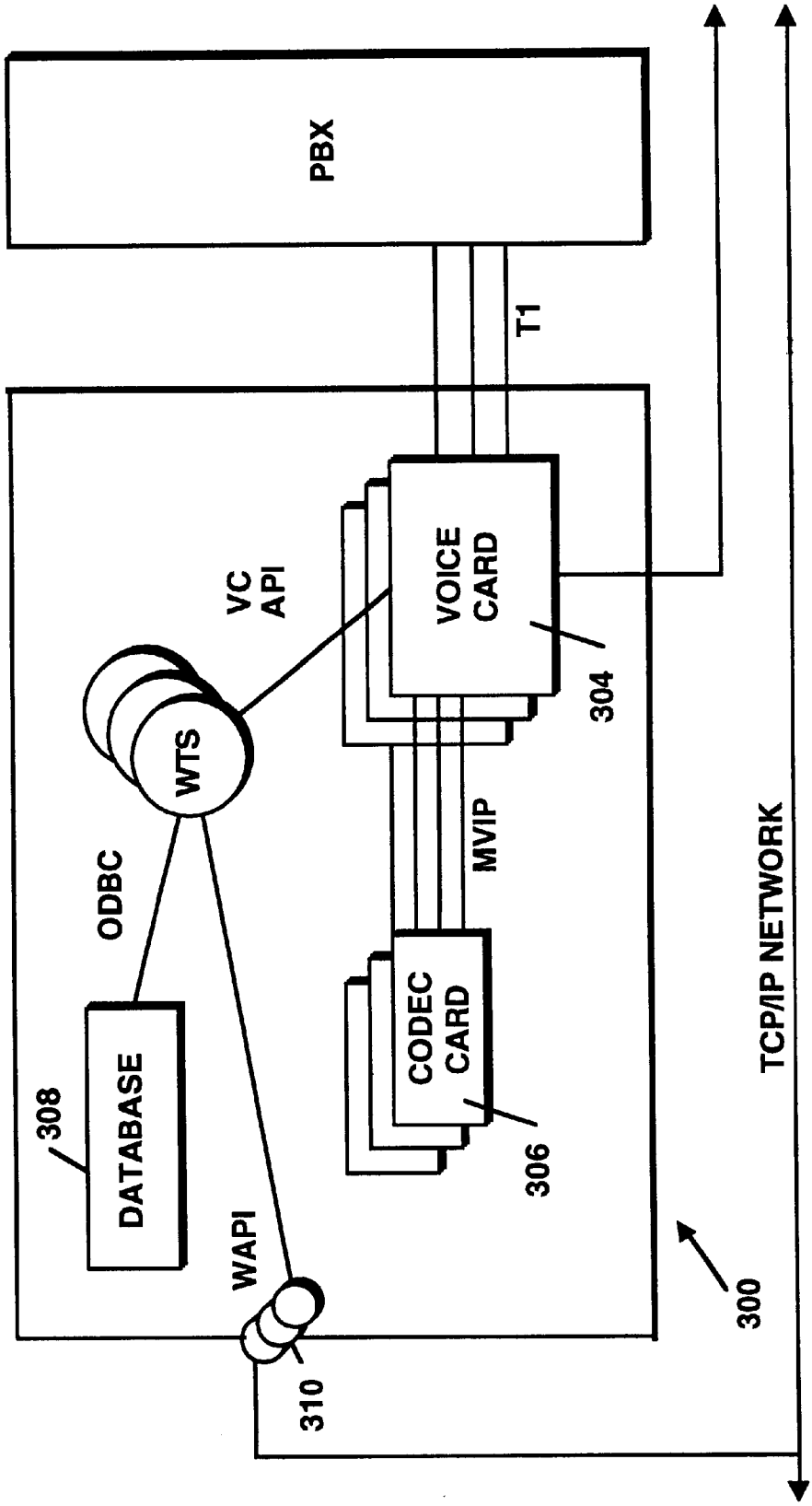

PBX
T1
VOICE CARD
304
VC API
WTS
ODBC
MVIP
CODEC CARD
306
DATABASE
308
WAPI
310
300
TCP/IP NETWORK

Figure 3

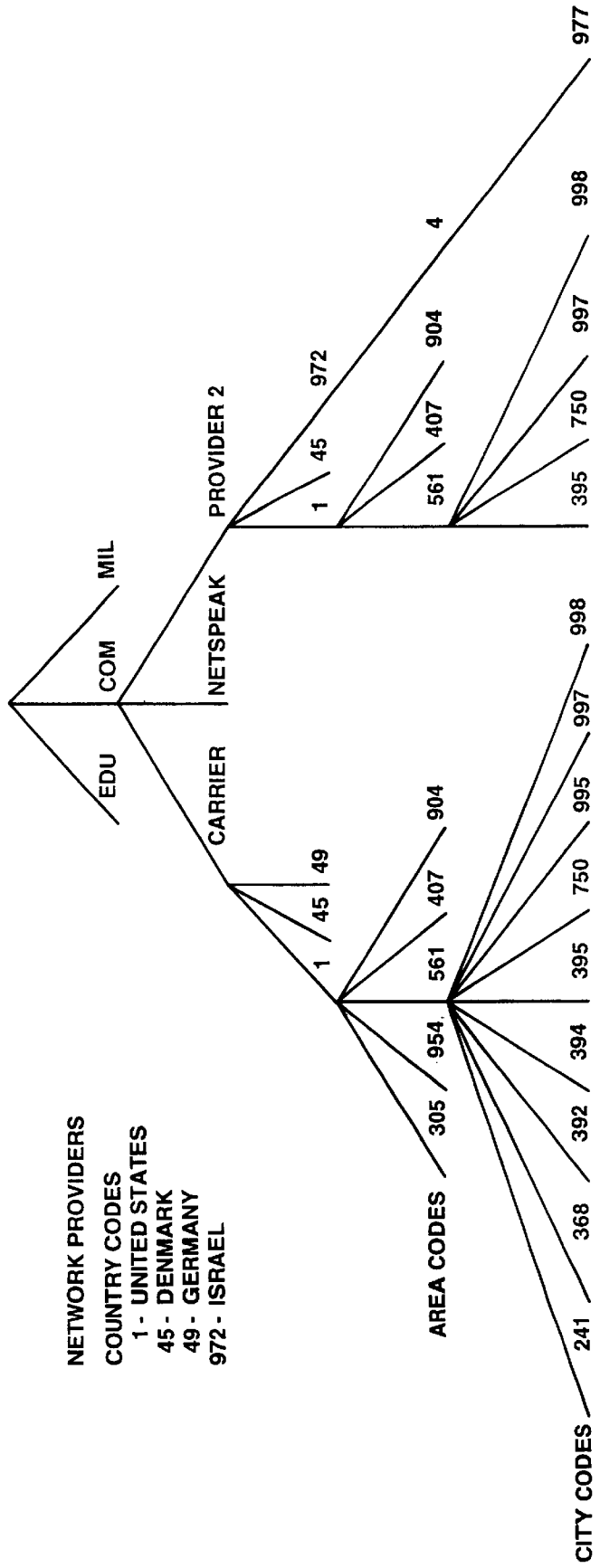

NETWORK PROVIDERS
COUNTRY CODES
1 - UNITED STATES
45 - DENMARK
49 - GERMANY
972 - ISRAEL
EDU
COM
MIL
CARRIER
NETSPEAK
PROVIDER 2
1
45
49
972
4
AREA CODES
305
954
561
407
904
CITY CODES
241
368
392
394
395
750
995
997
998
977

Figure 4

SUBDOMAIN	NETWORK ADDRESS
1 45 49	XXX. XXX. XXX. XXX

Figure 5A

SUBDOMAIN	NETWORK ADDRESS
305 954 516 407 904	XXX. XXX. XXX. XXX

Figure 5B

SUBDOMAIN	NETWORK ADDRESS
241 368 392 394 395 750 995 997 998	XXX. XXX. XXX. XXX

Figure 5C

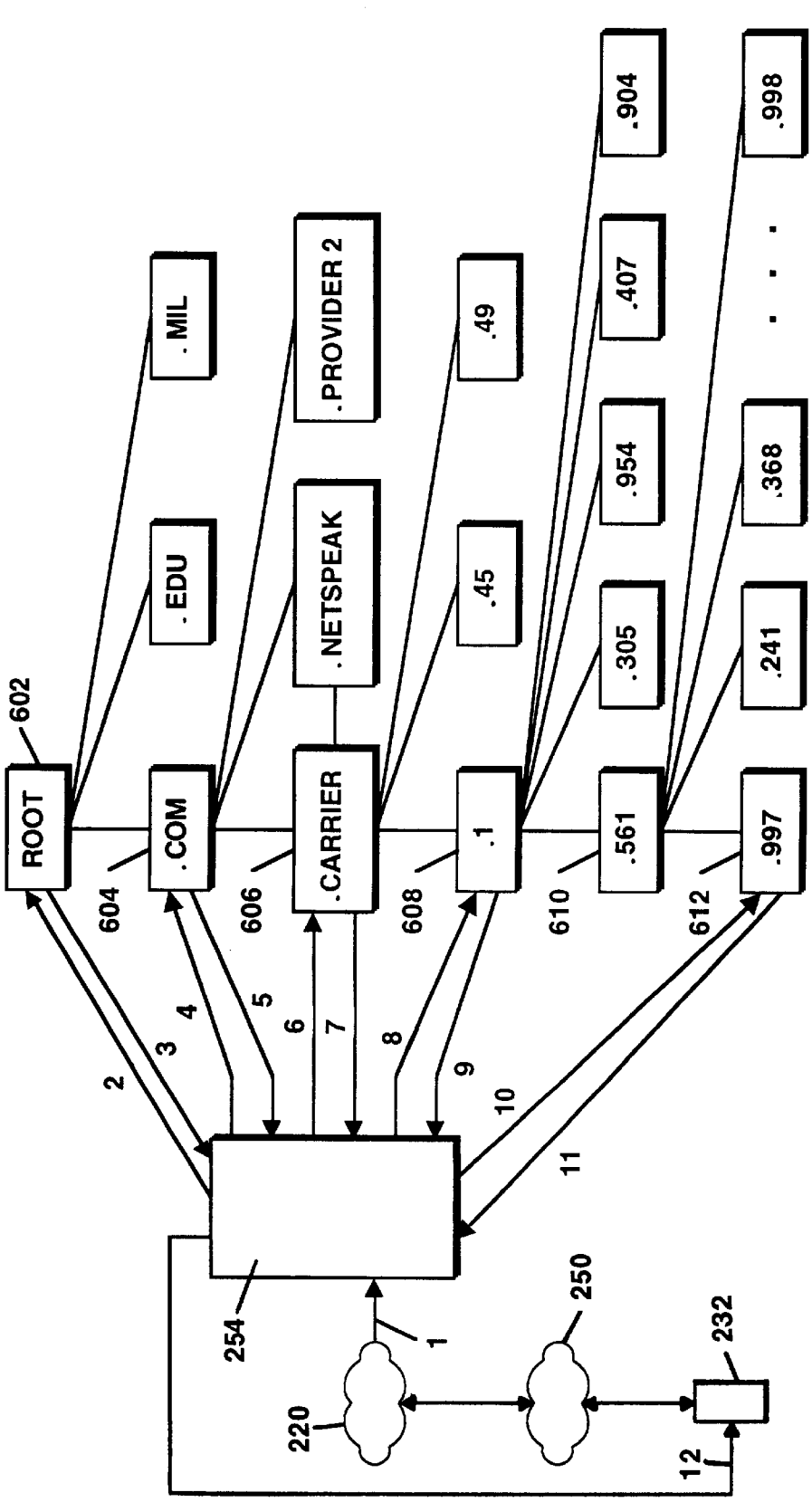

602
ROOT
.COM
.EDU
.MIL
.CARRIER
.NETSPEAK
.PROVIDER 2
.1
.45
.49
.561
.305
.954
.407
.904
.997
.241
.368
.998
604
606
608
610
612
2
3
4
5
6
7
8
9
10
11
254
1
220
250
232
12

Figure 6

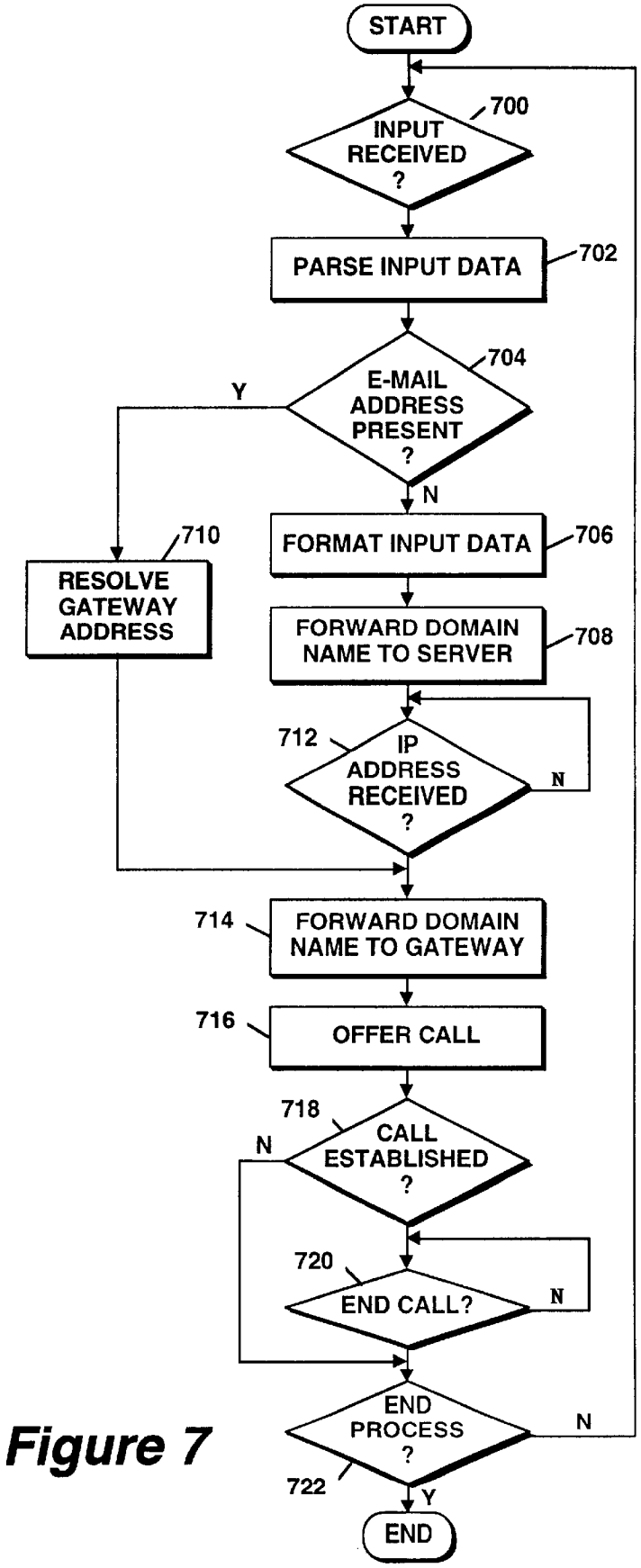

START
700
INPUT RECEIVED ?
PARSE INPUT DATA
702
704
E-MAIL ADDRESS PRESENT ?
Y
N
710
RESOLVE GATEWAY ADDRESS
FORMAT INPUT DATA
706
FORWARD DOMAIN NAME TO SERVER
708
712
IP ADDRESS RECEIVED ?
N
714
FORWARD DOMAIN NAME TO GATEWAY
716
OFFER CALL
718
CALL ESTABLISHED ?
N
720
END CALL?
N
END PROCESS ?
N
722
Y
END

Figure 7

DOMAIN NAME SERVER ARCHITECTURE FOR TRANSLATING TELEPHONE NUMBER DOMAIN NAMES INTO NETWORK PROTOCOL ADDRESSES

RELATED APPLICATIONS

This application claims priority to U.S. Provisional Patent Application No. 60/023,891 entitled Apparatus For Placing Internet/Intranet Calls by Keith C. Kelly, filed Aug. 16, 1996.

This application is the one of two U.S. patent applications filed on an even date herewith and commonly assigned, including Ser. No. 08/911,133, by Keith C. Kelly, entitled Method and Apparatus for Establishing Communications Between Packet-Switched and Circuit-Switched Networks, the subject matter of which, by this reference, is incorporated herein.

In addition, the subject matters of the following related applications are incorporated herein by reference:

U.S. patent application Ser. No. 08/533,115 entitled Point-to-Point Internet Protocol, by Glenn W. Hutton, filed Sep. 25, 1995;

U.S. patent application Ser. No. 08/719,894, entitled Directory Server For Providing Dynamically Assigned Network Protocol Addresses, by Mattaway filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/721,316, entitled Graphic User Interface For Internet Telephony Application, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/719,891, entitled Method And Apparatus For Distribution And Presentation Of Multimedia Data Over A Computer Network, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/719,554, entitled Point-to-point Computer Network Communication Utility Utilizing Dynamically Assigned Network Protocol Addresses, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/719,640, entitled Method And Apparatus For Dynamically Defining Data Communication Utilities, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/719,898, entitled Method And Apparatus For Providing Caller Identification Based Out-going Messages In A Computer Telephony Environment, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/718,911, entitled Method And Apparatus For Providing Caller Identification Based Call Blocking In A Computer Telephony Environment, by Mattaway et al., filed Sep. 25, 1996;

U.S. patent application Ser. No. 08/719,639, entitled Method And Apparatus For Providing Caller Identification Responses In A Computer Telephony Environment, by Mattaway et al., filed Sep. 25, 1996; and

U.S. patent application Ser. No. 08/832,746, entitled Virtual Circuit Switching Architecture, by Mattaway et al., filed Apr. 4, 1997;

FIELD OF THE INVENTION

The invention relates, generally, to data processing systems and telecommunication systems, and, more specifically, to a technique for enabling communication connections between circuit-switched communication networks and packet-switched data processing networks.

BACKGROUND OF THE INVENTION

Two fundamentally different switching technologies exist that enable digital communications. The first type, circuit-switched networks, operate by establishing a dedicated connection or circuit between two points, similar to public switched telephone networks(PSTN). A telephone call causes a circuit to be established from the originating phone through the local switching office across trunk lines, to a remote switching office and finally to the intended destination telephone. While such circuit is in place, the call is guaranteed a data path for digitized or analog voice signals regardless of other network activity. The second type packet-switched networks, typically connect computers and establish an asynchronous "virtual" channel between two points. In a packet-switched network, data, such as a voice signal, is divided into small pieces called packets which are then multiplexed onto high capacity connections for transmission. Network hardware delivers packets to specific destinations where the packets are reassembled into the original data set. With packet-switched networks, multiple communications among different computers can proceed concurrently with the network connections shared by different pairs of computers concurrently communicating. Packet-switched networks are, however, sensitive to network capacity. If the network becomes overloaded, there is no guarantee that data will be timely delivered. Despite this drawback, packet-switched networks have become quite popular, particularly as part of the Internet and Intranets, due to their cost effectiveness and performance.

In a packet-switched data network one or more common network protocols hide the technological differences between individual portions of the network, making interconnection between portions of the network independent of the underlying hardware and/or software. A popular network protocol, the Transmission Control Protocol/Internet Protocol (TCP/IP) is utilized by the Internet and Intranets>Intranets are private networks such as Local Area Networks (LANs) and Wide Area Networks (WAN). The TCP/IP protocol utilizes universal addressing as well as a software protocol to map the universal addresses into low level machine addresses. For purposes of this discussion, networks which adhere to the TCP/IP protocol will be referred to hereinafter "IP-based" or as utilizing "IP addresses" or "Internet Protocol address".

It is desirable for communications originating from an IP-based network to terminate at equipment in a PSTN network, and vice versa, or for calls which originate and terminate on a PSTN network to utilize a packet-switched data network as an interim communication medium. Problems arise, however, when a user on an IP-based or other packet switched data network tries to establish a communication link beyond the perimeter of the network, due to the disparity in addressing techniques among other differences used by the two types of networks.

The exchange/subscriber addressing scheme utilized by public switched telephone networks is closely related to the actual physical architecture of the network and therefore to the geographic location of terminating apparatus, i.e. telephone withthin the network. For example, in the United States, a telephone number may be partitioned into a three-digit area code, a three-digit exchange, and a four-digit subscriber number within the exchange. PSTN carriers currently have large, well-established networks with multitudes of subscribers utilizing such naming systems and are set up to interact with other PSTN carrier networks and to account for the variances in local geographic dialing patterns.

Conversely, packet-switched data networks adhere to a network protocol such as the TCP/IP protocol utilizes a hierarchical naming system which is neither based on the underlying hardware of the network nor the geographic

locus of the various hardware components. Instead, the TCP/IP protocol partitions computers along lines of authority irrespective of physical location. In TCP/IP, hierarchical machine names are assigned according to structures of organization for parts of name space, not necessarily according to the structure of a physical network interconnection. The TCP/IP protocol implements a naming hierarchy called the Domain Name System (DNS). The Domain Name System utilizes a hierarchical naming scheme referred to as domain names. Domain names consist of a sequence of subdomain names, separated by a delimiter character, i.e. "." The subdomain names of a domain name are sometimes referred to as labels. For example, the domain name "www.netspeak.com" contains three labels: "www", "netspeak", and "com". Any suffix of a label in a domain name is called a domain. In the above example, the top-level domain is "com." The domain name system is well documented in various public specifications and will not be described hereinafter for the sake of brevity.

The reader will appreciate that due to the naming schemes utilized by PSTN networks and TCP/IP based networks, such as the Internet establishment of direct connections from the Internet to public switched telephone network subscribers and vice versa, is impractical.

In light of the above, a need currently exists for a mechanism which enables translation of a conventional telephone number from a client task on an IP-based network into a network protocol address representing a gateway capable of contacting the subscriber apparatus associated with the telephone number.

A need further exists for a mechanism which accounts for the dilemma of geographic dialing patterns and the nongeographic nature of IP-based networks such as the Internet, versus the hardwired, geographic nature of public switched telephone networks.

A need further exists for a mechanism which facilitates communication between packet-switched networks and circuit-switched networks and which accommodate the existing infrastructure of circuit-switched networks, including inoperability among carriers, subscriber volumes and billing logistics.

SUMMARY OF THE INVENTION

The invention describes a method and apparatus which enables traditional telephone numbers formatted as domain names to be resolved into network protocol addresses, either by a single domain name server apparatus or multiple domain name server apparatus.

According to a first aspect of the present invention, a method for resolving data representing a telephone number comprises the steps of receiving a telephone number domain name identifying a telephone from a source, resolving the telephone number domain name into a network protocol address, and supplying a network protocol address to the source. In one embodiment, a portion of the telephone number domain name represents the country code, area code, exchange, or data segments of a telephone number. In an alternative embodiment, the method entails resolving country code, area code or exchange subdomain names into the network protocol addresses of their respective domain name servers.

In accordance with a second aspect of the present invention, a domain name server apparatus comprises a processor for manipulating data, a memory couple to the process for storing data, connection logic configured to couple the name server to the computer network, at least one domain name stored in the memory, the domain name having associated therewith a network protocol address and having a portion thereof representing the country code, area code, exchange, subscriber number or carrier of a telephone number and logic for generating the network protocol address associated with the domain name. In one embodiment, a plurality of domain name labels are stored in memory, each label having associated therewith a network protocol address, each domain name label representing at least one of the carrier, country code, area code, exchange or subscriber number of a telephone number.

In accordance with a third aspect of the invention, a computer program product for use with a computer system comprises a computer usable medium having program code embodied in the medium for enabling translation of data representing a telephone number of subscriber apparatus into a network protocol address. The program code comprises code for receiving a telephone number domain name from a source, program code responsive to a portion of the telephone number domain name for generating a network protocol address, and program code for forwarding the network protocol address to the source. In one embodiment, the program code further comprises code for storing in the computer system at least one domain name having associated therewith a network protocol address, the domain name representing at least one of the country code, area code and exchange data of a telephone number. In an alternative embodiment, the program code comprises code for storing in the computer system a plurality of domain names, each having associated therewith a network protocol address, each of the domain names representing the carrier, country code, area code, exchange, or subscriber number data of a telephone number.

In accordance with a fourth aspect of the invention, a system for facilitating communication between client tasks executing on a packet-switched data network and subscriber apparatus on a circuit-switched communication network comprises a domain name server operatively coupled to the packet-switched network. The domain name server comprises resolution logic responsive to a telephone number domain name for generating a network protocol address associated with the domain name. The system further comprises a gateway server addressable by a network protocol address. The gateway comprises logic configured to initiate a communication link with subscriber apparatus on a circuit-switched communication network in response to receipt of a telephone number domain name.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other features, objects, and advantages of the invention will be better understood by referring to the following description in conjunction with the accompanying drawing in which:

FIG. **1** is a block diagram of a computer systems suitable for use with the present invention;

FIG. **2** is a conceptual illustration of a communications network environment in which the present invention may be utilized;

FIG. **3** is a block diagram of a gateway system in accordance with the present invention;

FIG. **4** illustrates conceptually a hierarchical domain naming structure utilized with the server of the present invention;

FIGS. **5**A–C illustrate conceptually the data structures used to implement the present invention;

FIG. **6** illustrates conceptually the inventive steps utilized to resolve a telephone number to a network address of a gateway in accordance with the present invention, and

FIG. **7** is a flow chart illustrating the process steps of the method in accordance with the present invention.

DETAILED DESCRIPTION

FIG. **1** illustrates the system architecture for a computer system **100**, such as an IBM PS/2® computer on which the invention can be implemented. The exemplary computer system of FIG. **1** is for descriptive purposes only. Although the description below may refer to terms commonly used in describing particular computer systems, such as an IBM PS/2 computer, the description and concepts equally apply to other systems, including systems having architectures dissimilar to FIG. **1**.

The computer system **100** includes a central processing unit (CPU) **105**, which may include a conventional microprocessor, a random access memory (RAM) **110** for temporary storage of information, and a read only memory (ROM) **115** for permanent storage of information. A memory controller **120** is provided for controlling system RAM **110**. A bus controller **125** is provided for controlling bus **130**, and an interrupt controller **135** is used for receiving and processing various interrupt signals from the other system components. Mass storage may be provided by diskette **142**, CD ROM **147** or hard drive **152**. Data and software may be exchanged with computer system **100** via removable media such as diskette **142** and CD ROM **147**. Diskette **142** is insertable into diskette drive **141** which is, in turn, connected to bus **130** by a controller **140**. Similarly, CD ROM **147** is insertable into CD ROM drive **146** which is connected to bus **130** by controller **145**. Hard disk **152** is part of a fixed disk drive **151** which is connected to bus **130** by controller **150**.

User input to computer system **100** may be provided by a number of devices. For example, a keyboard **156** and mouse **157** are connected to bus **130** by controller **155**. An audio transducer **196**, which may act as both a microphone and a speaker, is connected to bus **130** by audio controller **197**, as illustrated. It will be obvious to those reasonably skilled in the art that other input devices such as a pen and/or tablet and a microphone for voice input may be connected to computer system **100** through bus **130** and an appropriate controller/software. DMA controller **160** is provided for performing direct memory access to system RAM **110**. A visual display is generated by video controller **165** which controls video display **170**. Computer system **100** also includes a communications adaptor **190** which allows the system to be interconnected to a local area network (LAN) or a wide area network (WAN), schematically illustrated by bus **191** and network **195**.

Computer system **100** is generally controlled and coordinated by operating system software, such the OS/2® operating system, available from International Business Machines Corporation, Armonk, N.Y. The operating system controls allocation of system resources and performs tasks such as process scheduling, memory management, and networking and I/O services, among other things.

Telecommunication Environment

FIG. **2** illustrates a telecommunications environment in which the invention may be practiced such environment being for exemplary purposes only and not to be considered limiting. Network **200** of FIG. **2** illustrates a hybrid telecommunication environment including both a traditional public switched telephone network as well as Internet and Intranet networks and apparatus bridging between the two. The elements illustrated in FIG. **2** are to facilitate an understanding of the invention. Not every element illustrated in FIG. **2** or described herein is necessary for the implementation or the operation of the invention.

A pair of PSTN central offices **210**A–B serve to operatively couple various terminating apparatus through either a circuit switched network or a packet switched network. Specifically, central offices **210**A–B are interconnected by a toll network **260**. Toll network **260** may be implemented as a traditional PSTN network including all of the physical elements including routers, trunk lines, fiber optic cables, etc. Connected to central office **210**A is a traditional telephone terminating apparatus **214** and an Internet telephone **232**A. Terminating apparatus **214** may be implemented with either a digital or analog telephone or any other apparatus capable of receiving a call such as modems, facsimile machines, etc., such apparatus being referred to collectively hereinafter as a terminating apparatus, whether the network actually terminates. Further, the PSTN network may be implemented as either an integrated services digital network (ISDN) or a plain old telephone service (POTS) network. The Internet telephony is conceptually illustrated as a telephone icon symbolizing the Internet telephone client application executing on a personal computer and interconnected to central office **210**A via a modem **270**A. Similarly, telephone **214**C is connected to central office **210**D and WebPhone **232**C is connected to central office **210**B via modem **270**C. Central offices **210**A–B are, in turn, operatively coupled to Internet **220** by ISP **250**B and **250**C, respectively. In addition, central office **210**A is coupled to ISP**250**B by gateway **218**B. Similarly, central office **210**B is connected to ISP **250**C by gateway **218**C. In addition, a telephone **214**B and Internet telephone **232**B, similar to telephone **214**A and Internet telephone **232**A, respectively, are interconnected to Internet **220** via PBX **212**, gateway **218**A and ISP **258**A. In addition, global server **252** is coupled to the Internet **220** are a domain name system server **254** and **255**. Global server **252** may be implemented as described in U.S. patent application Ser. No. 08/719,894, entitled Directory Server for Providing Dynamically Assigned Network Protocol Addresses, previously referenced and incorporated herein. A global server suitable for use as Global Server **252** is commercially available from NetSpeak Corporation in the form of a collection of intelligent software modules including connection server Part No. CSR1, information server, Model ISR1, and database server, Model DBSR1. Name servers **254** and **255** are described as set forth hereinafter. Finally, Internet Service Providers (ISPs) **250**A–D may comprise any number of currently commercially available Internet service providers such as America On Line, the IBM Global Network, etc. An Intranet implemented as LAN **275** is coupled to Internet **220** via ISP **250**D and server **256**. Server **256** may have the architecture as illustrated in FIG. **1** and functions as a proxy server for LAN **275** to which WebPhone **232**E is connected via a LAN-based TCP/IP network connector **280**. A plurality of Internet telephone **232**F and **232**E are coupled to LAN **275** via LAN connectors **280**. The gateways and Internet telephony client applications may be implemented as set forth in greater detail hereinafter.

WebPhone Client

Internet telephone **232** may be implemented as described in the previously referenced U.S. patent applications incorporated herein by reference. An Internet telephony application suitable for use with the present invention is the WebPhone 1.0, 2.0 or 3.0, client software application commercially available from NetSpeak Corporation, Boca Raton, Fla., referred to hereafter as the WebPhone client. For the remainder of this description, the Internet telephone will be referred to as the WebPhone client. It will be obvious to

those reasonably skilled in the arts that other Internet telephone applications implementing similar functionality may be substituted for the WebPhone without affecting the inventive concepts contained herein. The WebPhone comprises a collection of intelligent software modules which perform a broad range of Internet telephony functions. For the purpose of this disclosure, a "Virtual" WebPhone client refers to the same functionality embodied in the WebPhone client application without a graphic user interface. Such virtual WebPhone client can be embedded into a gateway, automatic call distribution, server, or other apparatus which do not require extensive visual input/output from a user and may interact with any other WebPhone clients or servers adhering to the WebPhone protocol. For the purpose of this disclosure, WebPhone **232** or any of the virtual WebPhones implemented in other apparatus, may be considered WebPhone client applications, "WebPhone Clients", as opposed to other apparatus such as the connection/information server, which adheres to the WebPhone Protocol.

The WebPhone software applications may run on the computer system described with reference to FIG. **1**, or a similar architecture whether implemented as a personal computer or dedicated server. In such an environment, the sound card **197** accompanying the computer system **100** of FIG. **1**, may be an MCI compliant sound card while communication controller **190** may be implemented through either an analog modem **270** or a LAN-based TCP/IP network connector **280** to enable Internet/intranet connectivity.

The WebPhones, as well as any other apparatus having a virtual WebPhone embodied therein, each have their own unique E-mail address and adhere to the WebPhone Protocol and packet definitions, as extensively described in the previously referenced U.S. patent applications. For the reader's benefit, short summary of a portion of the WebPhone Protocol is set forth to illustrate the interaction of WebPhone clients with each other and the connection/information server when establishing a communication connection.

Each WebPhone client, may serve either as a calling party or a caller party, i.e. the party being called. The calling party transmits an on-line request packet to a connection/information server upon connection to an IP-based network, e.g. the Internet or an Intranet. The on-line request packet contains configuration and settings information, a unique E-mail address and a fixed or dynamically assigned IP address for the WebPhone client. The callee party, also a utilizing a WebPhone client, transmits a similar on-line request packet containing its respective configuration and setting information, E-mail address and IP address to the same or a different connection server upon connection to an IP-based network. The calling party originates a call by locating the callee party in a directory associated with either its own WebPhone client or the connection/information server to which it is connected. The callee party may be identified by alias, E-mail address or key word search criteria. Once the E-mail address of the calling party is identified, the calling party's WebPhone forwards a request packet to the connection/information server, the request packet containing the callee party's E-mail address. The connection/information server uses the E-mail address in the received request packet to locate the last known IP address assigned to the callee party. The connection/information server then transmits to the calling party an information packet containing the IP address of the callee party. Upon receipt of the located IP address from the connection server, the calling party's WebPhone initiates a direct point-to-point communication link with the callee party by sending a call packet directly to the IP address of the callee party. The callee party either accepts or rejects the call with appropriate response packets. If the call is accepted, a communication session is established directly between the caller and the callee, without intervention of the connection/information server. The above scenario describes establishment of a communication link which originates and terminates with clients on an IP-based network.

Gateway Architecture

To facilitate interaction with WebPhone clients, a virtual WebPhone is implemented in the gateway **218**, either executable in RAM memory or embedded in ROM memory associated with such apparatus. The gateway **218** comprises a virtual WebPhone which acts as a proxy device and voice processing hardware that bridges from an IP-based network to a PSTN network. The gateway **218** may be implemented with either a microprocessor based architecture or with dedicated digital signal processing logic and embedded software. A gateway suitable for use as gateway **218** with the present invention is either NetSpeak Model Nos. WGX-MD/24, a 24-port digital T-1 IP telephony gateway, or WGX-M/16, a 16-port analog IP telephony gateway, both commercially available from NetSpeak Corporation, Boca Raton, Fla. Gateway **218** is described in greater detail with reference to FIG. **3** hereinafter.

FIG. **3** illustrates the system architecture for gateway **300**, which may be implemented as gateway **218**. Gateway **300** may be implemented using a computer architecture similar to computer system **100** described with reference to FIG. **1**, such elements having been described in detail and not shown in FIG. **3**. In addition, gateway **300** comprises a server **302**, one or more voice cards **304**, one or more compression/decompression (codec) cards **306**, a database **308** and a network interface **310**. Server **302** may embody one or both of the architecture function of the connection server and information servers similar to those embodied in global server **252** and as described in detail in the previously referenced copending patent application. The server **300** is a multiple-instance server implementing the WebPhone protocol that functions as a connection server to provide connectivity to global server **252** and as a virtual WebPhone on behalf of a telephone that is connected to either a public branch exchange, such as PBX **212**, or a central office, such as central office **210**A. Each instance of the server **300** interfaces with one channel of voice card **304** via the application program interface on the voice card, to provide a conversation path between a conventional telephone and a WebPhone or database **308**.

The voice card **304** provides a T-1 or analog connection to the PBX or central office or analog telephone lines which have a conventional telephony interface, for example, DID, ENM. The voice card application program interface enable the instance of server **300** to emulate a conventional telephone on a PBX or central office of a PSTN carrier. Multichannel audio compression and decompression is accessed by server **300** via application program interfaces on the respective sound cards and is processed by the appropriate audio codec. Any number of commercially available voice cards may be used to implement voice card **304**. Similarly, any number of commercially available audio codecs providing adequate audio quality may be utilized. Each instance of server **300** interfaces with the TCP/IP network through a series of ports which adhere to the WebPhone protocol. Gateway **300** interfaces with the T1 line of the PSTN network through the interfaces contained within voice card(s) **304**.

Gateway **300** may optionally include a database **308** which may be implemented using any number of commer-

cially available database server products including Microsoft SQL Server 6.X. Database **308** interacts with each instance of server **302** typically via ODBC format.

In the preferred embodiment, gateway **300** is a combined software hardware implementation which runs on an operating system such as Windows NT.

Telephone Domain Name Server Architecture

FIG. **4** illustrates conceptually a small part of the Internet domain hierarchy tree in accordance with the present invention. At the top of the tree is the unnamed root.

Underneath the root are top level subdomains including .edu, .com, .mil, as illustrated, as well as .gov, .net, .arpa, .int, and various country code abbreviations, (not illustrated). Underneath the ".com" domain are the subdomains "carrier", "NetSpeak" and "provider **2**". Underneath the "carrier" and "provider **2**" domains exists a tree hierarchy of each their respective subdomains representing the country codes, area codes and exchange codes of traditional telephone numbers. For example, the country codes for the United States, Denmark, and Germany exist under the domain name node "carrier.com." Under the domain name "1" for the United States country code, exists a number of subdomain name representing selected area codes in the United States. Similarly, underneath the domain name for area code "561", exists a plurality of subdomains for the exchanges contained within area code 561, etc. The parent domain name for provider**2**.com is arranged similarly with different respective values. In this manner, the geographical dialing patterns of traditional PSTN telephone numbers have been arranged into an organization hierarchy similar to the domain name system hierarchy used by the domain name system and which further identify the IP address of the gateway to dial the terminating equipment.

Any of the domain name servers, including primary domain name server **254**, as well as the domain name servers used to resolve a telephone domain name into network protocol addresses may be implemented utilizing the computer architecture described with reference to FIG. **1** herein. The actual implementation of the data structures utilized to resolve a domain name or subdomain name to a network protocol address may be implemented as illustrated in FIG. **5**A–C. FIG. **5**A illustrates a look-up table-type data structure which may be retained in system memory and utilized by a carrier domain name server to resolve country code subdomain names into the network protocol addresses of a country code domain name server. FIG. **5**B illustrates a look-up table type data structure which may be retained in system memory and utilized by a country code domain name server to resolve area code subdomain names into the network protocol addresses of area code domain name servers. FIG. **5**C illustrates a look-up table type data structure which may be retained in system memory utilized by an area code domain name server which may be utilized to resolve exchange subdomains into the network protocol addresses of either the respective corresponding gateway or a subscriber domain name server. In FIG. **5**A–B, the network protocol addresses are illustrated as having a format similar to that utilized by the Internet protocol. It will be obvious to those reasonably skilled in the art that as the Internet protocol evolves and/or new network protocols are adopted, that the actual format and implementation of the network protocol address may be modified accordingly without effecting the telephone domain name resolution process. Further, the actual format of the subdomain identifier into each of the look-up tables in FIG. **5**A–C may be implemented with any number of reference formats.

The invention contemplates that the data structures illustrated in FIGS. **5**A–C, respectively may be combined and implemented on a single server. In a single server solution, the server may be the carrier domain name server and can resolve all references to country code, area code, exchange code and possibly subscriber subdomains directly to the appropriate network protocol address of the relevant gateway internally within the carrier domain name server itself or with an appropriate messaging protocol without having to return to either the WebPhone client application or the primary name server at each level during the domain name resolution process. Such a single server embodiment lends itself more readily to recursive resolution of a telephone number domain name. In such an embodiment, the primary domain name server need only contact the root domain name server for the domain name system and possibly one or two additional domain name servers before reaching the carrier domain name server which performs the majority of the resolution translation activities.

Alternatively, the data structures for the domain name servers illustrated in FIGS. **5**A–C each may be implemented with separate respective domain name server platforms, such embodiment lending itself more readily to a distributed domain name server environment.

For the reader's benefit, some background information is provided to illustrate how name queries are resolved within the Domain Name System using the name resolver protocol and name servers. The mechanism in which the Domain Name System maps domain names to addresses consists of an independent cooperative system of name servers. A name server such as server **254** of FIG. **2** comprises a server program that supplies name-to-address translation, and mapping from domain names to Internet Protocol addresses. The server comprises the server software typically referred to as the name resolver and the dedicated processor which is referred to as the name server. The name resolver may use one or more name servers when translating a domain name query.

There are two ways to use the domain name system to resolve a domain name query. First, domain name servers may be contacted one at a time for each domain address. Alternatively, a single domain name server may perform a complete translation. In either case, the client software forms a domain name query that contains the name to be resolved. If a client application requests a complete translation, i.e., a recursive resolution, the a primary name server contacts a domain name server that can resolve the name and return the answer to the client. If the client application requests a nonrecursive solution, i.e. an iterative solution, the primary name server will not supply an answer but only generates a reply that specifies the name server the client application should contact to resolve the next portion of the name. In accordance with the Domain Name System, each Domain Name server is required to know the address of at least one route server and must know the address of the server for the domain immediately above it, i.e., the parent domain. Domain name servers use predefined ports for all communications so that clients know how to communicate with a name server once the IP address of the computer on which the server is executing is known. Numerous publicly-available documents describe the domain name system in greater detail including a work entitled Internet Working With TCP/IP Volume 1 by Douglas E. Comer, 2d Edition.

Multiple communication scenarios are available through the network illustrated in FIG. **2**. To facilitate a better understanding of a communication between a packet-switched network and a circuit switched network, examples

of communications over each of those types of networks are first set forth for the reader's benefit. In FIG. **2**, a circuit switched telephone network may comprise the operative interconnection of telephone **214**A, central office **210**A, toll network **260**, central office **210**B and telephone **214**C. A caller at telephone **214**A would place a call by dialing the appropriate telephone number of telephone **214**C. Central offices **210**A–B of the respective carriers will make the appropriate connections through the toll network **260** to establish the connection. Since a toll network is utilized, this scenario likely entails a long distance connection. A simpler example of a circuit switched connection would be the operative interconnection of telephone **214**A, central office **210**A, PBX **212** and telephone **214**B. Since toll network **260** is not used this scenario is more characteristic of a telephone call within a single exchange.

A communication link over a packet-switched network may be established with the network illustrated in FIG. **2**, using the WebPhone protocol as disclosed in U.S. patent application Ser. No. 08/533,115 entitled "POINT-TO-POINT INTERNET PROTOCOL" by Glenn W. Hutton, filed Sep. 25, 1995, previously incorporated herein by reference. Specifically, WebPhone **232**A may connect to Internet **220** through central office **210**A, ISP **250**B and register with global server **252**, notifying server **252** of its current dynamically signed Internet protocol address. Subsequently, WebPhone **232**A may inquire as to the current Internet protocol address of another WebPhone client, for example, WebPhone **232**C. If WebPhone **252** is currently connected to the Internet and has likewise registered with the global server **252** will return the Internet protocol address of WebPhone **232**C to WebPhone **232**A. WebPhone **232**A may then establish a direct connection to WebPhone **232**C via central office **210**A, ISP **250**B, Internet **220**, ISP **250**C, and central office **210**B. Alternatively, a point-to-point connection over a packet-switched network may be established over a local area network **275** by means of a direct connection from WebPhone **232**E to **232**F, such connection being possible if the Internet protocol addresses of the respective WebPhones are fixed.

The present invention provides a solution for communications which cross between circuit-switched (PSTN) networks and packet switched (TCP/IP) networks. The invention addresses at least three calling scenarios involving dialing techniques between any of the WebPhone clients and the gateways. The three dialing scenarios are: 1) WebPhone client to gateway to PSTN, 2) PSTN to gateway to WebPhone Client, and 3) PSTN to first gateway to second gateway to PSTN, as described in greater detail below.

Scenario1, WebPhone Client to GATEWAY to PSTN

In the illustrative embodiment, the invention enables a user to enter a traditional phone number through the graphic user interface of the WebPhone client and establish a call to the specified telephone exchange on a PSTN. A user enters a destination telephone number consisting of country code, area code, exchange and subscriber number segment or data, for example "1-561-997-4001", into the WebPhone client utilizing either the WebPhone virtual keypad or computer keyboard. For purposes of this disclosure, exchange data such as "997" in the above example, are considered to be the same as city codes. For convenience, the WebPhone client may be configured to recognize when either the country or area code information is missing and to insert, by default, a user-defined country code or area code. In the above example, entering 997-4001 would default to a local country code of "**1**" and area code "561". Alternatively, the local gateway **218** may be configured with the appropriate country, area, and city codes, the user need only dial "997-4001" and gateway **218** will append the predefined country and area codes, as explained hereafter. The WebPhone client may also be configured with a default or user selectable PSTN carrier such as MCI, Sprint, AT&T, etc., such carrier will be referred to generically hereafter as "carrier.com" . Upon receiving the desired telephone number the WebPhone client reverses the number and appends the carrier's domain name resulting in a hybrid telephone/domain name having the form "4001-997-561-1.carrier.com". The hybrid telephone number domain name referred to hereafter as the "telephone number domain name" is passed by the WebPhone client to in an acceptable format the name resolver protocol executing on a DNS name server on the TCIP/IP network, as described hereinafter.

FIG. **6** illustrates conceptually the arrangement of domain name servers of the domain name hierarchy illustrated in FIG. **4**. In theory, each server knows the addresses of lower level servers for all subdomains within the domain for which that name server it is responsible. Realistically, a name server for an organization such as carrier.com is connected directly to the root name server. At each node of the domain name tree constructed under the "carrier.com" or "provider**2**.com" node, a name server is configured to return a referral packet containing the IP address of one or more child domain servers. It will be obvious to those skilled in the art that a separate physical name server does not have to exist for each subdomain. A single physical server may function as the name server for multiple domains, so that the entire domain hierarchical tree structure for "carrier.com" or "provider**2**.com" could be configured and supported by one physical server apparatus.

Referring to FIG. **6**, a recursive process of resolving the telephone number domain name previously entered into the WebPhone client to the appropriate IP address of a gateway on a PSTN is illustrated conceptually: In step **1**, the WebPhone client **232** forwards the telephone number domain name to primary name server **254** in packetized form via Internet **220** and ISP **250**. Using a name packet, primary name server **254** queries the root name server of the domain name system (DNS) for the address of "4001.997.561.1.carrier.com" in step **2**. The name server for the DNS root returns a reference to the name server for ".com" in step **3**. Next, name server **254** queries the referenced name server ".com" for the address of "4001.997.561.1.carrier.com" in step **4**. In response, a referral to "carrier.com" is returned in step **5**. Name server **254** then queries the name server "carrier.com" for "4001.997.561.1.carrier.com" in step **6**. In response, a referral to "**1**.carrier.com" is returned in step **7**. Name server **254** then queries the name server to "**1**.carrier.com," for "4001.997.561.1.carrier.com" in step **8**. In response a reference of "561.1.carrier.com", is returned in step **9**. Name server **254** then queries name server for "561.1.carrier.com," in step **10** for "**4001.997.561.1**.carrier.com." In response, a reference to "997.561.1.carrier.com" is returned in step **11**. This last reference contains the IP address of the desired gateway which is then forwarded via Internet **220** and ISP **250** to WebPhone client **232** by name server **254** in step **12**. In the above-described process, the resolution is executed to the level of the exchange code "**997**." It will be obvious to those reasonably skilled in the art that the domain name level may be further resolved if the appropriate name server hierarchy is arranged to allow for resolution to the actual subscriber number domain level or partial subscriber number level. In such an embodiment, specific gateways would be associated with subscriber numbers, therefore allowing the additional step of resolution, using the domain name system.

The process illustrated by FIG. **6** is a recursive process for resolving a telephone number domain name to a network protocol address. The technique of using an iterative process to resolve a telephone number domain name to a network protocol address is substantially similar as that illustrated with reference to FIG. **6** except that the references or network protocol addresses to the respective name servers are returned to the WebPhone client application at each level, either directly or through the primary name server. With such embodiment, rather than the primary name server performing the complete domain name resolution, the WebPhone client is involved at multiple subdomain levels. For example, in step **3**, a reference to the name server for ".com" would be returned directly to the WebPhone client either from the name server for DNS route or through the primary name server. In response, the client application will pass a name packet containing the telephone number domain name to the name server for ".com" identified by the reference. Similar interactions between the client application and the respective domain name servers occur for steps **5–11**, which will not be described hereinafter.

After step **12** of FIG. **6**, the call packet containing the entire telephone number domain name entry "4001.997.561.1.carrier.com" is then sent to initiate a call session to the IP address of the gateway, for example, gateway **218**C, and the call is offered. The gateway **218**C, depending on available resources, then evaluates the call packet data, responds accordingly by dialing 1-561-997-4001 and accepts the call. A call session is then established. The gateway may be coupled directly to a PSTN without connection through either a central office, as illustrated in FIG. **2**. Once a call is established, the gateway serves to translate the packetized audio data from the packet switched-based network into analog or digital signals for the circuit-switched telephone network. The gateway further serves to translate either audio or digital signals from the PSTN network into packetized audio data suitable for transmission over a IP-based network to the WebPhone client.

The reader will appreciate from the foregoing description that the invention provides a method in which an Internet telephone such as the WebPhone clients may specify a traditional telephone number, have that telephone number translated into the IP address of an appropriate gateway by the existing Domain Name System on the Internet, and have the gateway perform the connection through to the telephone on the public switched telephone network. By utilizing the invention as previously described, a gateway may be selected on a least cost routing basis to minimize the toll charges on a traditional PSTN network. For example, if a call is originating with an Internet telephone in the United States to a PSTN apparatus in Germany, the invention may be utilized to identify a gateway in Germany proximate to the terminating apparatus rather than a gateway in the United States. As such, a substantial portion of the costs of the call can be transmitted over the Internet more cheaply than if traditional long-distance and overseas carrier lines were utilized.

The invention further contemplates a model for overriding a default preferred carrier. Suppose that the subscriber intends to use "provider2" to reach the same number. At the user interface, the entered number would be "1-561-997-4001 @@provider2.com". Implemented in this manner, the dialed entry fails the rules for valid E-MAIL addresses. The client WebPhone software, recognizing two @ symbols and the numeric data representing the terminating phone number, could parse the entered data and generate the required domain name of "997.561.1.provider2.com".

The mechanism could be modified to be more readily apparent to the user, but the technique would remain the same.

One or more gateways installed in the above network may be assigned to their own unique E-MAIL addresses. If this information is known to the subscriber, the technique illustrated herein may be modified slightly and be used to route a call to a specific gateway. If the desired WGX was "boca561.997@provider2.com", a user could place the call "1-561-997-4001@boca561.997provider2.com. The WebPhone client, recognizing this as a phone number, would resolve the IP address for "boca561.997@provider2.com" but pass 1-561-887-4001 as the phone number in the call packet when offering the call. If the desired gateway has a fixed IP address, the WebPhone client may connect to the gateway directly and pass the "name" phone number in the call packet when offering the call to the gateway. If the gateways IP address is dynamically assigned, the WebPhone client would contact the global server **252** for the current Internet protocol address of the desired server. Upon receiving such IP address, the WebPhone client would contact the server directly, passing the phone number in the call packet when offering the call. This particular technique is useful when a WebPhone client makes frequent calls to a specific gateway. The WebPhone client may be modified to include either a cashing algorithm or a small directory which includes the E-mail addresses and/or IP addresses of those gateways most frequently called by the WebPhone client. In this manner, the previously described technique for resolving the IP address of a gateway utilizing the domain name system may be avoided and the call initiated directly from the WebPhone client to the gateway, and, ultimately, to the PSTN subscriber.

AAs an alternative to the illustrative embodiment, a provider may install routers or define subnets or some combination thereof for the nodes in its respective domains. Additionally, routers or equipment that manage the subnets may apply algorithms optimized to phone number lookups rather than the general algorithms previously described for resolving names. In the Internet/Intranet environment, these methods are applicable independent of the physical medium on which the communication takes place.

FIG. **7** is a flow chart illustrating the basic process steps used by the WebPhone client in accordance with the present invention. The coding of the process steps of the illustrated flowchart into instructions suitable to control the WebPhone client will be understood by one having ordinary skill in the art of programming. Following power-on initialization and registration with global server **252**, the WebPhone client process remains idle until detecting a telephone number input from either the virtual keyboard or the hardware keyboard associated with the computer on which the WebPhone client is executing, as illustrated by decisional step **700**. The input "name" is then parsed by the WebPhone client as illustrated by process **702** to determine whether or not an E-mail address of a gateway is present within the input character string, as illustrated by decisional block **7504**. If no E-mail address for a carrier gateway is present within the input name, the name is formatted as illustrated in procedural step **706**. As described previously, formatting may include reversing the order of the actual segments of the phone number name as well as pending predefined or default values for country codes, area codes, and carriers, etc. Once the phone number domain name has been properly formatted, as illustrated in procedure step **706**, the names forwarded to a domain system server as previously described. The WebPhone client awaits the return of the IP

address of the gateway as determined by one or more domain name system servers in accordance with the process steps previously described with reference to FIG. **4**, e.g. iterative resolution of the domain name supplied. If, in decisional step **704**, the E-mail address of a gateway had been entered, the WebPhone client would attempt to resolve the IP address of the gateway either from its own local directory, in the event of a gateway having a fixed IP address, or through clearing the global server **252** to which the WebPhone client is operatively coupled over an IP-based network.

Having received the Internet protocol address of the appropriate gateway, either through process step **710** or from a domain name server, the WebPhone client forwards a call packet containing the telephone number domain name to the gateway and offers a call, as illustrated in procedural step **714** and **716**. If a call is established, as illustrated in decisional block **718**, the WebPhone client will function as described in the previously referenced patent applications. Otherwise, the WebPhone client will eventually return to an idle mode, awaiting input for an additional telephone number data. In the event a call is established, the WebPhone will detect the disconnection of the call as illustrated in procedural block **720** and will return to an idle state if not powered down, as illustrated by decisional block **722**.

The reader will appreciate that the inventive technique provides a method for entering traditional PSTN telephone numbers into an Internet telephone application and having the call completed through use of the domain name server and a gateway to complete the call in the manner which is efficient and may be substantially transparent to the user.

Scenario 2, PSTN to GATEWAY to WebPhone Client

A gateway user calling from the PSTN would be prompted for the desired phone number and user number and pin code via voice response or voice recognition techniques. The virtual WebPhone proxy device functions similar to Scenario. The destination phone number is resolved based on the configured location of the gateway (country code, area code, exchange, etc.), and the configured service provider. Scenario 2 is similar to Scenario 1 except that the gateway may be configured with user information, including E-MAIL and phone number of a specific WebPhone client, and then utilized as a inverse connection server to return the E-MAIL address of the desired parties' WebPhone client. The gateway through which a PSTN originating call is received would be configured with information indicating which connection server, similar to global server **252**, contains the E-mail address corresponding to the desired phone number. The gateway then connects to the global server to determine the current IP address of the WebPhone client callee. Upon receiving the IP address of the WebPhone client, the gateway then establishes a direct contact to the WebPhone client and offers the call. When the WebPhone client is reached, the user, through the capabilities of the virtual WebPhone proxy software of the gateway, has the ability to leave online or offline voice mail or communication in real time, if desired.

Scenario 3, PSTN to GATEWAY to GATEWAY to PSTN

The technique utilized in the previously described Scenario 2 may be utilized to establish calls originating and ending at PSTN apparatus, but which utilize a portion of a IP-based network, typically to reduce costs. For example, in the previously described scenario, instead of contacting a global server for the IP address of the destination WebPhone client, the gateway would be configured with information relating to which other gateway would be most suitable in accordance with any number of predetermined conditions, including lease cost routing, to complete the call. The initially contacted gateway would then contact the selected gateway, using either the E-mail address of the gateway or a fixed IP address and offer a call packet, including the destination phone number. If the call is accepted, a real-time communication link will be established from telephone **214**A, to gateway **218**B, through ISP **250**B, Internet **220**, ISP **250**C, gateway **218**C, and onto telephone **214**C. As such, the single communication link will originate on a circuit switched network, bridge a packet-switched data communication network, and terminate on a second circuit-switched network.

Subscriber Validation and Billing

In addition to the inventive technique for calling over the Internet/Intranet, the ability to validate a subscriber's identity and rights and perform billing services is desirable. In the WebPhone client, one piece of required information is a subscriber phone number. In the client implementation, this is intended to be a valid traditional phone number for the user's home or office desktop phone. If this phone number comprises a country code, area code, exchange and subscriber number, and additionally a user pin code (or 4 or more digits), the phone number can be resolved to a specific domain and gateway, and the user may be validated by their respective home access gateway. A gateway, which provides service for the user, is configured with similar information. For example, the user, configured with a home number and pin code desires to place a call to "972-4-99777-583" in Israel. Additionally, the user is currently located in Germany. The user configuration information, which may or may not employ encryption depending on the secure nature of the network, would be sent by the client in a user validation packet to the gateway servicing "4001.997.407.1.carrier.com". A response approving or disapproving the individual is returned. The client then repeats the above phone number resolution process for the terminating number "583.9977.4.972.provider2.com".

In a similar method to that described for subscriber validation, billing information may be collected by the gateway and collected later over the same network. Call statistics (destination, length of call, number of bytes transferred, quality of service, etc.) may be collected by the home gateway and periodically downloaded and analyzed.

The gateway may function as a personal attendant for the subscriber, since user network related queries may be resolved at the gateway, the gateway could offer services such as voice mail, find me—follow me services, custom outgoing message storage site, personal out going messages storage/playback based on caller id information, account information retrieval, LAPD personal directory and business card information, etc. Depending on available resources such as available memory, processor capability and network bandwidth, a gateway may provide services for 10,000 or more customers.

The techniques described herein for call resolution and user validation distributes the load while also improving reliability, and further, does not preclude alternate call resolution strategies when success is not achieved. The reliability and success rate of placing calls may be improved by employing alternate techniques according to error con-

ditions. Failure to resolve the gateway address may result in a second attempt. Clients sending ONLINE packets with a specified service provider may be forwarded a list of containing the E-MAIL or IP addresses for other redundant, alternate equipment in the event of an equipment failure along with instructions to direct the ONLINE packet to one of the alternate servers or gateways.

The above described inventive technique may be utilized regardless of the specific name of the domain under which the carrier name server is implemented. For example, the authority which regulates domain names may add a special ".tel" domain specifically designed for carriers and other businesses providing telecommunication services. Such a domain name would facilitate ease of use of the technique but would not change the technique described herein.

A software implementation may comprise a series of computer instructions either fixed on a tangible medium, such as a computer readable media, e.g. diskette **142**, CD-ROM **147**, ROM **115**, or fixed disk **152** of FIG. **1**A, or transmittable to a computer system, via a modem or other interface device, such as communications adapter **190** connected to the network **195** over a medium **191**. Medium **191** can be either a tangible medium, including but not limited to optical or analog communications lines, or may be implemented with wireless techniques, including but not limited to microwave, infrared or other transmission techniques. The series of computer instructions embodies all or part of the functionality previously described herein with respect to the invention. Those skilled in the art will appreciate that such computer instructions can be written in a number of programming languages for use with many computer architectures or operating systems. Further, such instructions may be stored using any memory technology, present or future, including, but not limited to, semiconductor, magnetic, optical or other memory devices, or transmitted using any communications technology, present or future, including but not limited to optical, infrared, microwave, or other transmission technologies. It is contemplated that such a computer program product may be distributed as a removable media with accompanying printed or electronic documentation, e.g., shrink wrapped software, preloaded with a computer system, e.g., on system ROM or fixed disk, or distributed from a server or electronic bulletin board over a network, e.g., the Internet or World Wide Web.

Although various exemplary embodiments of the invention have been disclosed, it will be apparent to those skilled in the art that various changes and modifications can be made which will achieve some of the advantages of the invention without departing from the spirit and scope of the invention. Further, many of the system components described herein such as the WebPhone client application and the gateway have been described using products from NetSpeak Corporation. It will be obvious to those reasonably skilled in the art that other components performing the same functions may be suitably substituted. Further, the methods of the invention may be achieved in either all software implementations, using the appropriate processor instructions, or in hybrid implementations which utilize a combination of hardware logic and software logic to achieve the same results. Such modifications to the inventive concept are intended to be covered by the appended claims.

What is claimed is:

1. A method for resolving data representing a telephone number of subscriber apparatus on a circuit-switched communication network into a network protocol address on a packet-switched data network, the method comprising:

A. receiving a telephone number domain name derived at least in part from a reordered portion of the telephone number;

B. resolving the telephone number domain name into a network protocol address; and

C. supplying the network protocol address to the source;

wherein the telephone number comprises a plurality of segments, and the reordered portion comprises the segments in a reverse order while maintaining the number sequence of each segment.

2. The method of claim **1** wherein at least a portion of the telephone number domain name represents one of the carrier, country code, area code, exchange and subscriber number segment of the telephone number.

3. The method of claim **1** wherein the telephone number domain name comprises a segment representing a carrier domain name and step B comprises:

B.1 resolving a carrier domain name into a network protocol address of a carrier name server;

B.2 forwarding at least a portion of the telephone number domain name to the carrier name server.

4. The method of claim **1** wherein the telephone number domain name further comprises a country code domain name and wherein step B further comprises:

B.1 resolving the country code domain name into a network protocol address of a country code domain name server;

B.2 forwarding the country code domain name to the country code domain name server.

5. The method of claim **1** wherein the telephone number domain name further comprises an area code domain name and wherein step B further comprises:

B.1 resolving the area code domain name into an network protocol address of an area code domain name server;

B.2 forwarding the area code domain name to an area code domain name server.

6. The method of claim **1** wherein the telephone number domain name further comprises an exchange domain name and wherein step B further comprises:

B.1 resolving the exchange domain name into a network protocol address of an exchange domain name server; and

B.2 forwarding the exchange domain name to the exchange domain name server.

7. The method of claim **1** wherein the telephone domain name comprises data representing the telephone number.

8. The method of claim **7** wherein the telephone domain name further comprises data representing a carrier domain name.

9. The method of claim **1** wherein the network protocol address comprises an Internet protocol address.

10. A domain name server apparatus for use on a computer network, the server apparatus accessible by one or more source client processes executing on the computer network, the domain name server apparatus comprising:

a processor for manipulating data;

a memory coupled to the processor for storing data;

connection logic, coupled to the processor and the memory, configured to couple the domain name server to a computer network;

at least one domain name stored in the memory, the domain name having associated therewith a network protocol address and having a portion thereof derived at least in part from a reordered portion of a telephone number;

resolution logic, responsive to at least a portion of a telephone number domain name received from a

source, for generating a network protocol address associated with the portion of the telephone number domain name and for supplying the generated network protocol address to the source;

wherein the telephone number comprises a plurality of segments, and the reordered portion comprises the segments in a reverse order while maintaining the number sequence of each segment.

11. The domain name server apparatus of claim **10** further comprising:

a plurality of domain name labels stored in memory, each label having associated therewith a network protocol address, each domain name label representing at least one of the carrier, country code, area code, exchange and subscriber number segment of a telephone number.

12. The domain name server apparatus of claim **10** further comprising:

a plurality of domain name labels stored in memory, each domain name label having associated therewith a network protocol address, each domain name label representing country code data of a telephone number.

13. The domain name server apparatus of claim **10** further comprising:

a plurality of domain name labels stored in memory, each domain name label having associated therewith a network protocol address, each domain name label representing area code data of a telephone number.

14. The domain name server apparatus of claim **10** further comprising:

a plurality of domain name labels stored in memory, each domain name label having associated therewith a network protocol address, each domain name label representing exchange number data of a telephone number.

15. A computer program product for use with a computer system, the computer system operatively coupled over a packet-switched data network to one or more executing tasks, the computer program product comprising a computer usable medium having program code embodied in the medium for enabling the translation of data representing the telephone number of subscriber apparatus on a circuit-switched network into a network protocol address, the telephone number comprising area code, exchange, and subscriber segments, each segment having a sequence of at least one number, the program code comprising:

program code for receiving a telephone number domain name from a source, the telephone number domain comprising the segments of a telephone number in reverse order but with the number sequence of each segment maintained;

program code responsive to at least a portion of the telephone number domain name for generating a network protocol address; and

program code, responsive to the generated network protocol address, for forwarding the network protocol address to the source.

16. The computer program product of claim **15** wherein the program code for generating a network protocol address comprises:

program code for storing in the computer system, at least one domain name having associated therewith a network protocol address, the domain name representing at least one of the carrier, country code, area code, exchange, and subscriber number data of a telephone number.

17. The computer program product of claim **15** wherein the program code for generating a network protocol address comprises:

program code for storing in the computer system, a plurality of domain names, each having associated therewith a network protocol address, each of the domain names representing country code data of a telephone number.

18. The computer program product of claim **15** wherein the program code for generating a network protocol address comprises:

program code for storing in the computer system, a plurality of domain names, each having associated therewith a network protocol address, each of the domain names representing area code data of a telephone number.

19. The computer program product of claim **15** wherein the program code for generating a network protocol address comprises:

program code for storing in the computer system, a plurality of domain names, each having associated therewith a network protocol address, each of the domain names representing exchange data of a telephone number.

20. A system for facilitating communication between processes executing on a packet-switched communication network, the system comprising:

a domain name server operatively coupled to the packet-switched network, the domain name server comprising resolution logic configured to receive, from a source process, a telephone number domain name derived at least in part from a reordered portion of a telephone number, and configured to generate a network protocol address associated with at least a portion of the telephone number domain name, wherein the telephone number comprises a plurality of segments, and the reordered portion comprises the segments in a reverse order while maintaining the number sequence of each segment; and

a gateway server, operatively coupled to the packet-switched data network and the circuit-switched communication network, the gateway server addressable by the network protocol address and comprising logic configured to initiate a communication link between subscriber apparatus on the circuit-switched communication network and the source process on the packet-switched data network.

21. The method of claim **1** wherein the telephone number comprises area code, exchange, and subscriber segments, each segment having a sequence of at least one number, and wherein the telephone number domain name comprises the segments of a telephone number in reverse order but with the number sequence of each segment maintained.

22. The apparatus of claim **10** wherein the telephone number comprises area code, exchange, and subscriber segments, each segment having a sequence of at least one number, and wherein the telephone number domain name comprises the segments of a telephone number in reverse order but with the number sequence of each segment maintained.

23. The system of claim **20** wherein the telephone number comprises area code, exchange, and subscriber segments, each segment having a sequence of at least one number, and wherein the telephone number domain name comprises the segments of a telephone number in reverse order but with the number sequence of each segment maintained.

* * * * *

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T H.323

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

(07/2003)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Systems and terminal equipment for audiovisual services

Packet-based multimedia communications systems

ITU-T Recommendation H.323

ITU-T H-SERIES RECOMMENDATIONS

AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
SYSTEMS AND TERMINAL EQUIPMENT FOR AUDIOVISUAL SERVICES	**H.300–H.399**
SUPPLEMENTARY SERVICES FOR MULTIMEDIA	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND AND TRIPLE-PLAY MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation H.323

Packet-based multimedia communications systems

Summary

This Recommendation describes terminals and other entities that provide multimedia communications services over Packet Based Networks (PBN) which may not provide a guaranteed Quality of Service. H.323 entities may provide real-time audio, video and/or data communications. Support for audio is mandatory, while data and video are optional, but if supported, the ability to use a specified common mode of operation is required, so that all terminals supporting that media type can interwork.

The packet based network over which H.323 entities communicate may be a point-to-point connection, a single network segment, or an internetwork having multiple segments with complex topologies.

H.323 entities may be used in point-to-point, multipoint, or broadcast (as described in ITU-T Rec. H.332) configurations. They may interwork with H.310 terminals on B-ISDN, H.320 terminals on N-ISDN, H.321 terminals on B-ISDN, H.322 terminals on Guaranteed Quality of Service LANs, H.324 terminals on GSTN and wireless networks, V.70 terminals on GSTN, and voice terminals on GSTN or ISDN through the use of Gateways.

H.323 entities may be integrated into personal computers or implemented in stand-alone devices such as videotelephones.

Note that the title of H.323 (1996) was "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service". The title changed in Version 2 to be consistent with its expanded scope.

Products claiming compliance with Version 1 of H.323 shall comply with all of the mandatory requirements of H.323 (1996) which references ITU-T Recs H.225.0 (1996) and H.245 (1997). Version 1 products shall be identified by H.225.0 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 2250 version (0) 1} and H.245 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 245 version (0) 2}.

Products claiming compliance with Version 2 of H.323 shall comply with all of the mandatory requirements of this Recommendation, H.323 (1998), which references ITU-T Recs H.225.0 (1998) and H.245 (1998 or later). Version 2 products shall be identified by H.225.0 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 2250 version (0) 2} and H.245 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 245 version (0) x}, where "x" is 3 or higher.

Products claiming compliance with Version 3 of H.323 shall comply with all of the mandatory requirements of this Recommendation, H.323 (1999), which references ITU-T Recs H.225.0 (1999) and H.245 (1999 or later). Version 3 products shall be identified by H.225.0 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 2250 version (0) 3} and H.245 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 245 version (0) x}, where "x" is 5 or higher.

Products claiming compliance with Version 4 of H.323 shall comply with all of the mandatory requirements of this Recommendation, H.323 (2000), which references ITU-T Recs H.225.0 (2000) and H.245 (2000 or later). Version 4 products shall be identified by H.225.0 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 2250 version (0) 4} and H.245 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 245 version (0) x}, where "x" is 7 or higher.

Products claiming compliance with Version 5 of H.323 shall comply with all of the mandatory requirements of this Recommendation, H.323 (2003), which references ITU-T Recs H.225.0 (2003) and H.245 (02/2003 or later). Version 5 products shall be identified by H.225.0 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 2250 version (0) 5} and H.245 messages containing a **protocolIdentifier** = {itu-t (0) recommendation (0) h (8) 245 version (0) x}, where "x" is 9 or higher.

This version of ITU-T Rec. H.323 integrates without further modifications Annexes M3 (07/2001), P (01/2003), Q (07/2001) and R (07/2001), as well as Annex O, approved independently 07/2003.

Source

ITU-T Recommendation H.323 was approved by ITU-T Study Group 16 (2001-2004) under the ITU-T Recommendation A.8 procedure on 14 July 2003.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Page

Page

ITU-T Recommendation H.323

Packet-based multimedia communications systems

1 Scope

This Recommendation covers the technical requirements for multimedia communications systems in those situations where the underlying transport is a Packet Based Network (PBN) which may not provide a guaranteed Quality of Service (QOS). These packet based networks may include Local Area Networks, Enterprise Area Networks, Metropolitan Area Networks, Intra-Networks and Inter-Networks (including the Internet). They also include dial up connections or point-to-point connections over the GSTN or ISDN which use an underlying packet based transport such as PPP. These networks may consist of a single network segment, or they may have complex topologies which incorporate many network segments interconnected by other communications links.

This Recommendation describes the components of an H.323 system. This includes Terminals, Gateways, Gatekeepers, Multipoint Controllers, Multipoint Processors and Multipoint Control Units. Control messages and procedures within this Recommendation define how these components communicate. Detailed descriptions of these components are contained in clause 6.

H.323 terminals provide audio and optionally video and data communications capability in point-to-point or multipoint conferences. Interworking with other H-series terminals, GSTN or ISDN voice terminals, or GSTN or ISDN data terminals is accomplished using Gateways. See Figure 1. Gatekeepers provide admission control and address translation services. Multipoint Controllers, Multipoint Processors and Multipoint Control Units provide support for multipoint conferences.

The scope of H.323 does not include the network interface, the physical network or the transport protocol used on the network. Examples of these networks include but are not limited to:

– Ethernet (IEEE 802.3);
– Fast Ethernet (IEEE 802.3u);
– FDDI;
– Token Ring (IEEE 802.5);
– ATM.

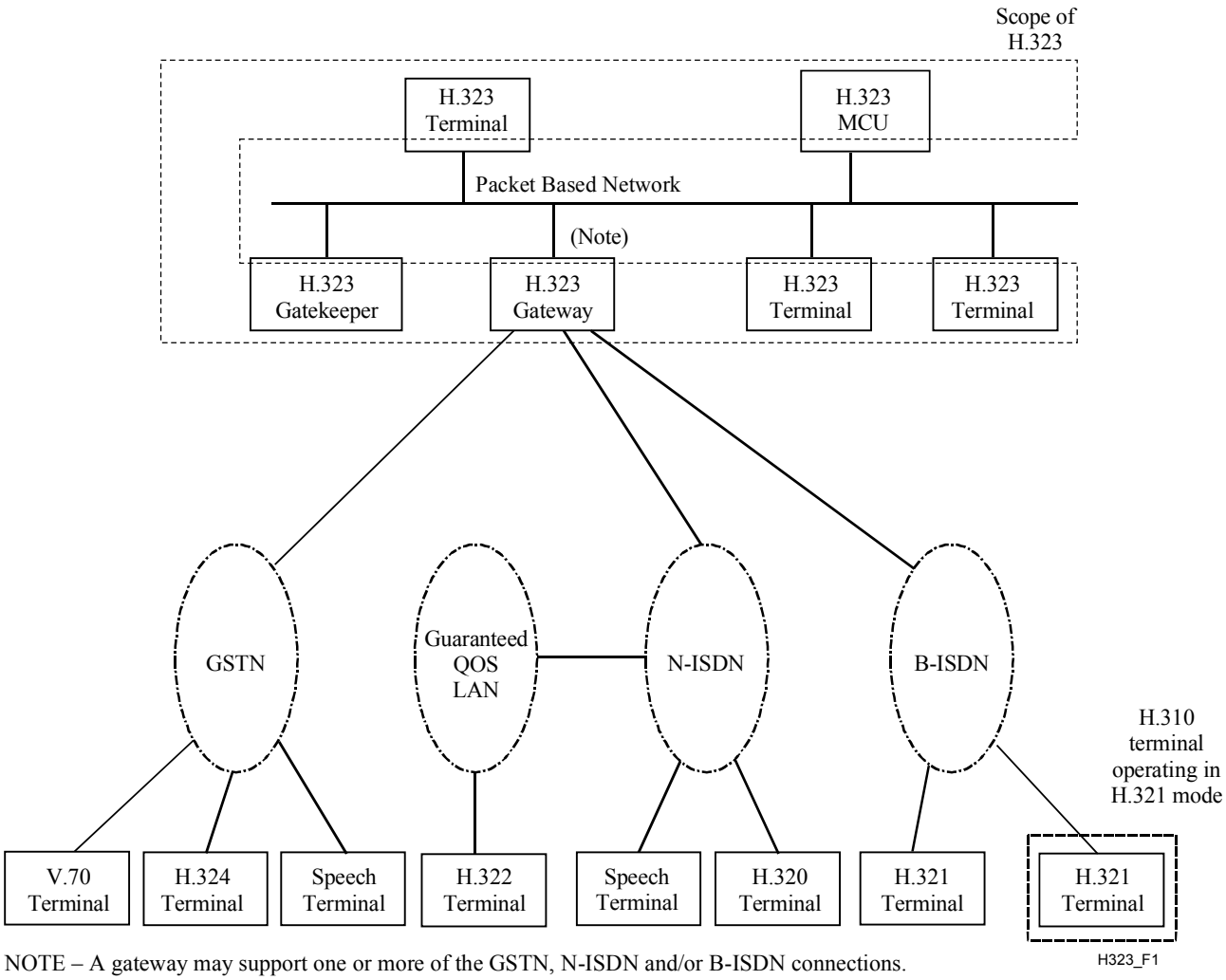


Figure 1/H.323 – Interoperability of H.323 terminals

2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[1] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

[2] ITU-T Recommendation H.245 (2003), *Control protocol for multimedia communication*.

[3] ITU-T Recommendation G.711 (1988), *Pulse Code Modulation (PCM) of voice frequencies*.

[4] ITU-T Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s*.

[5] ITU-T Recommendation G.723.1 (1996), *Speech coders: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s*.

[6] ITU-T Recommendation G.728 (1992), *Coding of speech at 16 kbit/s using low-delay code excited linear prediction*.

[7] ITU-T Recommendation G.729 (1996), *Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)*.

[8] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at p × 64 kbit/s*.

[9] ITU-T Recommendation H.263 (1998), *Video coding for low bit rate communication*.

[10] ITU-T Recommendation T.120 (1996), *Data protocols for multimedia conferencing*.

[11] ITU-T Recommendation H.320 (1999), *Narrow-band visual telephone systems and terminal equipment*.

[12] ITU-T Recommendation H.321 (1998), *Adaptation of H.320 visual telephone terminals to B-ISDN environments*.

[13] ITU-T Recommendation H.322 (1996), *Visual telephone systems and terminal equipment for local area networks which provide a guaranteed quality of service*.

[14] ITU-T Recommendation H.324 (2002), *Terminal for low bit-rate multimedia communication*.

[15] ITU-T Recommendation H.310 (1998), *Broadband audiovisual communication systems and terminals*.

[16] ITU-T Recommendation Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control*.

[17] ITU-T Recommendation Q.932 (1998), *Digital subscriber signalling system No. 1 – Generic procedures for the control of ISDN supplementary services*.

[18] ITU-T Recommendation Q.950 (2000), *Supplementary services protocols, structure and general principles*.

[19] ISO/IEC 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (USC) – Part 1: Architecture and basic multilingual plane*.

[20] ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan*.

[21] ITU-T Recommendation H.246 (1998), *Interworking of H-series multimedia terminals with H-series multimedia terminals and voice/voiceband terminals on GSTN and ISDN*.

[22] ITU-T Recommendation H.235 (2003), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.

[23] ITU-T Recommendation H.332 (1998), *H.323 extended for loosely coupled conferences*.

[24] ITU-T Recommendation H.450.1 (1998), *Generic functional protocol for the support of supplementary services in H.323*.

[25] ITU-T Recommendation I.363.5 (1996), *B-ISDN ATM Adaptation Layer specification: Type 5 AAL*.

[26] ITU-T Recommendation Q.2931 (1995), *Digital subscriber signalling system No. 2 – User-network interface (UNI) layer 3 specification for basic call/connection control*.

[27] ITU-T Recommendation I.356 (2000), *B-ISDN ATM layer cell transfer performance*.

[28] ITU-T Recommendation I.371 (2000), *Traffic control and congestion control in B-ISDN*.

[29] ITU-T Recommendation I.371.1 (2000), *Guaranteed frame rate ATM transfer capability*.

[30] ITU-T Recommendation Q.2961.2 (1997), *Digital subscriber signalling system No. 2 – Additional traffic parameters: Support of ATM transfer capability in the broadband bearer capability information element*.

[31] ITU-T Recommendation H.282 (1999), *Remote device control protocol for multimedia applications*.

[32] ITU-T Recommendation H.283 (1999), *Remote device control logical channel transport*.

[33] ATM Forum AF-SAA-0124.000 (1999), *H.323 Media Transport Over ATM*.

[34] ITU-T Recommendation Q.2941.2 (1999), *Digital subscriber signalling system No. 2 – Generic identifier transport extensions*.

[35] ITU-T Recommendation H.450.2 (1998), *Call transfer supplementary service for H.323*.

[36] ITU-T Recommendation H.450.4 (1999), *Call hold supplementary service for H.323*.

[37] ITU-T Recommendation H.248 (2000), *Gateway control protocol*.

[38] ISO/IEC 11571:1998, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Networks – Addressing*.

[39] ITU-T Q.951.x family Recommendations, *Stage 3 description for number identification supplementary services using DSS 1*.

[40] ITU-T Recommendation H.450.3 (1998), *Call diversion supplementary service for H.323*.

[41] ITU-T Recommendation H.450.5 (1999), *Call park and call pickup supplementary services for H.323*.

[42] ITU-T Recommendation H.450.6 (1999), *Call waiting supplementary service for H.323*.

[43] ITU-T Recommendation H.450.7 (1999), *Message waiting indication supplementary service for H.323*.

[44] ITU-T Recommendation H.450.8 (2000), *Name identification supplementary service for H.323*.

[45] ISO/IEC 11572:2000, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Circuit mode bearer services – Inter-exchange signalling procedures and protocol*.

[46] ITU-T Recommendation H.222.0 (2000), *Information technology – Generic coding of moving pictures and associated audio information: Systems*.

[47] ITU-T Recommendation H.223 (2001), *Multiplexing protocol for low bit rate multimedia communication*.

[48] IETF RFC 2068 (1997), *Hypertext Transfer Protocol – HTTP/1.1*.

[49] IETF RFC 2045 (1996), *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*.

[50] ITU-T Recommendation Z.100 (2002), *Specification and Description Language (SDL)*.

[51] IETF RFC 1738 (1994), *Uniform Resource Locators (URL)*.

[52] IETF RFC 2234 (1997), *Augmented BNF for Syntax Specifications: ABNF*.

[53] ISO 4217:2001, *Codes for the representation of currencies and funds*.

[54] ITU-T Recommendation V.21 (1988), *300 bits per second duplex modem standardized for use in the general switched telephone network.*

[55] ITU-T Recommendation T.30 (2003), *Procedures for document facsimile transmission in the general switched telephone network*.

[56] ITU-T Recommendation T.38 (2002), *Procedures for real-time Group 3 facsimile communication over IP networks*.

[57] ISO/IEC 10646-1:2000, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*.

[58] IETF RFC 2833 (2000), *RTP Payload for DTMF Digits, Telephony Tones and Telephony Signals*.

3 Definitions

For the purposes of this Recommendation the definitions given in clause 3/H.225.0 [1] and clause 3/H.245 [2] apply along with those in this clause. These definitions apply to the packet based network side only. Other terms may be appropriate when referring to the Switched Circuit Network (SCN) side. See clause 5, Conventions, for information on the use of terms in this Recommendation.

3.1 access gateway: A Gateway that connects one network to another network (such as an SS7 network to a QSIG network) and performs some interworking function between the different networks.

3.2 active MC: An MC that has won the master-slave determination procedure and is currently providing the multipoint control function for the conference.

3.3 ad hoc multipoint conference: An Ad Hoc Multipoint conference was a point-to-point conference that had been expanded into a multipoint conference at some time during the call. This can be done if one or more of the terminals in the initial point-to-point conference contains an MC, if the call is made using a Gatekeeper that includes MC functionality, or if the initial call is made through an MCU as a multipoint call between only two terminals.

3.4 addressable: An H.323 entity on the network having a Transport Address is addressable. Not the same as being callable. A terminal, Gateway, or MCU is addressable and callable. A Gatekeeper is addressable but not callable. An MC or MP is neither callable nor addressable but is contained within an endpoint or Gatekeeper that is. In a composite Gateway, both the MGC and the MG are addressable, but only the MGC is callable.

3.5 audio mute: Suppressing of the audio signal of a single or all source(s). Send muting means that the originator of an audio stream mutes its microphone and/or does not transmit any audio signal at all. Receive mute means that the receiving terminal ignores a particular incoming audio stream or mutes its loudspeaker.

3.6 broadcast conference: A Broadcast conference is one in which there is one transmitter of media streams and many receivers. There is no bidirectional transmission of control or media streams. Such conferences should be implemented using network transport multicast facilities, if available. Also see ITU-T Rec. H.332 [23].

3.7 broadcast panel conference: A Broadcast Panel conference is a combination of a Multipoint conference and a Broadcast conference. In this conference, several terminals are engaged in a multipoint conference, while many other terminals are only receiving the media streams. There is bidirectional transmission between the terminals in the multipoint portion of the conference and no bidirectional transmission between them and the listening terminals. Also see ITU-T Rec. H.332.

3.8 call: Point-to-point multimedia communication between two H.323 endpoints. The call begins with the call set-up procedure and ends with the call termination procedure. The call consists of the collection of reliable and unreliable channels between the endpoints. A call may be directly

between two endpoints or may include other H.323 entities such as a Gatekeeper or MC. In case of interworking with some SCN endpoints via a Gateway, all the channels terminate at the Gateway where they are converted to the appropriate representation for the SCN end system. Typically, a call is between two users for the purpose of communication, but may include signalling-only calls. An endpoint may be capable of supporting multiple simultaneous calls.

3.9 call signalling channel: Reliable channel used to convey the call setup and teardown messages (following ITU-T Rec. H.225.0) between two H.323 entities.

3.10 callable: Capable of being called as described in clause 8 or in the supplementary services ITU-T (H.450.x). In other words, an H.323 entity is generally considered callable if a user would specify the entity as a destination. Terminals, MCUs, Gateways, and MGCs are callable, but Gatekeepers, MCs and MGs are not.

3.11 centralized multipoint conference: A Centralized Multipoint conference is one in which all participating terminals communicate in a point-to-point fashion with an MCU. The terminals transmit their control, audio, video, and/or data streams to the MCU. The MC within the MCU centrally manages the conference. The MP within the MCU processes the audio, video and/or data streams and returns the processed streams to each terminal.

3.12 composite gateway: A Gateway that does not separate the Media Gateway Controller and Media Gateway functions.

3.13 control and indication: End-to-end signalling between terminals, consisting of Control, which causes a state change in the receiver, and Indication which provides for information as to the state or functioning of the system (see also ITU-T Rec. H.245 [2] for additional information and abbreviations).

3.14 data: Information stream other than audio, video, and control, carried in the logical data channel (see ITU-T Rec. H.225.0 [1]).

3.15 decentralized multipoint conference: A Decentralized Multipoint conference is one in which the participating terminals multicast their audio and video to all other participating terminals without using an MCU. The terminals are responsible for:

a) summing the received audio streams; and

b) selecting one or more of the received video streams for display.

No audio or video MP is required in this case. The terminals communicate on their H.245 Control Channels with an MC which manages the conference. The data stream is still centrally processed by the MCS-MCU which may be within an MP.

3.16 decomposed gateway: A Gateway that is functionally separated into a Media Gateway Controller and one or more Media Gateways.

3.17 endpoint: An H.323 terminal, Gateway, or MCU. An endpoint can call and be called. It generates and/or terminates information streams.

3.18 gatekeeper: The Gatekeeper (GK) is an H.323 entity on the network that provides address translation and controls access to the network for H.323 terminals, Gateways and MCUs. The Gatekeeper may also provide other services to the terminals, Gateways and MCUs such as bandwidth management and locating Gateways.

3.19 gateway: An H.323 Gateway (GW) is an endpoint on the network which provides for real-time, two-way communications between H.323 Terminals on the packet based network and other ITU Terminals on a switched circuit network or to another H.323 Gateway. Other ITU Terminals include those complying with ITU-T Rec. H.310 (H.320 on B-ISDN), H.320 (ISDN), H.321 (ATM), H.322 (GQOS-LAN), H.324 (GSTN), H.324M (Mobile), and V.70 (DSVD).

3.20 H.323 entity: Any H.323 component, including terminals, Gateways, Gatekeepers, MCs, MPs and MCUs.

3.21 H.245 control channel: Reliable Channel used to carry the H.245 control information messages (following ITU-T Rec. H.245) between two H.323 endpoints.

3.22 H.245 session: The part of the call that begins with the establishment of an H.245 Control Channel and ends with the receipt of the H.245 **EndSessionCommand** or termination due to failure. Not to be confused with a call, which is delineated by the H.225.0 Setup and Release Complete messages.

3.23 hybrid multipoint conference – centralized audio: A Hybrid Multipoint – Centralized Audio conference is one in which terminals multicast their video to other participating terminals and unicast their audio to the MP for mixing. The MP returns a mixed audio stream to each terminal.

3.24 hybrid multipoint conference – centralized video: A Hybrid Multipoint – Centralized Video conference is one in which terminals multicast their audio to other participating terminals and unicast their video to the MP for switching or mixing. The MP returns a video stream to each terminal.

3.25 information stream: A flow of information of a specific media type (e.g., audio) from a single source to one or more destinations.

3.26 lip synchronization: Operation to provide the feeling that speaking motion of the displayed person is synchronized with his speech.

3.27 local area network (LAN): A shared or switched medium, peer-to-peer communications network that broadcasts information for all stations to receive within a moderate-sized geographic area, such as a single office building or a campus. The network is generally owned, used and operated by a single organization. In the context of this Recommendation, LANs also include internetworks composed of several LANs that are interconnected by bridges or routers.

3.28 logical channel: Channel used to carry the information streams between two H.323 endpoints. These channels are established following the H.245 **OpenLogicalChannel** procedures. An unreliable channel is used for audio, audio control, video and video control information streams. A reliable channel is used for data and H.245 control information streams. There is no relationship between a logical channel and a physical channel.

3.29 media gateway: The Media Gateway converts media provided in one type of network to the format required in another type of network. For example, an MG could terminate bearer channels from a switched circuit network (i.e., DS0s) and media streams from a packet network (e.g., RTP streams in an IP network). This Gateway may be capable of processing audio, video and T.120 alone or in any combination, and will be capable of full duplex media translations. The MG may also play audio/video messages and perform other IVR functions or may perform media conferencing.

3.30 media gateway controller: Controls the parts of the call state that pertain to connection control for media channels in an MG.

3.31 mixed multipoint conference: A Mixed Multipoint conference (see Figure 2) has some terminals (D, E and F) participating in a centralized mode while other terminals (A, B and C) are participating in a decentralized mode. A terminal is not aware of the mixed nature of the conference, only of the type of conference it is participating in. The MCU provides the bridge between the two types of conferences.

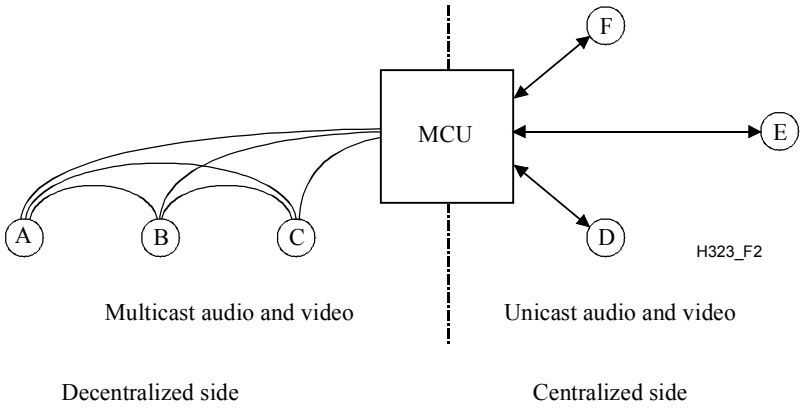


Figure 2/H.323 – Mixed multipoint conference

3.32 multicast: A process of transmitting PDUs from one source to many destinations. The actual mechanism (i.e., IP multicast, multi-unicast, etc.) for this process may be different for different network technologies.

3.33 multipoint conference: A Multipoint conference is a conference between three or more terminals. The terminals may be on the network or on the SCN. The multipoint conference shall always be controlled by an MC. Various multipoint conference types are defined in this subclause but they all require a single MC per conference. They may also involve one or more H.231 MCUs on the SCN. A terminal on the network may also participate in an SCN multipoint conference by connecting via a Gateway to an SCN-MCU. This does not require the use of an MC.

3.34 multipoint control unit: The Multipoint Control Unit (MCU) is an endpoint on the network which provides the capability for three or more terminals and Gateways to participate in a multipoint conference. It may also connect two terminals in a point-to-point conference which may later develop into a multipoint conference. The MCU generally operates in the fashion of an H.231 MCU; however, an audio processor is not mandatory. The MCU consists of two parts: a mandatory Multipoint Controller and optional Multipoint Processors. In the simplest case, an MCU may consist only of an MC with no MPs. An MCU may also be brought into a conference by the Gatekeeper without being explicitly called by one of the endpoints.

3.35 multipoint controller: The Multipoint Controller (MC) is an H.323 entity on the network which provides for the control of three or more terminals participating in a multipoint conference. It may also connect two terminals in a point-to-point conference which may later develop into a multipoint conference. The MC provides for capability negotiation with all terminals to achieve common levels of communications. It may also control conference resources such as who is multicasting video. The MC does not perform mixing or switching of audio, video and data.

3.36 multipoint processor: The Multipoint Processor (MP) is an H.323 entity on the network which provides for the centralized processing of audio, video and/or data streams in a multipoint conference. The MP provides for the mixing, switching or other processing of media streams under the control of the MC. The MP may process a single media stream or multiple media streams depending on the type of conference supported.

3.37 multi-unicast: A process of transferring PDUs where an endpoint sends more than one copy of a media stream, but to different endpoints. This may be necessary in networks which do not support multicast.

3.38 network address: The network layer address of an H.323 entity as defined by the (inter)network layer protocol in use (e.g., an IP address). This address is mapped onto the layer one address of the respective system by some means defined in the (inter)networking protocol.

3.39 packet based network (also network): Any shared, switched, or point-to-point medium which provides peer-to-peer communications between two or more endpoints using a packet based transport protocol.

3.40 point-to-point conference: A Point-to-Point conference is a conference between two terminals. It may be either directly between two H.323 terminals or between an H.323 terminal and an SCN terminal via a Gateway. A call between two terminals (see Call).

3.41 RAS channel: Unreliable channel used to convey the registration, admissions, bandwidth change, and status messages (following ITU-T Rec. H.225.0) between two H.323 entities.

3.42 reliable channel: A transport connection used for reliable transmission of an information stream from its source to one or more destinations.

3.43 reliable transmission: Transmission of messages from a sender to a receiver using connection-mode data transmission. The transmission service guarantees sequenced, error-free, flow-controlled transmission of messages to the receiver for the duration of the transport connection.

3.44 RTP session: For each participant, the session is defined by a particular pair of destination Transport Addresses (one Network Address plus a TSAP identifier pair for RTP and RTCP). The destination Transport Address pair may be common for all participants, as in the case of IP multicast, or may be different for each, as in the case of individual unicast network addresses. In a multimedia session, the media audio and video are carried in separate RTP sessions with their own RTCP packets. The multiple RTP sessions are distinguished by different Transport Addresses.

3.45 switched circuit network (SCN): A public or private switched telecommunications network such as the GSTN, N-ISDN, or B-ISDN.

NOTE – While B-ISDN is not strictly a switched circuit network, it exhibits some of the characteristics of an SCN through the use of virtual circuits.

3.46 terminal: An H.323 Terminal is an endpoint on the network which provides for real-time, two-way communications with another H.323 terminal, Gateway, or Multipoint Control Unit. This communication consists of control, indications, audio, moving colour video pictures, and/or data between the two terminals. A terminal may provide speech only, speech and data, speech and video, or speech, data and video.

3.47 transport address: The transport layer address of an addressable H.323 entity as defined by the (inter)network protocol suite in use. The Transport Address of an H.323 entity is composed of the Network Address plus the TSAP identifier of the addressable H.323 entity.

3.48 transport connection: An association established by a transport layer between two H.323 entities for the transfer of data. In the context of this Recommendation, a transport connection provides reliable transmission of information.

3.49 trunking gateway: a Gateway that connects two like networks (for example, two SS7 networks or two QSIG networks), in which tunnelling is used to create full transparency and a true tandem function.

3.50 TSAP identifier: The piece of information used to multiplex several transport connections of the same type on a single H.323 entity with all transport connections sharing the same Network Address, (e.g., the port number in a TCP/UDP/IP environment). TSAP identifiers may be (pre)assigned statically by some international authority or may be allocated dynamically during the setup of a call. Dynamically assigned TSAP identifiers are of transient nature, i.e., their values are only valid for the duration of a single call.

3.51 unicast: A process of transmitting messages from one source to one destination.

3.52 unreliable channel: A logical communication path used for unreliable transmission of an information stream from its source to one or more destinations.

3.53 unreliable transmission: Transmission of messages from a sender to one or more receivers by means of connectionless-mode data transmission. The transmission service is *best-effort* delivery of the PDU, meaning that messages transmitted by the sender may be lost, duplicated, or received out of order by (any of) the receiver(s).

3.54 well-known TSAP identifier: A TSAP identifier that has been allocated by an (international) authority that is in charge of the assignment of TSAP identifiers for a particular (inter)networking protocol and the related transport protocols (e.g., the IANA for TCP and UDP port numbers). This identifier is guaranteed to be unique in the context of the respective protocol.

3.55 zone: A Zone (see Figure 3) is the collection of all terminals (Tx), Gateways (GW) and Multipoint Control Units (MCUs) managed by a single Gatekeeper (GK). A Zone has one and only one Gatekeeper. A Zone may be independent of network topology and may be comprised of multiple network segments which are connected using routers (R) or other devices.

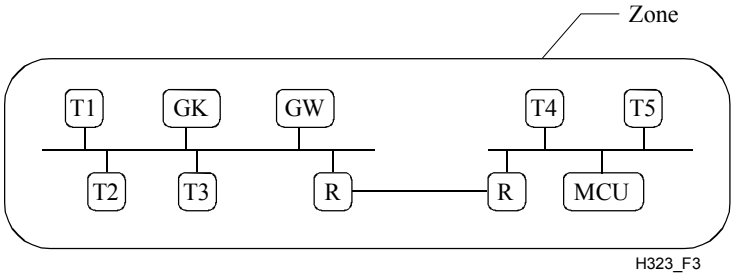


Figure 3/H.323 – Zone

4 Symbols and abbreviations

This Recommendation uses the following abbreviations:

4CIF 4 times CIF

16CIF 16 times CIF

ABNF Augmented Backus-Naur Form

ABR Available Bit Rate

ABT/DT ATM Block Transfer/Delayed Transmission

ABT/IT ATM Block Transfer/Immediate Transmission

ACF Admission Confirmation

AGW Access Gateway

APE Application Protocol Entity

ARJ Admission Reject

ARQ Admission Request

ATC ATM Transfer Capability

ATM Asynchronous Transfer Mode

BAS Bit rate Allocation Signal

BCF Bandwidth Change Confirmation

BCH Bose, Chaudhuri, and Hocquengham

B-HLI	Broadband High Layer Information
B-ISDN	Broadband Integrated Services Digital Network
BRJ	Bandwidth Change Reject
BRQ	Bandwidth Change Request
BTC	Broadband Transfer Capability
CAS	Channel Associated Signalling
CDV	Cell Delay Variation
CED	Called Terminal Identification Tone
CER	Cell Error Ratio
CID	Conference Identifier
CIF	Common Intermediate Format
CLR	Cell Loss Ratio
CMR	Cell Misinsertion Rate
CNG	Calling Tone
CTD	Cell Transfer Delay
DBR	Deterministic Bit Rate
DCF	Disengage Confirmation
DNS	Domain Name System
DRQ	Disengage Request
DSVD	Digital Simultaneous Voice and Data
DTMF	Dual-Tone MultiFrequency
FAS	Facility Associated Signalling
FIR	Full Intra Request
GCC	Generic Conference Control
GCF	Gatekeeper Confirmation
GID	Global Call Identifier
GIT	Generic Identifier Transport
GK	Gatekeeper
GQOS	Guaranteed Quality of Service
GRJ	Gatekeeper Reject
GRQ	Gatekeeper Request
GSTN	General Switched Telephone Network
GW	Gateway
HDLC	High Level Data Link Control
HTTP	Hypertext Transfer Protocol
IACK	Information Acknowledgment
IANA	Internet Assigned Numbers Authority

ID	Identifier
IE	Information Element
IMT	Inter-machine Trunk
INAK	Information Negative Acknowledgment
IP	Internet Protocol
IPX	Internetwork Protocol Exchange
IRQ	Information Request
IRR	Information Request Response
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
LAN	Local Area Network
LCF	Location Confirmation
LRJ	Location Reject
LRQ	Location Request
MC	Multipoint Controller
MCS	Multipoint Communications System
MCU	Multipoint Control Unit
MG	Media Gateway
MGC	Media Gateway Controller
MIME	Multipurpose Internet Mail Extensions
MP	Multipoint Processor
MTU	Maximum Transmission Unit
NACK	Negative Acknowledge
NFAS	Non-facility Associated Signalling
N-ISDN	Narrow-band Integrated Services Digital Network
NNI	Network-to-Network Interface
NSAP	Network Service Access Point
OLC	H.245 **openLogicalChannel** message
PBN	Packet Based Network
PDU	Packet Data Unit
PPP	Point-to-Point Protocol
PRI	Primary Rate Interface
QCIF	Quarter CIF
QOS	Quality of Service
QSIG	Signalling between the Q reference points defined in [45]
RAS	Registration, Admission and Status

RAST	Receive and Send Terminal
RCF	Registration Confirmation
RIP	Request in Progress
RRJ	Registration Reject
RRQ	Registration Request
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SBE	Single Byte Extension
SBR1	Statistical Bit Rate configuration 1
SBR2	Statistical Bit Rate configuration 2
SBR3	Statistical Bit Rate configuration 3
SCI	Service Control Indication
SCM	Selected Communications Mode
SCN	Switched Circuit Network
SCR	Service Control Response
SDL	Specification and Description Language
SECBR	Severely Errored Cell Block Ratio
SPX	Sequential Protocol Exchange
SQCIF	Sub QCIF
SS7	Signalling System No. 7
SSRC	Synchronization Source Identifier
TCP	Transport Control Protocol
TGW	Trunking Gateway
TSAP	Transport layer Service Access Point
UCF	Unregister Confirmation
UDP	User Datagram Protocol
UNI	User-to-Network Interface
URJ	Unregister Reject
URQ	Unregister Request
VC	Virtual Channel

5 Conventions

In this Recommendation, the following conventions are used:

"Shall" indicates a mandatory requirement.

"Should" indicates a suggested but optional course of action.

"May" indicates an optional course of action rather than a recommendation that something take place.

References to clauses, subclauses, annexes and appendices refer to those items within this Recommendation unless another specification is explicitly listed. For example, 1.4 refers to 1.4 of this Recommendation; 6.4/H.245 refers to 6.4 in ITU-T Rec. H.245.

Throughout this Recommendation, the term "network" is used to indicate any packet based network regardless of the underlying physical connection or the geographic scope of the network. This includes Local Area Networks, internetworks, and other packet based networks. The term "Switched Circuit Network" or "SCN" is used explicitly when referring to switched circuit networks such as GSTN and ISDN.

Where items exist on both the packet based network and the SCN, references to the SCN item will be explicit. For example, an MCU is an H.323 MCU on the packet based network, an SCN-MCU is an MCU on the SCN.

This Recommendation describes the use of three different message types: H.245, RAS and H.225.0 call signalling. To distinguish between the different message types, the following convention is followed. H.245 message and parameter names consist of multiple concatenated words highlighted in bold typeface (**maximumDelayJitter**). RAS message names are represented by three letter abbreviations (ARQ). H.225.0 call signalling message names consist of one or two words with the first letters capitalized (Call Proceeding).

6 System description

This Recommendation describes the elements of the H.323 components. These are Terminals, Gateways, Gatekeepers, MCs and MCUs. These components communicate through the transmission of Information Streams. The characteristics of these components are described in this clause.

6.1 Information streams

Visual telephone components communicate through the transmission of Information Streams. These Information Streams are classified into video, audio, data, communications control and call control as follows.

Audio signals contain digitized and coded speech. In order to reduce the average bit rate of audio signals, voice activation may be provided. The audio signal is accompanied by an audio control signal.

Video signals contain digitized and coded motion video. Video is transmitted at a rate no greater than that selected as a result of the capability exchange. The video signal is accompanied by a video control signal.

Data signals include still pictures, facsimile, documents, computer files and other data streams.

Communications control signals pass control data between remote-like functional elements and are used for capability exchange, opening and closing logical channels, mode control and other functions that are part of communications control.

Call control signals are used for call establishment, disconnect and other call control functions.

The information streams described above are formatted and sent to the network interface as described in ITU-T Rec. H.225.0.

6.2 Terminal characteristics

An example of an H.323 terminal is shown in Figure 4. The diagram shows the user equipment interfaces, video codec, audio codec, telematic equipment, H.225.0 layer, system control functions and the interface to the packet based network. All H.323 terminals shall have a System Control

Unit, H.225.0 layer, Network Interface and an Audio Codec Unit. The Video Codec Unit and User Data Applications are optional.

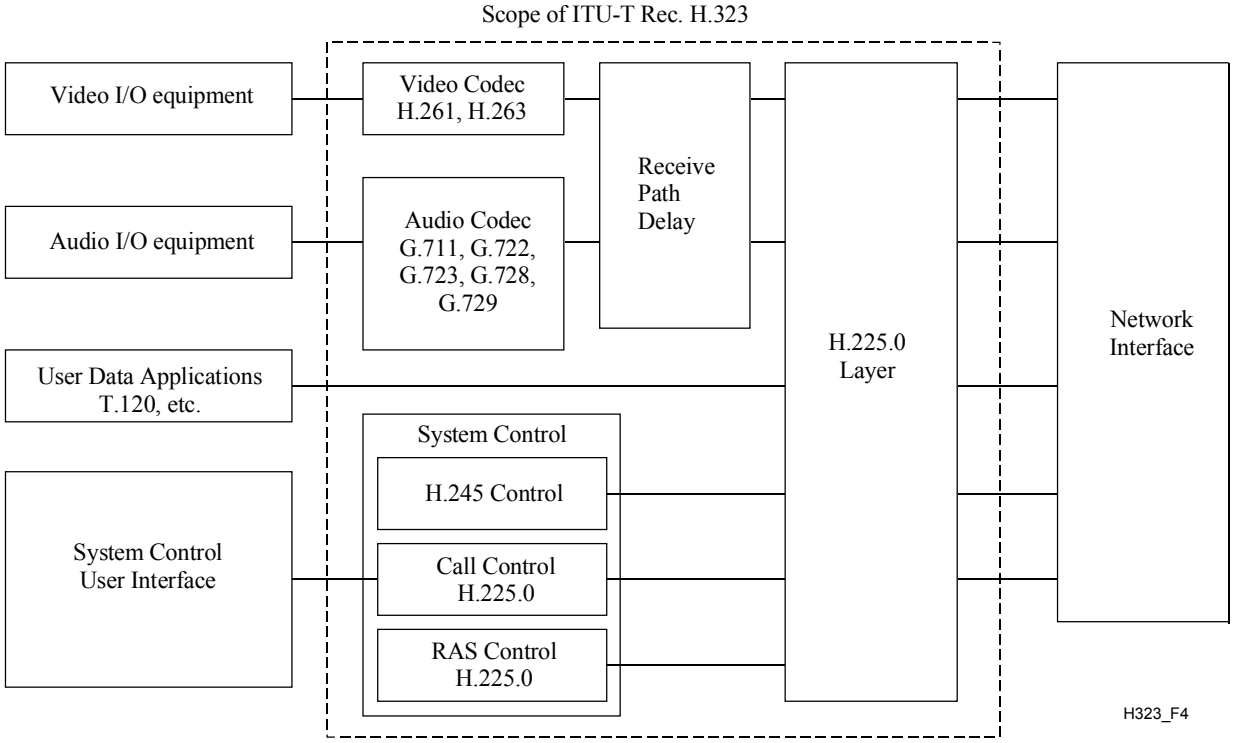


Figure 4/H.323 – H.323 terminal equipment

6.2.1 Terminal elements outside the scope of this Recommendation

The following elements are not within the scope of this Recommendation and are therefore not defined within this Recommendation:

- Attached audio devices providing voice activation sensing, microphone and loudspeaker, telephone instrument or equivalent, multiple microphones mixers, and acoustic echo cancellation.
- Attached video equipment providing cameras and monitors, and their control and selection, video processing to improve compression or provide split screen functions.
- Data applications and associated user interfaces which use T.120 or other data services over the data channel.
- Attached Network Interface, which provides the interface to the packet based network, supporting appropriate signalling and voltage levels, in accordance with national and international standards.
- Human user system control, user interface and operation.

6.2.2 Terminal elements within the scope of this Recommendation

The following elements are within the scope of this Recommendation and are therefore subject to standardization and are defined within this Recommendation:

- The Video Codec (H.261, etc.) encodes the video from the video source (i.e., camera) for transmission and decodes the received video code which is output to a video display.
- The Audio Codec (G.711, etc.) encodes the audio signal from the microphone for transmission and decodes the received audio code which is output to the loudspeaker.

- The Data Channel supports telematic applications such as electronic whiteboards, still image transfer, file exchange, database access, audiographics conferencing, etc. The standardized data application for real-time audiographics conferencing is ITU-T Rec. T.120. Other applications and protocols may also be used via H.245 negotiation as specified in 6.2.7.
- The System Control Unit (H.245, H.225.0) provides signalling for proper operation of the H.323 terminal. It provides for call control, capability exchange, signalling of commands and indications, and messages to open and fully describe the content of logical channels.
- H.225.0 Layer (H.225.0) formats the transmitted video, audio, data and control streams into messages for output to the network interface and retrieves the received video, audio, data and control streams from messages which have been input from the network interface. In addition, it performs logical framing, sequence numbering, error detection and error correction as appropriate to each media type.

6.2.3 Packet based network interface

The packet based network interface is implementation-specific and is outside the scope of this Recommendation. However, the network interface shall provide the services described in ITU-T Rec. H.225.0. This includes the following: Reliable (e.g., TCP, SPX) end-to-end service is mandatory for the H.245 Control Channel, the Data Channels, and the Call Signalling Channel. Unreliable (e.g., UDP, IPX) end-to-end service is mandatory for the Audio Channels, the Video Channels and the RAS Channel. These services may be duplex or simplex, unicast or multicast depending on the application, the capabilities of the terminals, and the configuration of the network.

6.2.4 Video codec

The video codec is optional. If video capability is provided, it shall be provided according to the requirements of this Recommendation. All H.323 terminals providing video communications shall be capable of encoding and decoding video according to H.261 QCIF. Optionally, a terminal may also be capable of encoding and decoding video according to the other modes of H.261 or H.263. If a terminal supports H.263 with CIF or higher resolution, it shall also support H.261 CIF. All terminals which support H.263 shall support H.263 QCIF. The H.261 and H.263 codecs, on the network, shall be used without BCH error correction and without error correction framing.

Other video codecs and other picture formats, may also be used via H.245 negotiation. More than one video channel may be transmitted and/or received, as negotiated via the H.245 Control Channel. The H.323 terminal may optionally send more than one video channel at the same time, for example, to convey the speaker and a second video source. The H.323 terminal may optionally receive more than one video channel at the same time, for example, to display multiple participants in a distributed multipoint conference.

The video bit rate, picture format and algorithm options that can be accepted by the decoder are defined during the capability exchange using H.245. The encoder is free to transmit anything that is within the decoder capability set. The decoder should have the possibility to generate requests via H.245 for a certain mode, but the encoder is allowed to simply ignore these requests if they are not mandatory modes. Decoders which indicate capability for a particular algorithm option shall also be capable of accepting video bit streams which do not make use of that option.

H.323 terminals shall be capable of operating in asymmetric video bit rates, frame rates, and, if more than one picture resolution is supported, picture resolutions. For example, this will allow a CIF capable terminal to transmit QCIF while receiving CIF pictures.

When each video logical channel is opened, the selected operating mode to be used on that channel is signalled to the receiver in the H.245 **openLogicalChannel** message. The header within the video logical channel indicates which mode is actually used for each picture, within the stated decoder capability.

The video stream is formatted as described in ITU-T Rec. H.225.0.

6.2.4.1 Terminal-based continuous presence

H.323 terminals may receive more than one video channel, particularly for multipoint conferencing. In these cases, the H.323 terminal may need to perform a video mixing or switching function in order to present the video signal to the user. This function may include presenting the video from more than one terminal to the user. The H.323 terminal shall use H.245 simultaneous capabilities to indicate how many simultaneous video streams it is capable of decoding. The simultaneous capability of one terminal should not limit the number of video streams which are multicast in a conference (this choice is made by the MC).

6.2.5 Audio codec

All H.323 terminals shall have an audio codec. All H.323 terminals shall be capable of encoding and decoding speech according to ITU-T Rec. G.711. All terminals shall be capable of transmitting and receiving A-law and μ-law. A terminal may optionally be capable of encoding and decoding speech using other audio codecs which can be signalled via H.245 negotiation. The audio algorithm used by the encoder shall be derived during the capability exchange using H.245. The H.323 terminal should be capable of asymmetric operation for all audio capabilities it has declared within the same capability set, e.g., it should be able to send G.711 and receive G.728 if it is capable of both.

If G.723.1 audio is provided, the audio codec shall be capable of encoding and decoding according to both the 5.3 kbit/s mode and the 6.3 kbit/s mode.

The audio stream is formatted as described in ITU-T Rec. H.225.0.

The H.323 terminal may optionally send more than one audio channel at the same time, for example, to allow two languages to be conveyed.

Audio packets should be delivered to the transport layer periodically at an interval determined by the audio codec Recommendation in use (audio frame interval). The delivery of each audio packet shall occur no later than 5 ms after a whole multiple of the audio frame interval, measured from delivery of the first audio frame (audio delay jitter). Audio coders capable of further limiting their audio delay jitter may so signal using the H.245 **maximumDelayJitter** parameter of the **h2250Capability** structure contained within a terminal capability set message, so that receivers may optionally reduce their jitter delay buffers. This is not the same as the RTCP interarrival jitter field.

NOTE – The testing point for the maximum delay jitter is at the input to network transport layer. Network stack, network, driver, and interface card jitter are not included.

6.2.5.1 Audio mixing

H.323 terminals may receive more than one audio channel, particularly for multipoint conferencing. In these cases, the H.323 terminal may need to perform an audio mixing function in order to present a composite audio signal to the user. The H.323 terminal shall use H.245 simultaneous capabilities to indicate how many simultaneous audio streams it is capable of decoding. The simultaneous capability of one terminal should not limit the number of audio streams which are multicast in a conference.

6.2.5.2 Maximum audio-video transmit skew

To allow H.323 terminals to appropriately set their receive buffer(s) size, H.323 terminals shall transmit the **h2250MaximumSkewIndication** message to indicate the maximum skew between the audio and video signals as delivered to the network transport. **h2250MaximumSkewIndication** shall be sent for each pair of associated audio and video logical channels. This is not required for

audio only or hybrid conferences. Lip synchronization, if desired, shall be achieved via use of time-stamps.

6.2.5.3 Low bit rate operation

G.711 audio cannot be used in an H.323 conference being carried over low bit rate (< 56 kbit/s) links or segments. An endpoint used for multimedia communications over such low bit rate links or segments should have an audio codec capable of encoding and decoding speech according to ITU-T Rec. G.723.1. An endpoint used for audio-only communications over such low bit rate links or segments should have an audio codec capable of encoding and decoding speech according to ITU-T Rec. G.729. An endpoint may support several audio codecs in order to provide the widest possible interoperability with those endpoints which only support one low bitrate audio codec. The endpoint shall indicate in the H.245 Capability Exchange procedures at the beginning of each call the capability to receive audio according to the available audio Recommendations which can be supported within the known bit rate limitations of the connection. An endpoint which does not have this low bit rate audio capability may not be able to operate when the end-to-end connection contains one or more low bit rate segments.

The endpoint shall also comply with the requirement of 6.2.5 to be capable of encoding and decoding speech according to ITU-T Rec. G.711. However, the endpoint need not indicate this capability if it is sure that it is communicating through a low bit rate segment. If an endpoint is unaware of the presence, in the end-to-end connection, of any links or segments with insufficient capacity to support G.711 audio (along with other intended media streams, if any), then the endpoint shall declare the capability to receive audio according to ITU-T Rec. G.711.

6.2.6 Receive path delay

Receive path delay includes delay added to a media stream in order to maintain synchronization and to account for network packet arrival jitter. Media streams may optionally be delayed in the receiver processing path to maintain synchronization with other media streams. Further, a media stream may optionally be delayed to allow for network delays which cause packet arrival jitter. An H.323 terminal shall not add delay for this purpose in its transmitting media path.

Intermediate processing points such as MCUs or Gateways may alter the video and audio time tag information and shall transmit appropriately modified audio and video time tags and sequence numbers, reflecting their transmitted signals. Receiving endpoints may add appropriate delay in the audio path to achieve lip synchronization.

6.2.7 Data channel

One or more data channels are optional. The data channel may be unidirectional or bidirectional depending on the requirements of the data application.

ITU-T Rec. T.120 is the default basis of data interoperability between an H.323 terminal and other H.323, H.324, H.320 or H.310 terminals. Where any optional data application is implemented using one or more of the ITU-T Recommendations, which can be negotiated via H.245, the equivalent T.120 application, if any, shall be one of those provided.

Note that non-standard data applications (**dataApplicationCapability.application = non-standard** application) and Transparent User Data (**dataApplicationCapability.application = userData** application, **dataProtocolCapability = transparent**) may be used whether the equivalent T.120 application is provided or not.

T.120 capability shall be signalled using **dataApplicationCapability.application = t120** application, **dataProtocolCapability = separateLANStack**.

Within the **MediaDistributionCapability**, the **distributedData** structure shall be used if multicast T.120 is available and/or the **centralizedData** structure if unicast T.120 is available. Any node that supports the T.120 data capability shall support the standard T.123 unicast stack.

In the **openLogicalChannel** message, the **distribution** choice of the **NetworkAccessParameters** structure is set to **unicast** if T.123 is to be used or **multicast** if Annex A/T.125 is to be used. The **networkAddress** choice is set to **localAreaAddress**, which should always be **unicastAddress**. Within the **iPAddress** sequence, the **network** field is set to the binary IP address and the **tsapIdentifier** is set to the dynamic port on which the T.120 stack will be calling or listening.

The Data channel is formatted as described in ITU-T Rec. H.225.0.

6.2.7.1 T.120 data channels

The T.120 connection is established during an H.323 call as an inherent part of the call. Procedures for establishing the T.120 connection prior to the H.323 connection are for further study.

The normal call setup procedures of 8.1 are followed. After the capability exchange takes place, a bidirectional logical channel shall be opened for the T.120 connection according to the normal H.245 procedures indicating that a new connection shall be created as described below.

The opening of a bidirectional logical channel for T.120 may be initiated by either entity sending an **openLogicalChannel** message and then following the bidirectional logical channel procedures of ITU-T Rec. H.245.

To actually open the logical channel, the initiating entity shall send an **openLogicalChannel** message indicating that a T.120 data channel is to be opened in the **forwardLogicalChannelParameters** as well as in the **reverseLogicalChannelParameters**. The initiator shall include a Transport Address in the **openLogicalChannel** message. The peer endpoint may choose to ignore the Transport Address. An endpoint may use a dynamic port number for the T.120 Transport Address instead of using port 1503 as specified in ITU-T Rec. T.123. If the peer (the responder) accepts this logical channel, it shall confirm the opening of the logical channel using **openLogicalChannelAck**. In the **openLogicalChannelAck**, the responder shall include a Transport Address even if it expects the initiator to originate the T.120 call. In all cases, the Transport Address for the T.120 connection shall be carried in the **separateStack** parameter and shall remain valid for the duration of the logical channel.

In the **openLogicalChannel** message, the **t120SetupProcedure** choice of the **NetworkAccessParameters** structure can optionally be set to indicate to the responder how the initiator would like to establish the T.120 call. The responder is free to override this preference. **originateCall** indicates that the initiator would like the responder to place the call. **waitForCall** indicates that the initiator would like the responder to receive the call. **issueQuery** is not used when indicating a preference.

In the **openLogicalChannelAck** message, the **t120SetupProcedure** choice of the **NetworkAccessParameters** structure should be set to indicate to the initiator how the T.120 call will be established. If neither endpoint has a preference, the T.120 call should be established in the same direction as the H.323 call. **originateCall** tells the initiator to place the call. **waitForCall** tells the initiator that it will receive the call. Whoever originates the call will issue either a join request or an invite request, depending on which endpoint won master/slave determination (the master is always hierarchically higher in the T.120 conference). **issueQuery** can be used by a Gateway to tell the initiator that it must originate the call and issue a query request to the remote endpoint. It must then set up the T.120 conference in accordance with the contents of the query response (as described in ITU-T Rec. T.124).

When possible, the T.120 call should be established in the same direction as the H.323 call. The OLC initiator should not indicate a preference unless there is a need to modify this default behaviour. When the initiator indicates a preference, the responder should not override it unless

necessary. When no preference is indicated, the responder should specify the default unless there is a need to do otherwise.

In both the **openLogicalChannel** and the **openLogicalChannelAck** messages, the **associateConference** parameter shall be set to FALSE.

ITU-T Rec. T.120 shall follow the procedures of ITU-T Rec. T.123 for the protocol stack indicated in the **dataProtocolCapability** except that the Transport Addresses as described above shall be employed for connection setup.

If an endpoint is the Active MC or master in a conference, which includes T.120, it should also be in control of the T.120 top provider node.

If an endpoint intends to create a conference, which includes audio and/or video plus T.120 data, then the H.245 Control Channel shall be established before the T.120 connection is made. This applies to conference create, join, and invite and the actions of an MC. The H.323 call setup procedures shall be used to establish the Active MC (if any), before a T.120 connection is made.

In order to establish a T.120 connection using a GCC-Join request, endpoints are required to know the T.120 conference name. If an alias exists which represents an H.323 conference name (**conferenceAlias**), then the same text which is used for the conference alias should be used as the text portion of the T.120 conference name. Likewise, the H.323 CID should be used as the numeric T.120 conference name as follows. Each byte of the H.323 CID is converted into a series of three ASCII characters, which represent the decimal value of the byte being converted. Note that this requires the value of some CID bytes to be converted such that "0" characters are used for padding. The result will be a string of 48 ASCII characters.

A T.120 MP may be queried for a list of existing conferences. The H.323 CID may be available by converting from the T.120 Numeric Conference name back into the 16-byte octet string. Likewise, the Text Conference name may be used as the H.323 conference alias. Note that a T.124 Conference Query may happen out-of-band from H.323 and prior to an endpoint setting up an H.323 call.

The termination of the associated T.120 conference does not imply the termination of the H.323 call. In other words, closing the T.120 channel shall only affect the Data stream of an H.323 call and shall not affect any other part of the H.323 call. By contrast, when an H.323 call or conference is terminated, then the associated T.120 conference shall also be terminated.

NOTE – The T.120 operation after completion of the connection setup is beyond the scope of this Recommendation.

6.2.7.2 Remote device control

H.323 endpoints may support remote device control through the H.282 protocol. The H.282 protocol shall be supported in an H.245 logical channel according to ITU-T Rec. H.283. ITU-T Rec. H.283 describes logical channel transport for the H.282 protocol in an H.323 conference.

ITU-T Rec. H.282 may also be used by T.120 systems and carried in a T.120 APE. Optionally H.323 systems may also support remote device control using ITU-T Rec. H.282 over T.120. However, this is an option and an H.323 system that supports H.282 shall support it with ITU-T Rec. H.283.

If both H.282 with H.283 and H.282 with T.120 are supported, then both may be used. Coordination of the two lower layer protocols under H.282 is a local matter. However, H.283 shall always be active to account for possible late joining nodes that support H.282 over H.283 but not H.282 over T.120.

6.2.8 H.245 control function

The H.245 Control Function uses the H.245 Control Channel to carry end-to-end control messages governing operation of the H.323 entity, including capabilities exchange, opening and closing of logical channels, mode preference requests, flow control messages, and general commands and indications.

H.245 signalling is established between two endpoints, an endpoint and an MC, or an endpoint and a Gatekeeper. The endpoint shall establish exactly one H.245 Control Channel for each call that the endpoint is participating in. This channel shall use the messages and procedures of ITU-T Rec. H.245. Note that a terminal, MCU, Gateway, or Gatekeeper may support many calls and thus many H.245 Control Channels. The H.245 Control Channel shall be carried on logical channel 0. Logical channel 0 shall be considered to be permanently open from the establishment of the H.245 Control Channel until the termination of this channel. The normal procedures for opening and closing logical channels shall not apply to the H.245 Control Channel.

ITU-T Rec. H.245 specifies a number of independent protocol entities which support endpoint-to-endpoint signalling. A protocol entity is specified by its syntax (messages), semantics, and a set of procedures which specify the exchange of messages and the interaction with the user. H.323 endpoints shall support the syntax, semantics, and procedures of the following protocol entities:

- Master/slave determination.
- Capability Exchange.
- Logical Channel Signalling.
- Bidirectional Logical Channel Signalling.
- Close Logical Channel Signalling.
- Mode Request.
- Round Trip Delay Determination.
- Maintenance Loop Signalling.

General commands and indications shall be chosen from the message set contained in ITU-T Rec. H.245. In addition, other command and indication signals may be sent which have been specifically defined to be transferred in-band within video, audio or data streams (see the appropriate Recommendation to determine if such signals have been defined).

H.245 messages fall into four categories: Request, Response, Command and Indication. Request and Response messages are used by the protocol entities. Request messages require a specific action by the receiver, including an immediate response. Response messages respond to a corresponding request. Command messages require a specific action, but do not require a response. Indication messages are informative only and do not require any action or response. H.323 terminals shall respond to all H.245 commands and requests as specified in Annex A and shall transmit indications reflecting the state of the terminal.

H.323 terminals shall be capable of parsing all H.245 **multimediaSystemControlMessage** messages and shall send and receive all messages needed to implement required functions and those optional functions which are supported by the terminal. Annex A contains a table showing which H.245 messages are mandatory, optional, or forbidden for H.323 terminals. H.323 terminals shall send the **functionNotSupported** message in response to any unrecognized request, response, or command messages that it receives.

An H.245 indication, **userInputIndication**, is available for transport of user input alphanumeric characters from a keypad or keyboard, equivalent to the DTMF signals used in analogue telephony or SBE number messages in ITU-T Rec. H.230. This may be used to manually operate remote equipment such as voice mail or video mail systems, menu-driven information services, etc.

H.323 terminals shall support the transmission of user input characters 0-9, "*", and "#". Transmission of other characters is optional.

Three H.245 request messages conflict with RTCP control packets. The H.245 **videoFastUpdatePicture**, **videoFastUpdateGOB** and **videoFastUpdateMB** requests should be used instead of the RTCP control packets Full Intra Request (FIR) and Negative Acknowledgement (NACK). The ability to accept FIR and NACK is signalled during the H.245 capability exchange.

6.2.8.1 Capabilities exchange

Capabilities exchange shall follow the procedures of ITU-T Rec. H.245, which provides for separate receive and transmit capabilities, as well as a method by which the terminal may describe its ability to operate in various combinations of modes simultaneously.

Receive capabilities describe the terminal's ability to receive and process incoming information streams. Transmitters shall limit the content of their transmitted information to that which the receiver has indicated it is capable of receiving. The absence of a receive capability indicates that the terminal cannot receive (is a transmitter only).

Transmit capabilities describe the terminal's ability to transmit information streams. Transmit capabilities serve to offer receivers a choice of possible modes of operation, so that the receiver may request the mode which it prefers to receive. The absence of a transmit capability indicates that the terminal is not offering a choice of preferred modes to the receiver (but it may still transmit anything within the capability of the receiver).

Receive-and-Transmit capabilities describe the terminal's ability to receive and transmit information streams when these capabilities are not independent and are required to be the same in both directions. For example, an endpoint might support only symmetrical codec operation for its codecs (G.711 both ways, or G.729 both ways, but not G.711 one way and G.729 the other way). A slave should reorder its codec preference in the same order as the master, e.g., if the slave's preference is {G.729, G.711} and the master's preference is {G.711, G.729}, the slave should reorder its preference to {G.711, G.729}. If the terminal capability set has already proceeded, it should consider its preferences as reordered when proceeding to opening logical channels.

The transmitting terminal assigns each individual mode the terminal is capable of operating in a number in a **capabilityTable**. For example, G.723.1 audio, G.728 audio, and CIF H.263 video would each be assigned separate numbers.

These capability numbers are grouped into **alternativeCapabilitySet** structures. Each **alternativeCapabilitySet** indicates that the terminal is capable of operating in exactly one mode listed in the set. For example, an **alternativeCapabilitySet** listing {G.711, G.723.1, G.728} means that the terminal can operate in any one of those audio modes, but not more than one.

These **alternativeCapabilitySet** structures are grouped into **simultaneousCapabilities** structures. Each **simultaneousCapabilities** structure indicates a set of modes the terminal is capable of using simultaneously. For example, a **simultaneousCapabilities** structure containing the two **alternativeCapabilitySet** structures {H.261, H.263} and {G.711, G.723.1, G.728} means that the terminal can operate either of the video codecs simultaneously with any one of the audio codecs. The **simultaneousCapabilities** set { {H.261}, {H.261, H.263}, {G.711, G.723.1, G.728} } means the terminal can operate two video channels and one audio channel simultaneously: one video channel per H.261, another video channel per either H.261 or H.263, and one audio channel per either G.711, G.723.1, or G.728.

When symmetrical codec operation is used (i.e., when the **receiveAndTransmitVideoCapability** or **receiveAndTransmitAudioCapability** are used), the master may reject an **openLogicalChannel** request from the slave if the master requires the user of symmetrical codecs and the proposed channel is not symmetrical. These conflict resolution procedures are described in C.4.1.3/H.245. The reason field in the **openLogicalChannelReject** shall be **masterSlaveConflict**.

NOTE 1 – The master may send a **requestMode** to the slave with the proper codec before sending the **openLogicalChannelReject** to explicitly request a specific codec.

NOTE 2 – The actual capabilities stored in the **capabilityTable** are often more complex than presented here. For example, each H.263 capability indicates details including the ability to support various picture formats at given minimum picture intervals and the ability to use optional coding modes. For a complete description, see ITU-T Rec. H.245.

The terminal's total capabilities are described by a set of **capabilityDescriptor** structures, each of which is a single **simultaneousCapabilities** structure and a **capabilityDescriptorNumber**. By sending more than one **capabilityDescriptor**, the terminal may signal dependencies between operating modes by describing different sets of modes which it can simultaneously use. For example, a terminal issuing two **capabilityDescriptor** structures, one { {H.261, H.263}, {G.711, G.723.1, G.728} } as in the previous example, and the other { {H.262}, {G.711} }, means the terminal can also operate the H.262 video codec, but only with the low-complexity G.711 audio codec.

Terminals may dynamically add capabilities during a communication session by issuing additional **capabilityDescriptor** structures or remove capabilities by sending revised **capabilityDescriptor** structures. All H.323 terminals shall transmit at least one **capabilityDescriptor** structure.

Non-standard capabilities and control messages may be issued using the **nonStandardParameter** structure defined in ITU-T Rec. H.245. Note that while the meaning of non-standard messages is defined by individual organizations, equipment built by any manufacturer may signal any non-standard message, if the meaning is known.

Terminals may reissue capability sets at any time, according to the procedures of ITU-T Rec. H.245.

6.2.8.2 Logical channel signalling

Each logical channel carries information from a transmitter to one or more receivers and is identified by a logical channel number which is unique for each direction of transmission.

Logical channels are opened and closed using the **openLogicalChannel** and **closeLogicalChannel** messages and procedures of ITU-T Rec. H.245. When a logical channel is opened, the **openLogicalChannel** message fully describes the content of the logical channel, including media type, algorithm in use, any options, and all other information needed for the receiver to interpret the content of the logical channel. Logical channels may be closed when no longer needed. Open logical channels may be inactive, if the information source has nothing to send.

Most logical channels in this Recommendation are unidirectional, so asymmetrical operation, in which the number and type of information streams is different in each direction of transmission, is allowed. However, if a receiver is capable only of certain symmetrical modes of operation, it may send a receive capability set that reflects its limitations, except where noted elsewhere in this Recommendation. Terminals may also be capable of using a particular mode in only one direction of transmission. Certain media types, including data protocols such as T.120, inherently require a bidirectional channel for their operation. In such cases a single bidirectional logical channel may be opened using the bidirectional channel opening procedures of ITU-T Rec. H.245.

Logical channels shall be opened using the following procedure:

The initiating terminal shall send an **openLogicalChannel** message as described in ITU-T Rec. H.245. If the logical channel is to carry a media type using RTP (audio or video), the **openLogicalChannel** message shall include the **mediaControlChannel** parameter containing the Transport Address for the reverse RTCP channel.

The responding terminal shall respond with an **openLogicalChannelAck** message as described in ITU-T Rec. H.245. If the logical channel is to carry a media type using RTP, the **openLogicalChannelAck** message shall include both the **mediaChannel** parameter containing the

RTP Transport Address for the media channel and the **mediaControlChannel** parameter containing the Transport Address for the forward RTCP channel.

Media types (such as T.120 data) which do not use RTP/RTCP shall omit the **mediaControlChannel** parameters.

If a corresponding reverse channel is opened for a given existing RTP session (identified by the RTP **sessionID**), the **mediaControlChannel** Transport Addresses exchanged by the **openLogicalChannel** process shall be identical to those used for the forward channel. **sessionID** values of 1, 2 and 3 are pre-assigned for primary audio, video and data sessions, respectively. Even the slave endpoint can open logical channels for these primary sessions without negotiating the **sessionID** value with the master endpoint. The master endpoint can open any additional session with a particular **sessionID** value greater than 3. The slave endpoint can open a corresponding session with the given **sessionID**. Otherwise, the slave endpoint can open additional sessions with **sessionID**=0 in the **openLogicalChannel** message, but it shall acquire the actual **sessionID** value from the master endpoint's **openLogicalChannelAck** message. Should a collision occur where both ends attempt to establish conflicting RTP sessions at the same time, the master endpoint shall reject the conflicting attempt as described in ITU-T Rec. H.245. The rejected **openLogicalChannel** attempt may then be retried at a later time.

Unless specified otherwise for a particular data type, reliable data channels are bidirectional channels and, as such, shall contain both the **forwardLogicalChannelParameters** and **reverseLogicalChannelParameters** elements without the **mediaChannel** elements. The endpoint accepting the channel shall return the **mediaChannel** element in the **reverseLogicalChannelParameters** element and be prepared to accept the reliable connection from the requesting endpoint prior to returning the **OpenLogicalChannelAck** message.

An endpoint that accepts a bidirectional reliable channel shall be prepared to accept a reliable connection from the requesting endpoint prior to returning the **OpenLogicalChannelAck** message.

6.2.8.3 Mode preferences

Receivers may request transmitters to send a particular mode using the H.245 **requestMode** message, which describes the desired mode. Transmitters should comply if possible.

An endpoint receiving the **multipointModeCommand** from the MC shall then comply with all **requestMode** commands, if they are within its capability set. Note that in a decentralized conference, as in a centralized conference, all terminal **requestMode** commands are directed to the MC. The MC may grant the request or not; the basis for this decision is left to the manufacturer.

6.2.8.4 Master-slave determination

The H.245 Master-slave determination procedures are used to resolve conflicts between two endpoints which can both be the MC for a conference or between two endpoints which are attempting to open a bidirectional channel. In this procedure, two endpoints exchange random numbers in the H.245 **masterSlaveDetermination** message, to determine the master and slave endpoints. H.323 endpoints shall be capable of operating in both master and slave modes. The endpoints shall set **terminalType** to the value specified in Table 1 below and set **statusDeterminationNumber** to a random number in the range 0 to $2^{24} - 1$. Only one random number shall be chosen by the endpoint for each call, except in the case of identical random numbers, as described in ITU-T Rec. H.245.

Table 1/H.323 – H.323 terminal types for H.245 master-slave determination

TerminalType value table	H.323 entity			
Feature set	**Terminal**	**Gateway**	**Gatekeeper**	**MCU**
Entity with No MC	50	60	NA	NA
Entity contains an MC but no MP	70	80	120	160
Entity contains MC with data MP	NA	90	130	170
Entity contains MC with data and audio MP	NA	100	140	180
Entity contains MC with data, audio and video MP	NA	110	150	190

The Active MC in a conference shall use a value of 240.

If a single H.323 entity can take part in multiple calls, then the value used for **terminalType** in the master-slave determination process shall be based on the features that the H.323 entity has assigned or will assign to the call in which it is being signalled.

An MC that is already acting as an MC shall always remain the Active MC. Therefore, once an MC has been selected as the Active MC in a conference, it shall use the Active MC value for all subsequent connections to the conference.

If no MC is active and the entities are of the same type, then the H.323 entity with the highest feature set (as shown in Table 1) shall win the master-slave determination. If no MC is active and the entities are of different types, then an MC that is located in an MCU shall have priority over an MC that is located in a Gatekeeper, which shall have priority over an MC that is located in a Gateway, which in turn shall have priority over an MC located in a terminal.

If an H.323 entity can be associated with two or more of the classifications shown in Table 1, then it should use the highest value for which it qualifies.

6.2.8.5 Timer and counter values

All timers defined in ITU-T Rec. H.245 should have periods of at least as long as the maximum data delivery time allowed by the data link layer carrying the H.245 Control Channel, including any retransmissions.

The H.245 retry counter N100 should be at least 3.

Procedures relating to H.245 protocol error handling are covered in 8.6.

6.2.8.6 Multiplexed stream transmission over a single logical channel

Multiple media streams may be multiplexed over a single logical channel. A multiplexed stream is a stream that contains multiple media streams using the multiplexing protocols H.222.0 [46] or H.223 [47] and transmitted as a series of RTP packets. By using these multiplexing protocols, an H.323 endpoint may take advantage of certain benefits, such as more efficient bandwidth usage, precise media synchronization, or low delay in multimedia transmission.

There are two ways to control the configuration of a multiplexed stream. One way is to transmit the H.245 messages inside the RTP packets of multiplexed streams. In this case, H.323 endpoints first open bidirectional logical channel for the multiplexed stream transmission using H.245 logical channel signalling procedures as normal RTP media streams. Then the control for multiplexed stream is done using the H.245 messages inside RTP packets of the target multiplexed stream. The control of the multiplexed stream includes capability exchange about the media codecs available for this multiplexed stream, multiplex table exchange, and open/close logical channels. The logical channel numbers over a multiplexed stream are independent from that of the other multiplexed streams or that of H.245 logical channels.

The other way to control the configuration of a multiplexed stream is to control the logical channels on the multiplexed stream in the same way as non-multiplexed logical channels, i.e., the H.245 messages for the multiplexed stream are transmitted in the same manner as other H.245 messages. In this case, an H.323 endpoint opens a unidirectional or bidirectional logical channel for the multiplexed stream transmission using H.245 logical channel signalling procedures as normal RTP media streams. Then the logical channels over the multiplexed stream are opened using logical channel signalling with parameters of the multiplex protocol configuration and logical channel number of the multiplex stream over which the new logical channel is being opened.

6.2.8.6.1 Capability exchange related to multiplexed stream

H.323 terminals supporting multiplexed streams indicate this capability by including a **MultiplexedStreamCapability** as a part of terminal capability. The parameter **controlOnMuxStream** in **MultiplexedStreamCapability** indicates whether the terminal supports the control of the multiplexed stream using H.245 messages or on the RTP packets of the multiplexed stream itself. If **controlOnMuxStream** is TRUE, the capability of codecs on the multiplexed stream may be set to **capabilityOnMuxStream**. If **capabilityOnMuxStream** does not exist, the terminal shall perform the capability exchange procedure by sending the H.245 messages in the RTP packets over the multiplexed stream once the logical channel for the multiplexed stream is opened. If **controlOnMuxStream** is FALSE, the capability of codecs on the multiplexed stream shall be set to **capabilityOnMuxStream**.

6.2.8.6.2 Logical channel signalling to transport multiplexed stream

The logical channel for the multiplexed stream is opened by sending an **openLogicalChannel** message with the **dataType** of a **MultiplexedStreamCapability** type and **multiplexParameters** of **h2250LogicalChannelParameters**. If the **controlOnMuxStream** in **MultiplexedStreamCapability** is TRUE, the logical channel shall be opened as bidirectional logical channel, i.e., **reverseLogicalChannelParameters** shall be set. Otherwise, the logical channel may be opened as unidirectional logical channel. Note that if the logical channel is opened as unidirectional, some of the multiplex protocol function may not be used, e.g., AL3 of H.223 cannot be used over unidirectional logical channels.

Terminal shall not open more than one logical channel with **multiplexFormat** of **h223Capability** and **controlOnMuxStream** of FALSE.

6.2.8.6.3 Logical channel signalling to transport media stream over multiplexed stream

The logical channel over multiplexed stream is opened by sending an **openLogicalChannel** message with the appropriate **dataType** for the media being delivered and **multiplexParameters** of the appropriate multiplex protocol in use (i.e., **h223logicalChannelParameters**). In case of H.223, multiplex table signalling procedure shall also be performed before or after this logical channel signalling as described in 6.4.2/H.324.

If **controlOnMuxStream** is TRUE, these H.245 messages are delivered within the RTP packets for the multiplexed stream over which the new logical channel is opened. In case of H.223, H.245 **MultimediaSystemControlMessage** messages are protected with the Simple Retransmission Protocol (SRP) and delivered over logical channel number 0 of the multiplexed stream, as described in 6.5.4/H.324.

If **controlOnMuxStream** is FALSE, these H.245 messages are delivered over H.245 Control Channel as usual. In case of H.222.0, **resourceID** of **h2220LogicalChannelParameters** are set to the logical channel number for the multiplexed stream over which this new logical channel is being opened. Note that in case of H.223, no such signalling is needed due to the fact that no more than one logical channel exists.

Logical channels over the multiplexed stream are closed by sending **closeLogicalChannel** messages, which are transmitted in the same way as the **openLogicalChannel** messages for the channel.

6.2.8.6.4 Logical channel signalling to close multiplexed stream

The logical channel for the multiplexed stream which is opened with **controlOnMuxStream** set to TRUE may be closed at any time by a **closeLogicalChannel** message. The logical channel for the multiplexed stream which is opened with **controlOnMuxStream** set to FALSE shall be closed only after all logical channels on the multiplexed stream is closed.

6.2.9 RAS signalling function

The RAS signalling function uses H.225.0 messages to perform registration, admissions, bandwidth changes, status, and disengage procedures between endpoints and Gatekeepers. The RAS Signalling Channel is independent from the Call Signalling Channel and the H.245 Control Channel. H.245 open logical channel procedures are not used to establish the RAS Signalling Channel. In network environments that do not have a Gatekeeper, the RAS Signalling Channel is not used. In network environments which contain a Gatekeeper (a Zone), the RAS Signalling Channel is opened between the endpoint and the Gatekeeper. The RAS Signalling Channel is opened prior to the establishment of any other channels between H.323 endpoints. This channel is described in detail in clause 7.

6.2.10 Call signalling function

The call signalling function uses H.225.0 call signalling to establish a connection between two H.323 endpoints. The Call Signalling Channel is independent from the RAS Channel and the H.245 Control Channel. H.245 open logical channel procedures are not used to establish the Call Signalling Channel. The Call Signalling Channel is opened prior to the establishment of the H.245 Channel and any other logical channels between H.323 endpoints. In systems that do not have a Gatekeeper, the Call Signalling Channel is opened between the two endpoints involved in the call. In systems which contain a Gatekeeper, the Call Signalling Channel is opened between the endpoint and the Gatekeeper or between the endpoints themselves as chosen by the Gatekeeper. This channel is described in detail in clause 7.

6.2.11 H.225.0 layer

Logical channels of video, audio, data or control information are established according to the procedures of ITU-T Rec. H.245. Logical channels are unidirectional and are independent in each direction of transmission. Some logical channels, such as for data, may be bidirectional and are associated through the bidirectional open logical channel procedure of ITU-T Rec. H.245. Any number of logical channels of each media type may be transmitted, except for the H.245 Control Channel of which there shall be one per call. In addition to the logical channels, H.323 endpoints use two signalling channels for call control, and Gatekeeper related functions. The formatting used for these channels shall conform to ITU-T Rec. H.225.0.

6.2.11.1 Logical channel numbers

Each logical channel is identified by a Logical Channel Number, in the range 0 to 65535, which serves only to associate logical channels with the transport connection. Logical channel numbers are selected arbitrarily by the transmitter, except that logical channel 0 shall be permanently assigned to the H.245 Control Channel. The actual Transport Address that the transmitter shall transmit to shall be returned by the receiver in the **openLogicalChannelAck** message.

6.2.11.2 Logical channel bit rate limits

A logical channel's bandwidth shall have an upper limit specified by the minimum of the endpoint's transmit capability (if present) and the receiving endpoint's receive capability. Based on this limit, an endpoint shall open a logical channel at a bit rate at or below this upper limit. A transmitter shall transmit an information stream within the logical channel at any bit rate at or below the open logical channel bit rate. The limit applies to the information streams which are the content of the logical channel(s), not including RTP headers, RTP payload headers and network headers and other overhead.

H.323 endpoints shall obey the **flowControlCommand** message of H.245, which commands a limit to the bit rate of a logical channel or the aggregate bit rate of all logical channels. H.323 endpoints that want to limit the bit rate of a logical channel or the aggregate bit rate of all logical channels should send the **flowControlCommand** message to the transmitting endpoint.

When the terminal has no information to send in a given channel, the terminal shall send no information. Fill data shall not be sent on the network in order to maintain a specific data rate.

6.3 Gateway characteristics

The Gateway shall provide the appropriate translation between transmission formats (for example H.225.0 to/from H.221) and between communications procedures (for example H.245 to/from H.242). This translation is specified in ITU-T Rec. H.246. The Gateway shall also perform call setup and clearing on both the network side and the SCN side. Translation between video, audio, and data formats may also be performed in the Gateway. In general, the purpose of the Gateway (when not operating as an MCU) is to reflect the characteristics of a network endpoint to an SCN endpoint, and the reverse, in a transparent fashion.

An H.323 endpoint may communicate with another H.323 endpoint on the same network directly and without involving a Gateway. The Gateway may be omitted if communications with SCN terminals (terminals not on the network) are not required. It may also be possible for a terminal on one segment of the network to call out through one Gateway and back onto the network through another Gateway in order to bypass a router or a low bandwidth link.

The Gateway has the characteristics of an H.323 Terminal or MCU on the network and of the SCN terminal or MCU on the SCN. The choice of terminal or MCU is left to the manufacturer. The Gateway provides the necessary conversion between the different terminal types. Note that the Gateway may initially operate as a terminal, but later using H.245 signalling begin to operate as an MCU for the same call that was initially point-to-point. Gatekeepers are aware of which terminals are Gateways since this is indicated when the terminal/Gateway registers with the Gatekeeper.

A Gateway which passes T.120 data between the SCN and the network may contain a T.120 MCS Provider which connects the T.120 MCS Providers on the network to the T.120 MCS Providers on the SCN.

Four examples of an H.323 Gateway are shown in Figure 5. The diagrams show the H.323 terminal or MCU function, the SCN terminal or MCU function, and the conversion function. The H.323 terminal function has the characteristics described in 6.2. The H.323 MCU function has the characteristics described in 6.5. The Gateway appears to the other H.323 terminals on the network as one or more H.323 terminals or an H.323 MCU. It communicates with the other H.323 terminals using the procedures in this Recommendation.

The SCN terminal or MCU function has the characteristics described in the appropriate Recommendation (H.310, H.320, H.321, H.322, H.324, V.70, GSTN or ISDN speech only terminals). The Gateway appears to the terminals on the SCN as one or more of the same terminal types or MCUs. It communicates to another terminal on the SCN using the procedures described in the appropriate Recommendation for that terminal. SCN signalling procedures are beyond the scope

of this Recommendation, including such topics as whether the H.323 Gateway appears as a terminal or a network to the SCN. Note that a Gateway may convert H.323 directly to H.324 or H.310 without going to H.320.

Gateways supporting interworking with speech only terminals on GSTN or ISDN should generate and detect DTMF signals corresponding to H.245 **userInputIndications** for 0-9, *, and #. Additionally, gateways may be able to generate and detect DTMF, telephony tones and telephony signals corresponding to these events transported with a special RTP payload type, as described in 10.5.

The conversion function provides the necessary conversion of transmission format, control, audio, video, and/or data streams between the different terminal Recommendations. At a minimum, the Gateway shall provide a conversion function for the transmission format, call setup signals and procedures, and communications control signals and procedures. When required, the Gateway shall provide for H.242 to H.245 conversion. The Gateway performs the appropriate conversion between the H.225.0 Call Signalling and the SCN signalling system (Q.931, Q.2931, etc.). The conversion between H.225.0 call signalling messages on the network and Q.931 messages on the SCN is described in ITU-T Rec. H.246.

All call signalling received by the Gateway from an SCN endpoint and not applicable to the Gateway should be passed through to the network endpoint and vice versa. This signalling includes, but is not limited to, Q.932, Q.950 and H.450-series messages. This will allow H.323 endpoints to implement the Supplementary Services defined in those Recommendations. The handling of other SCN call signalling systems is for further study.

This Recommendation describes the connection of one H.323 terminal on the network to one external terminal on the SCN through the Gateway. The actual number of H.323 terminals that can communicate through the Gateway is not subject to standardization. Similarly, the number of SCN connections, number of simultaneous independent conferences, audio/video/data conversion functions, and inclusion of multipoint functions is left to the manufacturer. If the Gateway includes an MCU function on the network side, that function shall be an H.323 MCU on the network side. If the Gateway includes an MCU function on the SCN side, it may appear as an H.231/H.243 MCU or as an MCU for H.310 or H.324 systems (these MCUs are indicated as for further study in the respective Recommendations) on the SCN side.

A Gateway may be connected via the SCN to other Gateways to provide communication between H.323 terminals which are not on the same network.

Equipment which provides transparent interconnection between networks without using H-series protocols (such as routers and remote dial in units) are not Gateways as defined within the scope of this Recommendation.

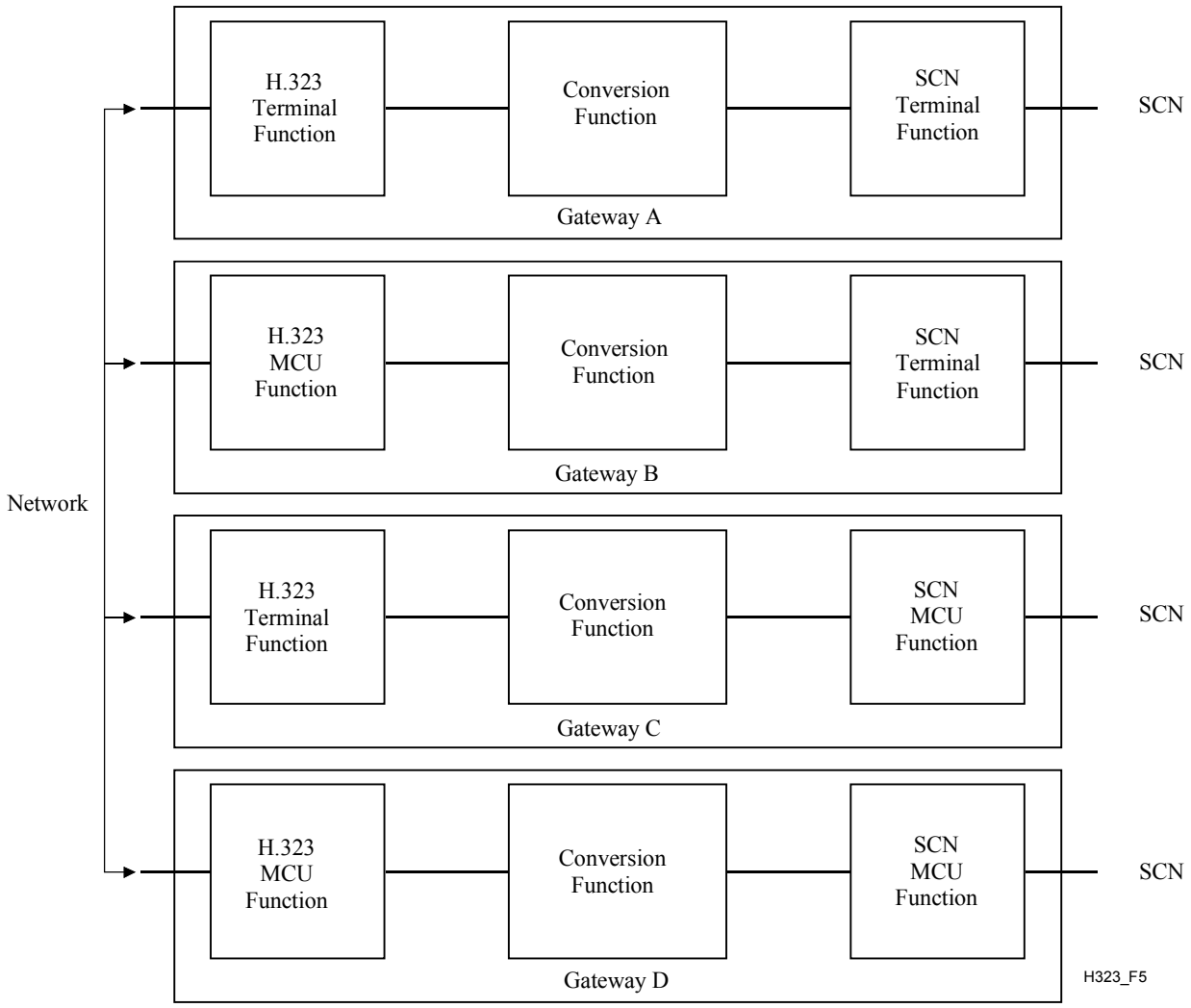


Figure 5/H.323 – H.323 gateway configurations

6.3.1 Gateway decomposition

This clause identifies a group of interfaces and functions to be used to decompose H.323 Gateways. It addresses each interface and its resulting protocol, but certain Gateway implementations may choose to group two or more functional components into a single physical device. For this reason, interfaces may provide a capability to transparently backhaul other protocols.

In Figure 6, the packet/circuit media component terminates SCN media channel and converts these streams to packet based media on the packet network interface. Interface A represents the device control protocol defined in ITU-T Rec. H.248, which is used to create, modify and delete Gateway media connections. The control logic component will accomplish signalling interworking between the SCN and H.323 sides of the Gateway.

The B interface represents the H.225.0 and H.245 protocol components that make up the H.323 signalling interfaces on the packet side of the Gateway.

Interface C will describe the ISDN type call control function between the FAS SCN services and the Gateway control logic. Interface D is a protocol that conveys the NFAS SCN signalling to the controller. This decomposition provides the flexibility to conserve SS7 code points and allows the SS7 switch to serve multiple decomposed Gateway Controllers.

The resource control elements differentiate between a high level understanding of resources in the Gateway Controller and a lower level understanding of resources in a Gateway device.

The SCN interfaces are described as a low-level interface that transports signalling and a high level SCN signalling termination that interfaces with the controller of this Gateway. This can be FAS signalling, such as ISDN PRI, or NFAS signalling, such as SS7.

Figure 6 does not represent a physical decomposition at this point. The challenge for Gateway vendors is to group these components into physical devices and implement the associated interfaces in order to produce highly scaleable, multi-vendor H.323 Gateways. The X interface is the external H.323 interface, the Y interface is the external packet media interface (i.e., RTP) and the Z interface is the external SCN interface.

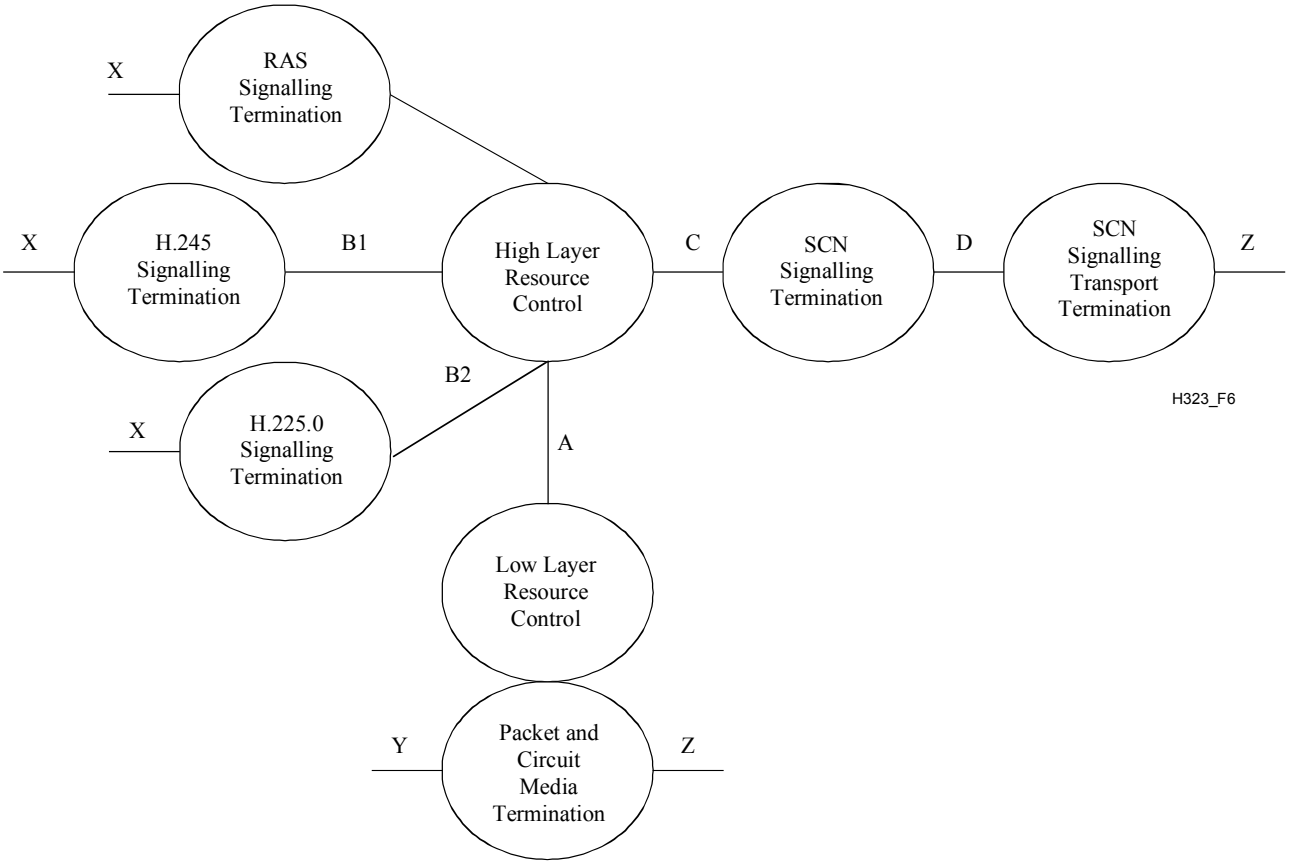


Figure 6/H.323 – Functional architecture of the decomposed gateway

6.3.1.1 Physical decompositions

This clause describes examples of possible Gateway decompositions and the internal interfaces that are required. In all cases the external interfaces, such as H.323 and SCN, remain unchanged. The controller portion of the physical Gateway is called the Media Gateway Controller (MGC). The MGC's functions are to:

- handle H.225.0 RAS messaging with an external gatekeeper;
- optionally handle the SS7 signalling interface;
- optionally handle the H.323 signalling interface.

The Media Gateway (MG) component:

- terminates the IP network interface;
- terminates the SCN network span;
- may handle H.323 signalling in some physical decompositions;
- may handle FAS SCN signalling in some physical decompositions.

Decomposed Gateways need not realize all interfaces but the MGC/MG split exposing interface A is a mandatory part of all decompositions. This will allow an MGC to control different types of MGs that may be optimized for certain applications (e.g., voice versus multimedia H.320/H.323 Gateways). The decomposition of interfaces B and C on the MG, which may require a protocol to backhaul signalling from the MG to the MGC, is for further study.

The MG terminates the IP or ATM media on the packet network side and bearer channels on the SCN network interfaces. The packet side may be IP, ATM, or an ATM network interface where audio and video packets traverse native ATM connections according to Annex C.

The MGC and MG differentiate between high-level and low-level resource management elements. The MGC is responsible for high-level resource management where it understands the availability of resources, such as echo cancellers, but does not assign specific resources to specific Gateway sessions. The MG is responsible for low-level resource allocation and management, as well as the hardware manipulations required to switch and process media streams within the Media Gateway.

6.3.1.1.1 Separate SS7 gateways

Figure 7 represents one possible Gateway decomposition for an ISUP-to-H.323 Gateway, where the SS7 Gateway, MGC, and MG functions are decomposed into separate physical devices. This arrangement exposes an ISUP signalling transport interface D and the device control interface A.

To facilitate interoperability, decomposed Gateway configurations must support interface A and contain internal H.323 and SCN signalling in the MGC.

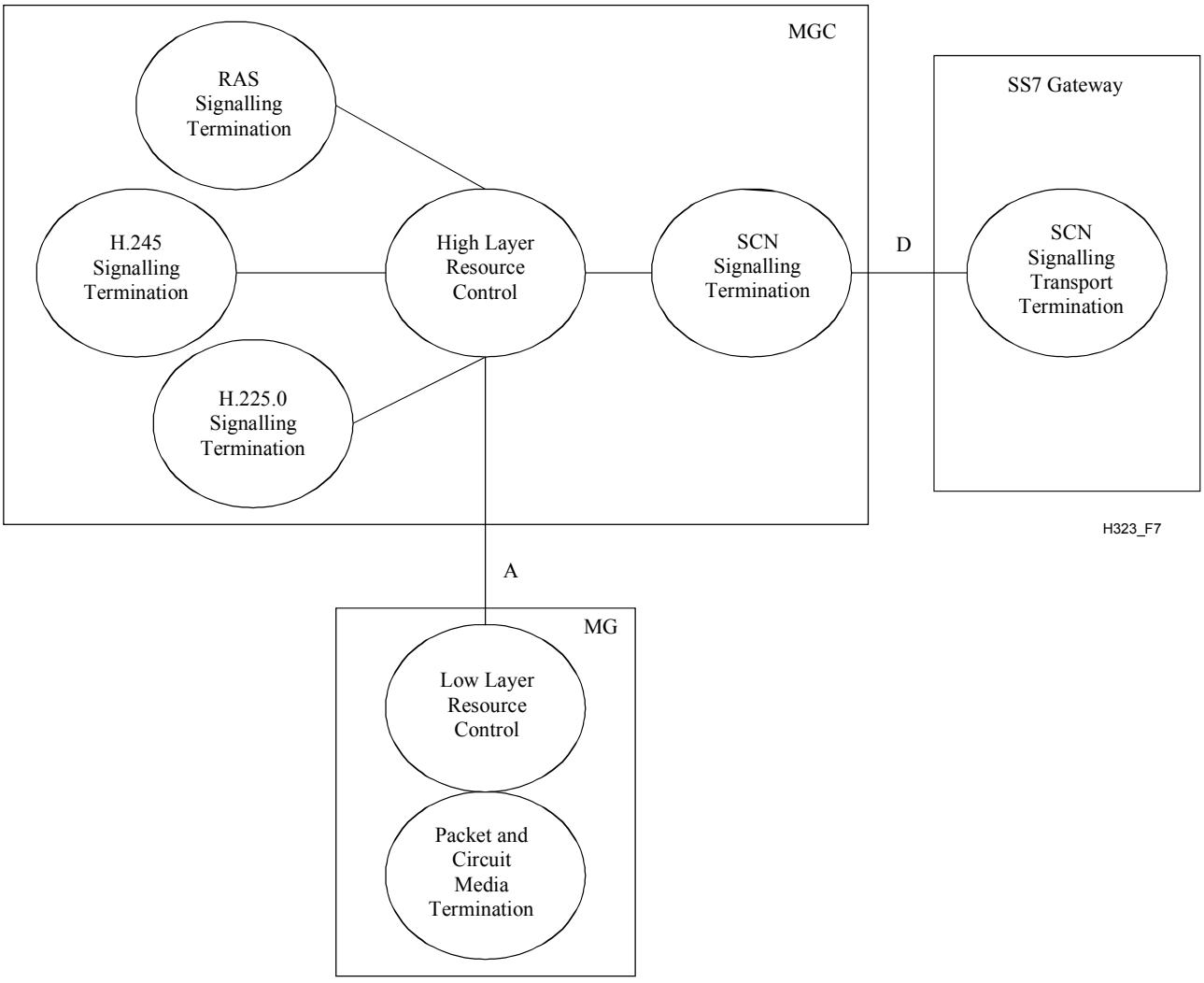


Figure 7/H.323 – SS7 gateway decomposition

6.3.1.1.2 FAS gateway decomposition

The Gateway decomposition shown in Figure 8 isolates the FAS SCN services, such as ISDN PRI on the MG, and retains the H.323 signalling on the MGC. This exposes the C and A interfaces between the MG and MGC.

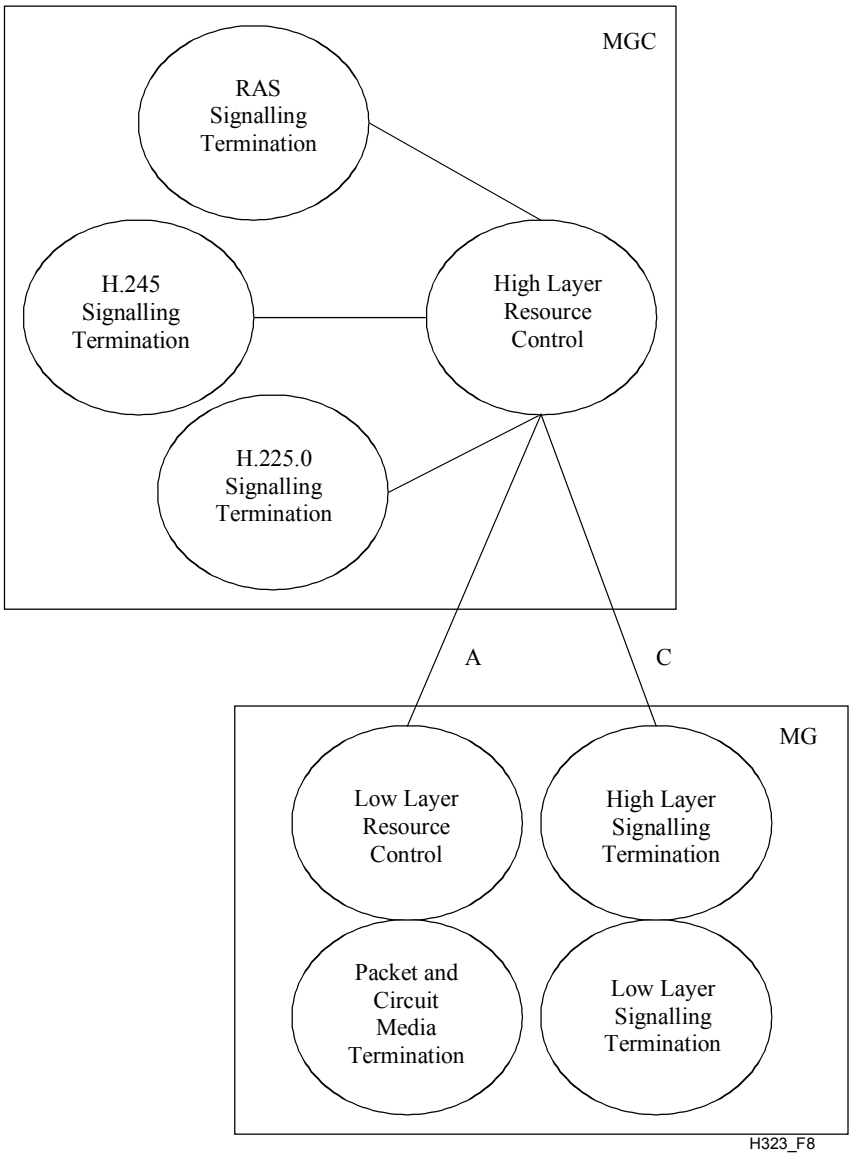


Figure 8/H.323 – FAS gateway with H.323 signalling in MG

6.3.1.1.3 SS7 gateway with H.323 signalling in the MG

The decomposition shown in Figure 9 leverages the SS7 interface of the MGC and deploys the H.323 signalling on the MG exposing interfaces D, A and B.

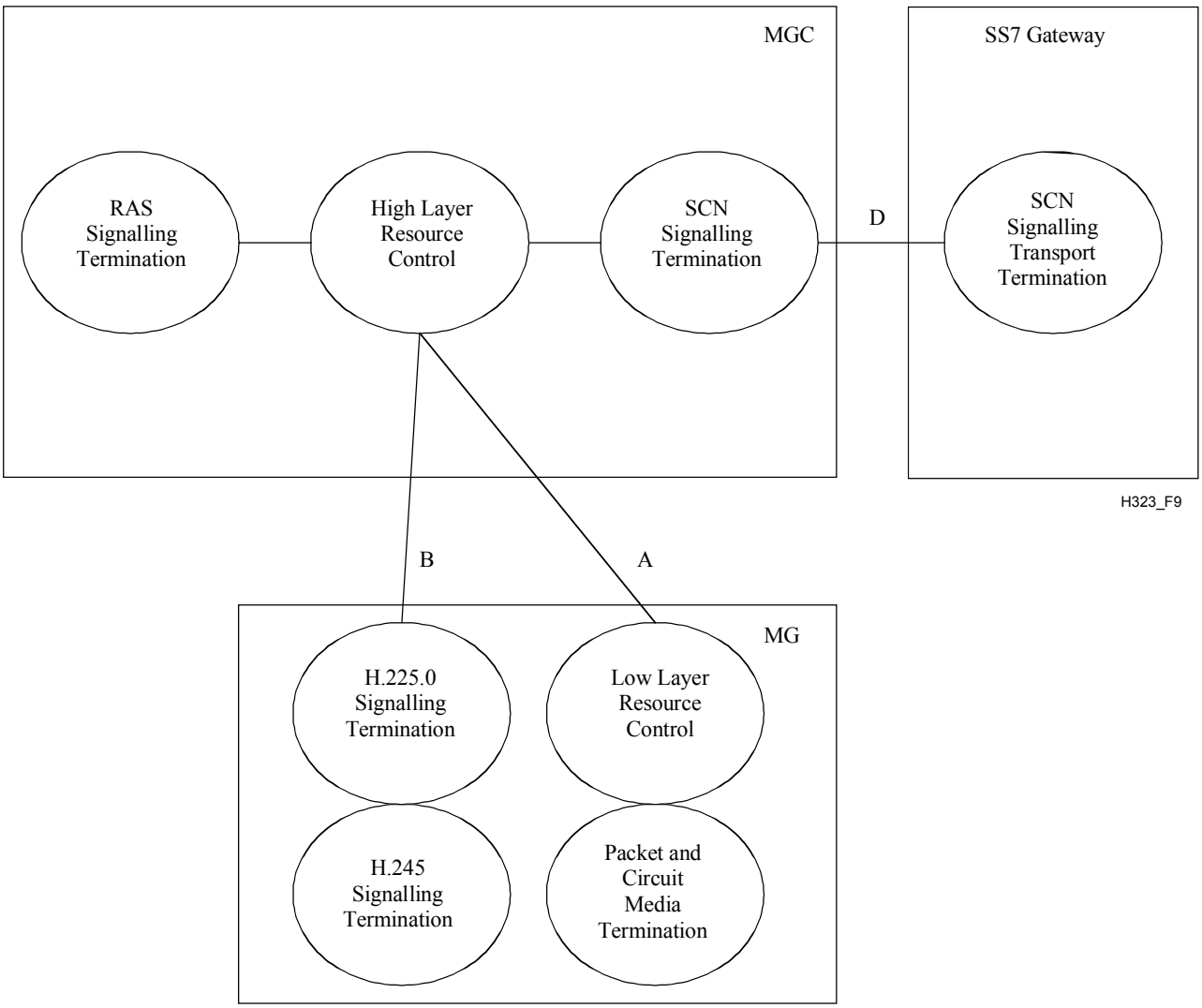


Figure 9/H.323 – SS7 terminated in the media gateway

6.3.1.1.4 FAS and H.323 signalling in the media gateway

Requirements exist for H.320 Gateways that are decomposed such that H.323 and SCN signalling are both present on the MG, along with the packet and circuit terminations. In this decomposition, signalling is handled locally by the MG and event notifications are reported to the MGC (see Figure 10).

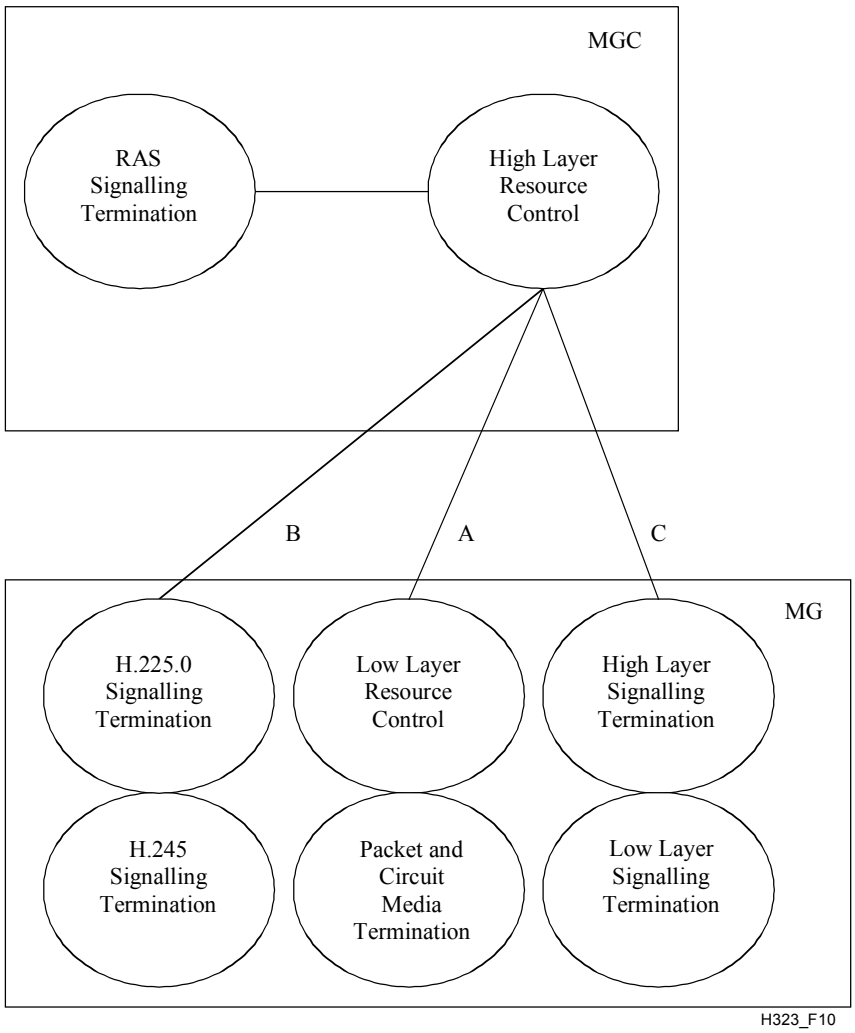


Figure 10/H.323 – FAS and H.323 signalling in the MG

6.3.1.1.5 SS7 in the media gateway

The decomposition shown in Figure 11 terminates the SS7 network in the MG and exposes the D interface between the MGC and MG.

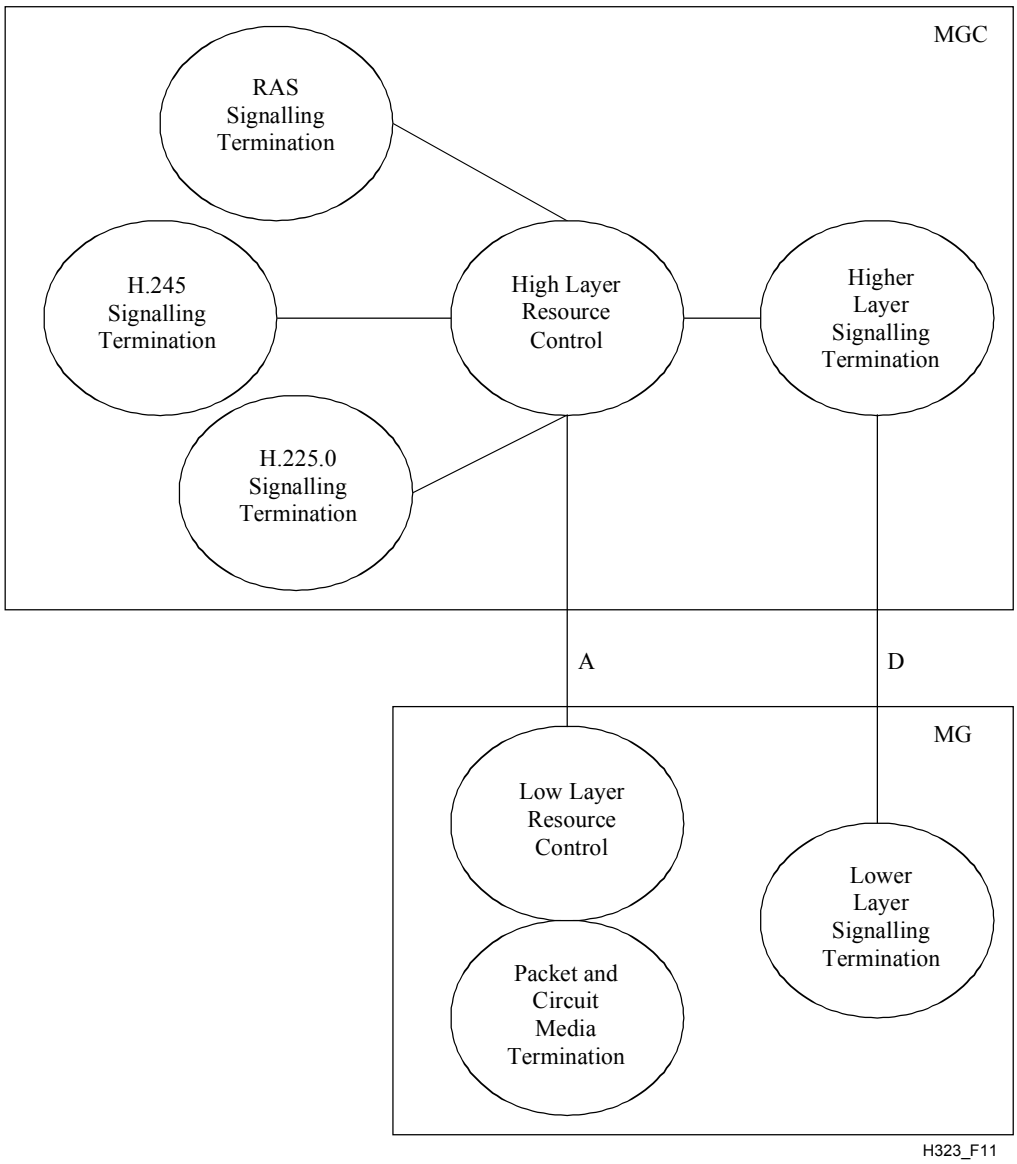


Figure 11/H.323 – SS7 terminated in the MG

6.3.2 Gateway applications

There are many applications for decomposed and composite gateways. Vendors and/or carriers may decide to use a composite or decomposed gateway depending on the application requirements. Decomposed gateways are mandated by H.248 to interwork with composite gateways.

This clause discusses some shared vocabulary between H.323 equipment, the SCN and H.248. It also provides examples of applications gateways. It is not meant to be a comprehensive list of all applications. It is also not intended to illustrate the only way such applications can be supported. In this clause the terms MG, MGC, and GW represent physical instantiations of these devices.

6.3.2.1 Overview of trunking and access gateways

The terms trunking and access gateways are used in both H.323 and H.248, and are also part of the terminology of circuit switching, where they are applied to tandem and access switches. Because the same words are used to mean different things in the context of three different architectures, this clause attempts to clarify the variations in terminology.

6.3.2.1.1 SCN terminology

In the SCN, a "tandem" or "trunking" switch refers to a switch that connects networks using an NNI protocol, such as SS7/ISUP or a CAS NNI protocol. An "access" switch refers to a switch that has user connections using BRI/PRI and is also connected via NNI protocols to a larger network. A "mixed" switch may have both functions.

6.3.2.1.2 H.323 terminology

In H.323 networks, a "trunking" gateway refers to a gateway that provides a true tandem function that is transparent to the attached networks. These attached networks might be SS7 networks, QSIG networks, or other networks. However, in all cases tunnelling is used to create full transparency and a true tandem function. Interworking between ISUP flavours is considered to take place outside of the H.323 network. Tunnelling is based on H.225.0 protocol negotiation and Annex M.

An H.323 "access" gateway provides an interworking function from another network, enterprise, or endpoint that is not fully transparent. The interworked protocols might include:

- SS7/ISUP, using Annex C/H.246;
- QSIG using H.450;
- H.320 using Annex A/H.246.

It should be noted that the H.323 "trunking" Gateway and the SCN "tandem" switch are filling the same function, but the H.323 "access gateway" and the SCN "access switch" fill very different roles. A particular point of confusion is that H.225.0 acts as both UNI and NNI signalling in the H.323 network, filling the roles of both ISUP and ISDN (BRI/PRI) in the SCN. H.323 does not make the kind of UNI/NNI signalling distinctions found in the SCN and call signalling is the same whether between endpoints directly or when mediated via network elements like an H.323 Gatekeeper or Border Element (BE).

Figure 12 summarizes the points above and also shows the relationship between H.323 domains, which have some SCN network-like characteristics. However, it is important to keep in mind that H.225.0 is also used for all call signalling, whether between terminals, zones, or domains. In addition, zones and domains are fundamentally virtual rather than physical, and switches (e.g., ATM switches used to route IP), although they may be present, are not visible from above the IP layer in the packet network.

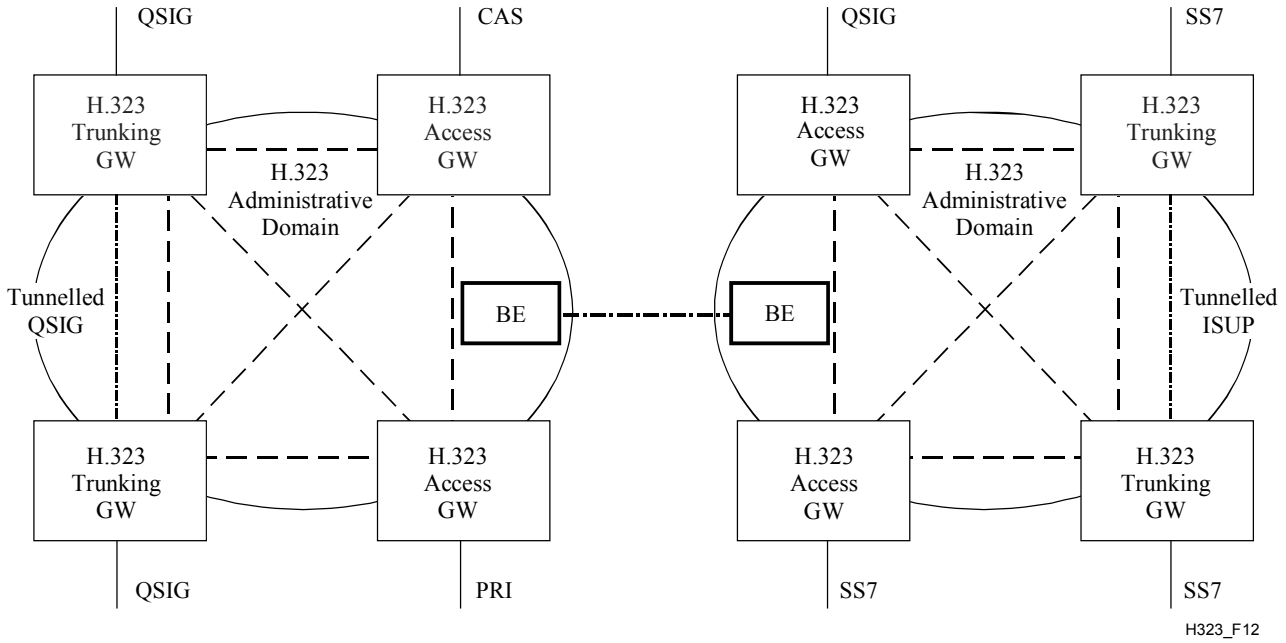


Figure 12/H.323 – Relationship of H.323, SCN and H.248 gateways

6.3.2.1.3 H.248 terminology

ITU-T Rec. H.248 also uses the terms "trunking" and "access" gateways. Noting that H.248 devices can be viewed as simply decompositions of H.323 composite gateways into MGC and MGs, it is assumed that the MGCs support H.323 and interwork using H.225.0, just as any other H.323 Gateway, including tunnelling of ISUP, etc. However, when viewed from a decomposed perspective, the terms take on slightly variant meanings. A "trunking" gateway is one in which the signalling is connected directly to the MGC, i.e., ISUP, while an "access" Gateway is one in which the signalling arrives at the MG and is then is passed via H.248 to the MGC. It is important to note that, although an "access" Gateway may support a UNI protocol, it may also support NNI CAS protocols, so that defining an H.248 "access" Gateway as a Gateway supporting a UNI interface is not accurate.

Figure 13 illustrates the architecture of ITU-T Rec. H.248. It should be noted that composed H.323 gateways are often used as "access" gateways in H.248 systems as illustrated. The diagram shows a collocated H.248 MGC and H.323 Gatekeeper.

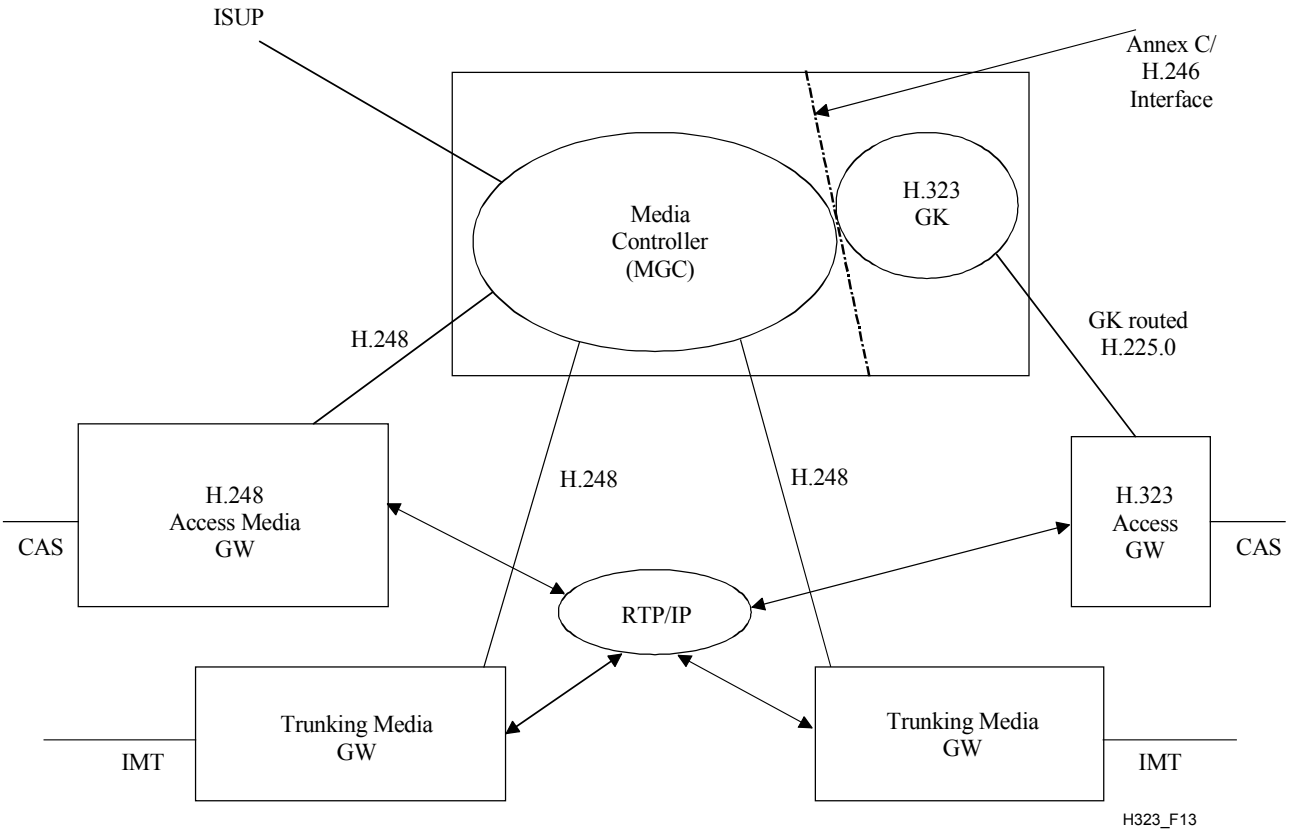


Figure 13/H.323 – Relationship of H.323 and H.248

6.3.2.2 Service provider trunking gateways

Figure 14 shows an example of a call routed across a packet switched network between two service provider trunking Gateways. In this application, the packet network acts as a tandem voice network for the service provider. For this application, interface A is used to control the Media Gateways. The packet network connects to the Switched Circuit Network via SS7 signalling and inter-machine trunks. Figure 14 illustrates the case in which SS7 A-links are used to connect to the SS7 network. In this case, the MGC terminates the signalling links directly instead of via a signalling Gateway. The MGCs pass signalling information between each other using interface X (for example, by tunnelling ISUP in an H.225.0 connection). The voice traffic flows between the two Gateways.

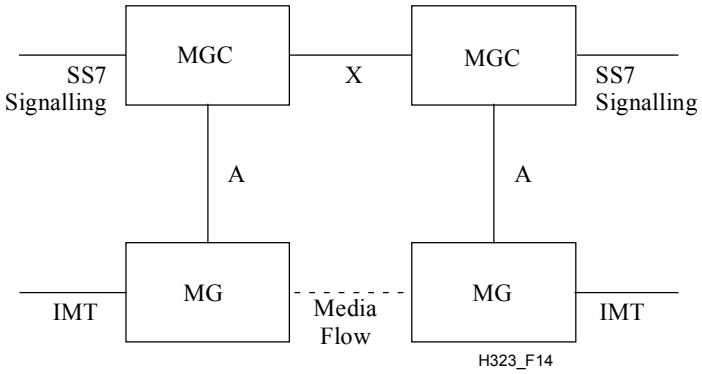


Figure 14/H.323 – Two decomposed service provider trunking gateways

6.3.2.3 Service provider access gateways

Figure 15 represents an example of a call routed across a packet switched network between a composite H.323 Service Provider Access Gateway and a decomposed service provider trunking Gateway. In this application, the service provider is providing a channel associated signalling interface to an enterprise PBX system for carrying voice calls over the provider's network. H.225.0 call signalling is used between the composite Gateway and the decomposed Gateway. The MGC performs the appropriate SS7 signalling to communicate with the service provider's SS7 network and SCN. In this example, X is H.225.0 and the MGC implements an Annex E/H.246 interworking function.

Although Recommendations exist which describe the interworking between various protocols, such as ISUP and H.323, service providers and manufacturers should carefully consider when it is appropriate to perform such interworking and the number of such interworking points. Interworking may not result in a perfect translation between two protocols and multiple translations may lead to a higher loss of transparency.

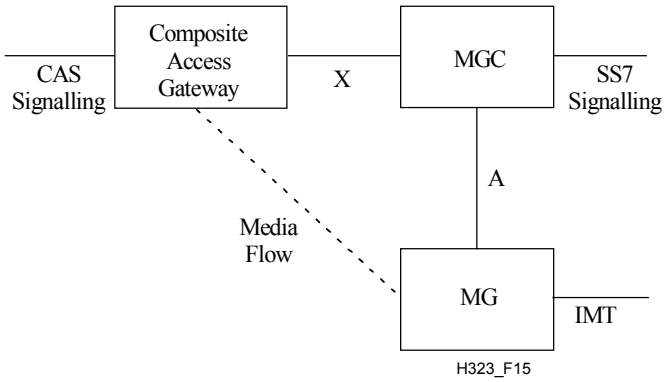


Figure 15/H.323 – A composite access gateway and decomposed trunking gateway

Figure 16 illustrates the same application in which the service provider access Gateway is also decomposed. In this case, interface A is used to control the channel associated signalling. The MGCs communicate with each other using interface X. In this particular case, if there is no signalling backhaul between the MG and the MGC, the amount of information on the call available to the MGC will be limited to what is defined by ITU-T Rec. H.248. In this example, X is H.225.0 and the MGC on the right is performing Annex E/H.246 ISUP interworking.

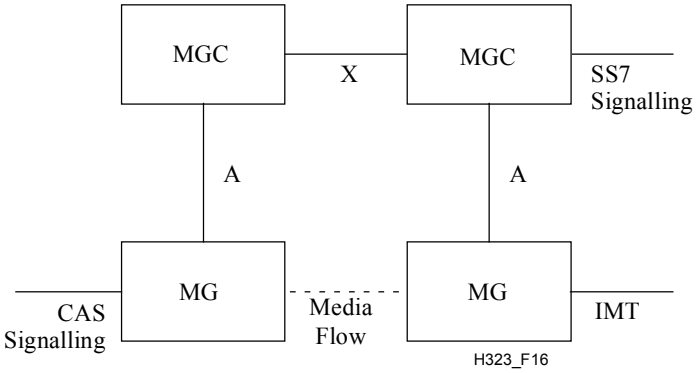


Figure 16/H.323 – Decomposed service provider access and trunking gateways

In considering which of these approaches might be best for a particular application, the following factors should be considered:

- Number of lines to be connected.
- Cost of trunks.
- Homologation issues.
- Capacity of the MGC.
- Number of access Gateways relative to trunking Gateways.
- Type of CAS protocols to be supported.
- Service provider call processing architecture.
- Network design.

For access Gateways, the application environment will determine whether a decomposed Gateway, an H.323 terminal using H.450.x, an Annex L stimulus terminal, or a composite Gateway is the most appropriate.

6.3.2.4 Enterprise trunking gateways

Figure 17 illustrates an enterprise gateway that is used between PBXs in a private voice network. The packet network is used instead of leased lines to connect the PBXs. In this case, QSIG is used for signalling between the PBXs. Since QSIG is a facility associated signalling type, the signalling may be backhauled from the Media Gateway to the Media Gateway Controller via interface C. Interface A is used between the MGC and MG for gateway control. MGCs communicate between each other over interface X, which may be H.225.0 tunnelling QSIG according to Annex M1.

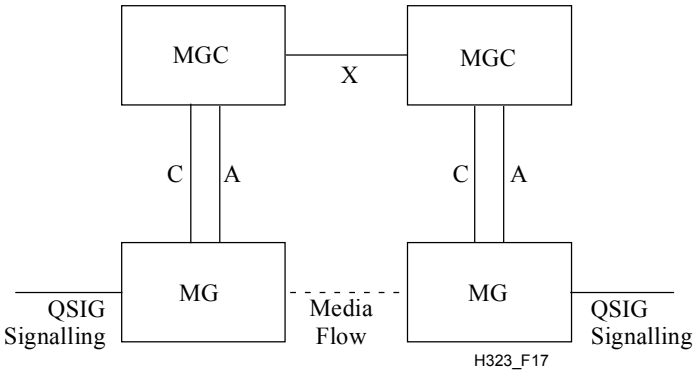


Figure 17/H.323 – Decomposed enterprise trunking gateways

Figure 18 illustrates Gateways that are used between PBXs in a private voice network. The packet network is used instead of leased lines to connect the PBXs. In this case, QSIG is also used for signalling between the PBXs. However, QSIG tunnelling over interface X is used to carry QSIG signalling between a composite Gateway and a decomposed Gateway. Other combinations, such as composite-composite and decomposed-decomposed, could also be used.

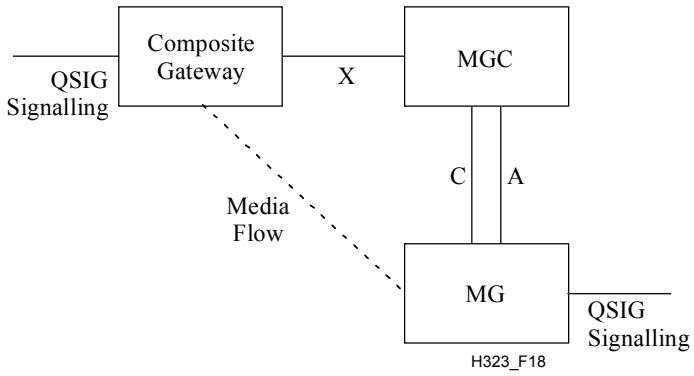


Figure 18/H.323 – QSIG tunnelling example

6.3.2.5 Enterprise to service provider access gateways

In some cases, an enterprise H.323 network will communicate with the PSTN via a decomposed gateway. This is illustrated in Figure 19. In this case, the decomposed gateway communicates to the H.323 endpoints via H.323 signalling (H.225, H.245, etc.). The decomposed gateway connects to the PSTN via ISDN PRI. The D-channel signalling can be backhauled via interface C.

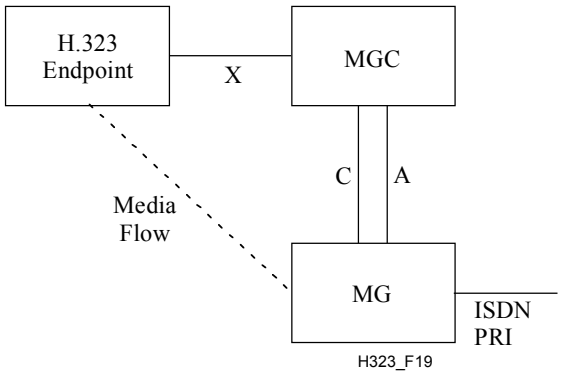


Figure 19/H.323 – Decomposed gateway and H.323 endpoint

Another enterprise access application uses H.248 to manage terminals, but appears as a composite Gateway to composite Gateways on other premises as shown in Figure 20. In this example, H.450.x is used to provide supplementary services interworking.

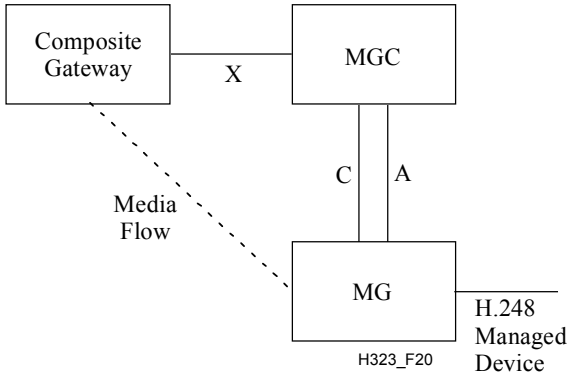


Figure 20/H.323 – Composite gateway and H.248 managed devices

An additional enterprise access application uses Annex L to manage terminals, but appears as a composite Gateway to composite Gateways on other premises as shown in Figure 21. In this example, H.450.x may be used to provide supplementary services interworking. In this example, X1 is H.225.0 with H.450, while X2 is H.225.0 with Annex L stimulus signalling.

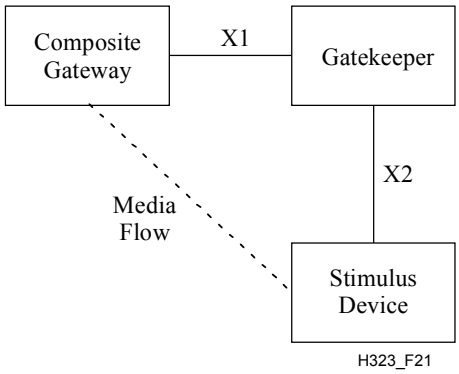


Figure 21/H.323 – Composite gateway and Annex L device

It should be noted that the Annex L terminals of Figure 21 may interwork with the H.248 managed terminals of Figure 20 using H.450.x. These configurations allow extensive feature innovation on the enterprise while supporting inter-enterprise interoperability using H.450.x. Note that Gatekeeper routed call signalling is used in Figure 21 in the enterprise Gatekeeper managing the Annex L terminals, although the other enterprise Gateways may use the direct call model and have a different Gatekeeper.

6.4 Gatekeeper characteristics

The Gatekeeper, which is optional in an H.323 system, provides call control services to the H.323 endpoints. More than one Gatekeeper may be present and communicate with each other in an unspecified fashion. The Gatekeeper is logically separate from the endpoints; however, its physical implementation may coexist with a terminal, MCU, Gateway, MC, or other non-H.323 network device.

There is one and only one Gatekeeper in a Zone at any given time, although multiple distinct devices may provide the Gatekeeper function in a Zone. Multiple devices that provide the RAS signalling function for the Gatekeeper are referred to as Alternate Gatekeepers. Each Alternate Gatekeeper may appear to endpoints as a distinct Gatekeeper. Communication between Alternate Gatekeepers and other devices that provide the Gatekeeper function for the Zone is outside the scope of the Recommendation.

When it is present in a system, the Gatekeeper shall provide the following services:

- Address Translation – The Gatekeeper shall perform alias address to Transport Address translation. This should be done using a translation table which is updated using the Registration messages described in clause 7. Other methods of updating the translation table are also allowed.
- Admissions Control – The Gatekeeper shall authorize network access using ARQ/ACF/ARJ H.225.0 messages. This may be based on call authorization, bandwidth, or some other criteria which is left to the manufacturer. It may also be a null function which admits all requests.
- Bandwidth Control – The Gatekeeper shall support BRQ/BRJ/BCF messages. This may be based on bandwidth management. It may also be a null function which accepts all requests for bandwidth changes.

- Zone Management – The Gatekeeper shall provide the above functions for terminals, MCUs, and Gateways which have registered with it as described in 7.2.

The Gatekeeper may also perform other optional functions such as:

- Call Control Signalling – The Gatekeeper may choose to complete the call signalling with the endpoints and may process the call signalling itself. Alternatively, the Gatekeeper may direct the endpoints to connect the Call Signalling Channel directly to each other. In this manner, the Gatekeeper can avoid handling the H.225.0 call control signals. The Gatekeeper may have to act as the network as defined in ITU-T Rec. Q.931 in order to support supplementary services. This operation is for further study.
- Call Authorization – Through the use of the H.225.0 signalling, the Gatekeeper may reject calls from a terminal due to authorization failure. The reasons for rejection may include, but are not limited to, restricted access to/from particular terminals or Gateways and restricted access during certain periods of time. The criteria for determining if authorization passes or fails is outside the scope of this Recommendation.
- Bandwidth Management – Control of the number of H.323 terminals permitted simultaneous access to the network. Through the use of the H.225.0 signalling, the Gatekeeper may reject calls from a terminal due to bandwidth limitations. This may occur if the Gatekeeper determines that there is not sufficient bandwidth available on the network to support the call. The criteria for determining if bandwidth is available is outside the scope of this Recommendation. Note that this may be a null function, i.e., all terminals are granted access. This function also operates during an active call when a terminal requests additional bandwidth.
- Call Management – For example, the Gatekeeper may maintain a list of ongoing H.323 calls. This information may be necessary to indicate that a called terminal is busy and to provide information for the Bandwidth Management function.
- Alias Address Modification – The Gatekeeper may return a modified Alias Address. If the Gatekeeper returns an alias address in an ACF, the endpoint shall use the Alias Address in establishing the connection.
- Dialled Digit Translation – The Gatekeeper may translate dialled digits into an E.164 number or a Private Network number.
- Gatekeeper management information data structure – For further study.
- Bandwidth reservation for terminals not capable of this function – For further study.
- Directory services – For further study.

In order to support ad hoc Multipoint Conferences, the Gatekeeper may choose to receive the H.245 Control Channels from the two terminals in a point-to-point conference. When the conference switches to a multipoint conference, the Gatekeeper can redirect the H.245 Control Channel to an MC. The Gatekeeper need not process the H.245 signalling; it only needs to pass it between the terminals or the terminals and the MC.

Networks which contain Gateways should also contain a Gatekeeper in order to translate incoming **dialledDigits** or **partyNumber** (including **e164Number** and **privateNumber**) addresses into Transport Addresses.

H.323 entities that contain a Gatekeeper shall have a mechanism to disable the internal Gatekeeper so that when there are multiple H.323 entities that contain a Gatekeeper on a network, the H.323 entities can be configured into the same Zone.

6.5 Multipoint controller characteristics

The MC provides control functions to support conferences between three or more endpoints in a multipoint conference. The MC carries out the capabilities exchange with each endpoint in a multipoint conference. The MC sends a capability set to the endpoints in the conference indicating the operating modes in which they may transmit. The MC may revise the capability set that it sends to the terminals as a result of terminals joining or leaving the conference or for other reasons.

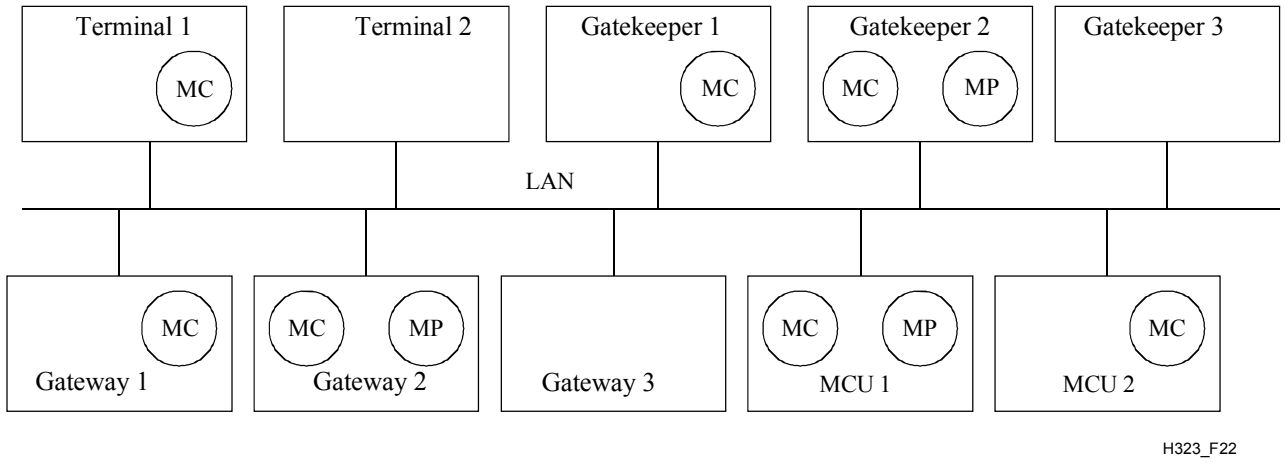
In this manner, the MC determines the Selected Communications Mode (SCM) for the conference. The SCM may be common for all endpoints in the conference. Alternatively, some endpoints may have a different SCM than other endpoints in the conference. The manner in which the MC determines an SCM is not within the scope of this Recommendation.

As part of multipoint conference setup, an endpoint will become connected to an MC on its H.245 Control Channel. This connection may occur:

– via an explicit connection with an MCU;
– via an implicit connection to the MC within a Gatekeeper;
– via an implicit connection to the MC within another terminal or Gateway in the multipoint conference;
– via an implicit connection through a Gatekeeper to an MCU.

The choice of conference mode (e.g., decentralized or centralized) occurs after connection with the MC using H.245 signalling. The choice of conference mode may be limited by the capability of the endpoints or the MC.

The MC may be located within a Gatekeeper, Gateway, terminal, or MCU. See Figure 22.



NOTE – Gateway, Gatekeeper and MCU can be a single device.

Figure 22/H.323 – Possible locations of MC and MP in H.323 system

An MC within a terminal is not callable. It can be included in the call in order to process the H.245 signalling to support ad hoc multipoint conferences. In this case, there may be no distinction between the MC and the H.245 Control Function (see 6.2.8) of the terminal. Communications between them are outside the scope of this Recommendation.

An MC located with the Gatekeeper is not callable; however, an MCU located with a Gatekeeper may be callable. An MCU located with a Gatekeeper may function as an independent MCU. An MC located with a Gatekeeper may be used to support ad hoc multipoint conferences when the Gatekeeper receives the H.245 Control Channels from the endpoints. In this manner, the Gatekeeper

can route the H.245 Control Channels to the MC at the start of the call or when the conference switches to multipoint.

The Gateway can function as a terminal or an MCU. When functioning as a terminal, the Gateway may contain an MC. This has the same characteristics as described above for an MC within a terminal.

An MCU always contains an MC. The MCU is callable and the MC processes the H.245 Control Channel from all of the endpoints.

When two or more endpoints are in a conference, the endpoints shall use the master slave resolution procedure of ITU-T Rec. H.245 to determine the MC that will control the conference.

After the capability exchange and master/slave determination, the MC may first assign a terminal number to a new endpoint using the **terminalNumberAssign**. The MC shall then notify the other endpoints of the new endpoint in the conference using **terminalJoinedConference**. The new endpoint may request a list of other endpoints in the conference using the **terminalListRequest**.

6.6 Multipoint processor characteristics

The MP receives audio, video and/or data streams from the endpoints involved in a centralized or hybrid multipoint conference. The MP processes these media streams and returns them to the endpoints.

Communications between the MC and the MP are not subject to standardization.

The MP may process one or more media stream types. When the MP processes video, it shall process the video algorithms and formats as described in 6.2.4. When the MP processes audio, it shall process the audio algorithms as described in 6.2.5. When the MP processes data, it shall process data streams as described in 6.2.7.

An MP which processes video shall provide either video switching or video mixing. Video switching is the process of selecting the video that the MP outputs to the terminals from one source to another. The criteria used to make the switch may be determined through detection of a change in speaker (sensed by the associated audio level) or through H.245 control. Video mixing is the process of formatting more than one video source into the video stream that the MP outputs to the terminals. An example of video mixing is combining four source pictures into a two-by-two array in the video output picture. The criteria for which sources and how many are mixed is determined by the MC until other controls are defined. The use of the T.120-series Recommendations for these control functions is for further study.

An MP which processes audio shall prepare N-audio outputs from M-audio inputs by switching, mixing, or a combination of these. Audio mixing requires decoding the input audio to linear signals (PCM or analogue), performing a linear combination of the signals and recoding the result to the appropriate audio format. The MP may eliminate or attenuate some of the input signals in order to reduce noise and other unwanted signals. Each audio output may have a different mix of input signals providing for private conversations. The terminals shall assume that their audio is not present in the audio stream returned to them. Terminal removal of its own audio from the MP audio output is for further study.

An MP which processes T.120 data shall be capable of acting as a non-leaf MCS provider and should be capable of acting as the Top MCS Provider. An MP may also process non-standard data, transparent user data and/or other types of data.

The MP may provide algorithm and format conversion, allowing terminals to participate in a conference at different SCMs.

The MP is not callable, the MCU which it is a part of is callable. The MP terminates and sources the media channels.

6.7 Multipoint control unit characteristics

The MCU is an endpoint which provides support for multipoint conferences. The MCU shall consist of an MC and zero or more MPs. The MCU uses H.245 messages and procedures to implement features similar to those found in ITU-T Rec. H.243.

A typical MCU that supports centralized multipoint conferences consists of an MC and an audio, video and data MP. A typical MCU that supports decentralized multipoint conferences consists of an MC and a data MP supporting ITU-T Rec. T.120. It relies on decentralized audio and video processing.

The network side of a Gateway may be an MCU. A Gatekeeper may also include an MCU. In either case, they are independent functions that happen to be colocated.

The MCU shall be callable by other endpoints using the procedures of clause 8.

6.8 Multipoint capability

6.8.1 Centralized multipoint capability

All endpoints shall have centralized multipoint capability. In this mode of operation they communicate with the MC of the MCU in a point-to-point manner on the control channel and with the MP on the audio, video and data channels. In this mode, the MC performs H.245 multipoint control functions, while the MP performs video switching or mixing, audio mixing, and T.120 multipoint data distribution. The MP transmits the resulting video, audio and data streams back to the endpoints. The MP may have the capability to convert between different audio, video and data formats and bit rates, allowing the endpoints to participate in the conference using different communications modes.

The MCU may use multicast to distribute the processed media streams if the endpoints in the conference can receive multicast transmissions. Multicast distribution of data is for further study.

This mode is signalled by the following H.245 capabilities: **centralizedControl**, **centralizedAudio**, **centralizedVideo** and **centralizedData**. Optionally, **distributedAudio** and **distributedVideo** may be used to indicate multicast distribution of media streams.

6.8.2 Decentralized multipoint capability

If the endpoints have decentralized multipoint capability, they communicate with the MC of an MCU, Gateway, Gatekeeper, or endpoint in a point-to-point mode on the H.245 Control Channel and optionally with an MP on data channels. The endpoints shall have the capability to multicast their audio and video channels to all other endpoints in the conference. The MC may control which endpoint or endpoints are actively multicasting audio and/or video (for example by using the **flowControlCommand** on either channel).

The endpoints receive multicast video channels and select one or more of the available channels for display to the user. The endpoints receive the multicast audio channels and perform an audio mixing function in order to present a composite audio signal to the user.

The MC may provide conference control functions such as chair control, video broadcast and video selection. This shall be done by receiving H.245 from an endpoint and then sending the appropriate control to other endpoints to enable or disable their video multicast. T.120 commands may optionally provide the same functions.

This mode is signalled by the following H.245 capabilities: **centralizedControl**, **distributedAudio**, **distributedVideo** and **centralizedData**.

6.8.3 Hybrid multipoint – Centralized audio

If the endpoints and MCU have hybrid multipoint-centralized audio capability, they may use distributed multipoint for video and centralized multipoint for audio. In this mode, the endpoints communicate with the MC in a point-to-point mode on the H.245 Control Channel and optionally with an MP on data channels.

The endpoints shall have the capability to multicast their video channels to all other endpoints in the conference. The MC may control which endpoint or endpoints are actively multicasting video. The endpoints receive multicast video channels and select one or more of the available channels for display to the user.

All of the endpoints in the conference transmit their audio channels to the MP. The MP performs the audio mixing function and outputs the resulting audio streams to the endpoints. The MP may produce an exclusive audio sum for each endpoint in the conference. Multicast distribution of processed audio is for further study.

This mode is signalled by the following H.245 capabilities: **centralizedControl**, **centralizedAudio**, **distributedVideo** and **centralizedData**.

6.8.4 Hybrid multipoint – Centralized video

If the endpoints and MCU have hybrid multipoint-centralized video capability, they may use distributed multipoint for audio and centralized multipoint for video. In this mode, the endpoints communicate with the MC in a point-to-point mode on the H.245 Control Channel and optionally with an MP on data channels.

The endpoints shall have the capability to multicast their audio channels to all other endpoints in the conference. The MC may control which endpoint or endpoints are actively multicasting audio. The endpoints receive multicast audio channels and perform a mixing function in order to present a composite audio signal to the user.

All of the endpoints in the conference transmit their video channels to the MP. The MP performs the video switching, mixing, or format conversion functions and outputs the resulting video streams to the endpoints. The MP may produce an exclusive video stream for each endpoint in the conference, or it may multicast a video stream to all participating endpoints, in order to minimize the bandwidth used on the network.

This mode is signalled by the following H.245 capabilities: **centralizedControl**, **distributedAudio**, **centralizedVideo** and **centralizedData**.

6.8.5 Establishment of common mode

The MC shall coordinate a common communications mode between the endpoints in the multipoint conference. The MC may force endpoints into a particular common mode of transmission (as allowed by their capability sets) by sending to the endpoint a receive capability set listing only the desired mode of transmission, or the MC may rely on **multipointModeCommand** and mode preference commands to enforce mode symmetry. The latter approach should be used since it allows the endpoints to know the full range of conference capabilities available that can be requested.

If the MCU has the capability to convert audio and/or video formats, it may not be necessary to force all endpoints into the same communications mode.

6.8.6 Multipoint rate matching

Since the endpoints on each link in a multipoint configuration may attempt to operate at different bit rates, the MC shall send H.245 **flowControlCommand** messages to limit the transmitted bit rates to those which can be sent to receivers.

6.8.7 Multipoint lip synchronization

An MP which is providing audio mixing in either the Centralized or Hybrid multipoint conferences shall modify the time tags of the audio and video streams, taking into account its own time base, in order to maintain audio and video synchronization. Further, when the MP processes the audio and/or video to generate a new stream sourced from the MP, the MP shall generate its own sequence numbers in the audio and video packets.

When mixing audio, the MP should synchronize each of the incoming audio streams to its own timing, mix the audio streams, and then shall generate a new audio stream based on its own timing with its own sequence numbers. If the MP is also switching video, the switched stream shall have its original time-stamp replaced with the MP time base to synchronize it with the mixed audio stream and shall have a new sequence number representing the stream from the MP.

In the case of distributed multipoint conferences, the receiving endpoint may be able to maintain lip synchronization by aligning the selected video stream and its associated audio by using the RTP time tags. Alignment of the other audio streams may not be necessary. If multiple video streams are displayed, the associated audio streams should be aligned.

It may not be possible to guarantee lip synchronization in hybrid multipoint conferences.

6.8.8 Multipoint encryption

In a centralized multipoint configuration, the MP is considered to be a trusted entity. Each port of the MP decrypts the information streams from each of the H.323 endpoints and encrypts the information streams to each endpoint in accordance with 10.1. Operation of an untrusted MCU is for further study.

6.8.9 Cascading multipoint control units

The multipoint control function may be distributed between several MCs. This is called cascading. Cascading allows two or more MCs to communicate with each other in order to control a multipoint conference. Cascading MCs consists of establishing an H.245 Control Channel between the MCs. One MC is defined as the Master MC while the other MCs are defined as Slave MCs.

The procedures for cascading MCs are defined in 8.4.5.

6.9 Models for supplementary services

The ability to support a large variety of supplementary services and features is a requirement for many telephony solutions, regardless of the underlying technologies.

For many such services, an associated requirement is that a high level of interoperability exists between equipment provided by different vendors. This requirement leads to standards-based solutions.

At the same time, equipment suppliers require the ability to provide services that highlight their own products. This can be achieved using proprietary means, but interoperability is compromised. In some cases, such a penalty may be acceptable or desirable, but often this is not so.

The goal, therefore, is to define a standard that is sufficiently flexible that it can support all (or most of) the services that a vendor may wish to supply.

Within the H.323 environment, there are several different methods by which services can be provided: the H.450.x series of Recommendations, ITU-T Rec. H.248 in association with its packages, Annex L and Annex K. Although there is commonality of certain design goals for each of these solutions, the emphasis varies and each is more appropriate for certain circumstances. These solutions represent a spectrum of options for system and feature implementation, from purely peer-peer (functional) control to purely master/slave (stimulus) control, using either first or third party control. Rather than competing, they are complementary, allowing for freedom of choice to the system developer.

The H.450.x series of Recommendations is designed for interoperability of services at a functional level. Its derivation from QSIG ensures interworking with many private networking systems. Services are defined for peer-peer relationships, with feature intelligence typically resident in the endpoint. An H.450-based service must normally be explicitly supported by each affected endpoint in the system. This distribution of service control allows endpoints to be more self-supporting and self-contained and is ideally supported by higher-end endpoints.

The other protocols provide for stimulus level control, where a full understanding of a service is normally only required by a single entity, typically in a master-slave relationship. Such stimulus-based methods use a set of well-defined atomic functions, which, in various combinations, provide any number of services.

Stimulus protocols simplify the introduction of new services. However, different implementations of the same service may differ sufficiently to complicate interoperability, even within the same network type.

Annex L, like H.450, builds on H.323 and all Annex L endpoints are H.323 compliant by definition. It allows standard H.323 procedures to be used for call signalling and media control. Feature intelligence beyond basic call control is implemented in a centralized Feature Server (associated with a Gatekeeper or H.323 endpoint). The protocol allows services to be provided by one or more Feature Servers. Thus Annex L represents a hybrid of peer control and master/slave control models, where intelligence is split between the endpoint and the Feature Server.

Annex K allows third party control of an H.323 call based on a separate control channel (using HTTP [48]) for user interaction. There is no fixed set of capabilities for the user interface, as various types of text formats, images and sounds may be utilized dynamically as registered MIME [49] types. The service provider (the HTTP server) is responsible for the mapping between HTTP events and call control actions (H.450 or other messages) for supplementary services, so the H.323 endpoint is unaware of the HTTP application. The service provider may be associated with the local Gatekeeper, the remote endpoint, or remote Gatekeeper within a call.

H.248 is a generic gateway "device control" protocol, based entirely on a master/slave (stimulus) control model wherein all control intelligence is maintained in a central entity (the Media Gateway Controller, or MGC) and the endpoint (the Media Gateway, or MG) is a slave. H.248 is designed to be independent of the call control protocol and therefore does not require that endpoints be H.323 compliant. H.248 was developed for control of media gateways and it implies a tight relationship between the MGC and the MG, where a user can subscribe to features from only one MGC at a time. H.248 is designed to be easily extensible by the use of packages to define specific support, so that the services that an H.248-based system can support are limited only by the packages supported by the MGC and the MG.

7 Call signalling

Call signalling is the messages and procedures used to establish a call, request changes in bandwidth of the call, get status of the endpoints in the call, and disconnect the call. Call signalling uses messages defined in ITU-T Rec. H.225.0 and the procedures described in clause 8. This clause describes some call signalling concepts.

7.1 Addresses

7.1.1 Network address

Each H.323 entity shall have at least one Network Address. This address uniquely identifies the H.323 entity on the network. Some entities may share a Network Address (i.e., a terminal and a colocated MC). This address is specific to the network environment in which the endpoint is located. Different network environments may have different Network Address formats.

An endpoint may use different Network addresses for different channels within the same call.

7.1.2 TSAP identifier

For each Network Address, each H.323 entity may have several TSAP Identifiers. These TSAP Identifiers allow multiplexing of several channels sharing the same Network Address.

Endpoints have one well-known TSAP Identifier defined: the Call Signalling Channel TSAP Identifier. Gatekeepers have one well-known TSAP Identifier defined: the RAS Channel TSAP Identifier and one well-known multicast address defined: Discovery Multicast Address. These are defined in Appendix IV/H.225.0.

Endpoints and H.323 entities should use dynamic TSAP Identifiers for the H.245 Control Channel, Audio Channels, Video Channels, and Data Channels. The Gatekeeper should use a dynamic TSAP Identifier for Call Signalling Channels. The RAS Channels and Signalling Channels may be redirected to dynamic TSAP Identifiers during the registration procedure.

7.1.3 Alias address

An endpoint may also have one or more alias addresses associated with it. An alias address may represent the endpoint or it may represent conferences that the endpoint is hosting. The alias addresses provide an alternate method of addressing the endpoint. These address include **dialledDigits** or **partyNumber** addresses (including private telephone numbers and public E.164 numbers), H.323 IDs (alphanumeric strings representing names, e-mail like addresses, etc.), and any others defined in ITU-T Rec. H.225.0. Alias addresses shall be unique within a Zone. Gatekeepers, MCs, and MPs shall not have alias addresses.

NOTE – Versions 1, 2 and 3 of ITU-T Recs H.323 and H.225.0 referred to dialled digits in general as E.164 addresses (and **dialledDigits** was **e164**), which they were not. Also, those versions of ITU-T Recs H.323 and H.225.0 referred to E.164 addresses as Public Party Numbers (**e164Number** was **publicPartyNumber**): nowhere was it made clear that public party numbers were E.164 numbers. This terminology change does not affect backward compatibility in any way. Refer to Appendix V for a detailed discussion on the usage of E.164 numbers.

When there is no Gatekeeper in the system, the calling endpoint shall address the called endpoint directly using the Call Signalling Channel Transport Address of the called endpoint. When there is a Gatekeeper in the system, the calling endpoint may address the called endpoint by its Call Signalling Channel Transport Address, or alias address. The Gatekeeper shall translate the latter into a Call Signalling Channel Transport Address.

The called endpoint's **dialledDigits** address may consist of an optional access code followed by a telephone number specific to the service provider's numbering plan. The access code consists of n digits from the set of 0 to 9, *, and #. The number of digits and their meaning is left to the discretion of the manufacturer. One purpose of such an access code might be to request access to a Gateway. The Gatekeeper may alter this address prior to sending it to the destination. The Gatekeeper may also provide a **partyNumber** to use in place of the **dialledDigits**.

The H.323 ID consists of a string of ISO/IEC 10646-1 characters as defined in ITU-T Rec. H.225.0. It may be a user name, conference name, e-mail name, or other identifier.

An endpoint may have more than one alias address (including more than one of the same type) which is translated to the same Transport Address.

7.1.4 H.323 URL scheme

One of the alias types defined by ITU-T Rec. H.323 is the **url-ID**, which is intended to contain standard URL schemes that may be used to reach resources. An H.323 entity may accept any valid URL that it understands, but should support the H.323 URL as defined in this clause.

The H.323 URL is intended to help an entity resolve the address of another H.323 entity. It is composed of two parts: the *user* and the *hostport*. The *user* specifies an alias for the entity, such as a user or a service, without carrying any information about the location of the entity. The *hostport*, on the other hand, is the domain name of the Endpoint, Gatekeeper, or Border Element.

The H.323 URL is defined in ABNF as shown below. Note that it utilizes the Core Rules specified in 6.1 of [52].

```
H323-URL          =   "h323:" address [ url-parameters ]
address           =   user / "@" hostport / user "@" hostport
user              =   1*(%x21-24 / %x26-3F / %x41-7F / escaped)
                      ; The symbols "%", "@", and symbols with a
                      ; character value below 0x21 may be represented
                      ; as escaped sequences.
hostport          =   host [ ":" port]
host              =   hostname / IPv4address / IPv6reference
hostname          =   *( domainlabel "." ) toplabel [ "." ]
domainlabel       =   alphanum / alphanum *( alphanum / "-" ) alphanum
toplabel          =   ALPHA / ALPHA *( alphanum / "-" ) alphanum
IPv4address       =   1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference     =   "[" IPv6address "]"
IPV6address       =   hexpart [ ":" IPv4address ]
hexpart           =   hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq            =   hex4 *( ":" hex4 )
hex4              =   1*4HEXDIG
port              =   1*DIGIT
url-parameters    =   *( ";" url-parameter )
url-parameter     =   1*(%x21-24 / %x26-3A / %x3C-7F / escaped)
                      ; Specific parameter definitions are for further
                      ; study.  The symbols "%", ";", and symbols with
                      ; a character value below 0x21 may be
                      ; represented as escaped sequences.
alphanum          =   ALPHA / DIGIT
escaped           =   "%" HEXDIG HEXDIG
```

The *host* is case insensitive.

The *user* is a Unicode [19] string that shall be UTF-8 [57] encoded and then escaped as necessary. Except for characters with a numeric value below 0x80, the *user* is case sensitive. The characters with a numeric value below 0x80 are case insensitive.

The character set and case sensitivity of the *url-parameter* is specified in each parameter definition.

If an endpoint registers with a Gatekeeper and does not provide a *hostport* string, the Gatekeeper may append a *hostport* string to the URL when it returns the endpoint's aliases in an RCF message. The endpoint shall accept the modified alias and use it when sending subsequent requests to the Gatekeeper, including URQ messages to unregister the alias.

7.2 Registration, Admission and Status (RAS) channel

The RAS Channel shall be used to carry messages used in the Gatekeeper discovery and endpoint registration processes which associate an endpoint's alias address with its Call Signalling Channel Transport Address. The RAS Channel shall be an unreliable channel.

Since the RAS messages are transmitted on an unreliable channel, H.225.0 recommends timeouts and retry counts for various messages. An endpoint or Gatekeeper which cannot respond to a request within the specified timeout may use the Request in Progress (RIP) message to indicate that it is still processing the request. An endpoint or Gatekeeper receiving the RIP shall reset its timeout timer and retry counter.

7.2.1 Gatekeeper discovery

Gatekeeper discovery is the process an endpoint uses to determine which Gatekeeper to register with. This may be done manually or automatically. Manual discovery relies on methods outside the scope of this Recommendation to determine which Gatekeeper an endpoint is associated with. The endpoint is configured with the Transport Address of the associated Gatekeeper. For example, it may be entered at endpoint configuration, or it may be entered into an initialization file. In this way, the endpoint knows *a priori* which Gatekeeper it is associated with. The endpoint can now register with that Gatekeeper.

The Automatic method allows the endpoint-Gatekeeper association to change over time. The endpoint may not know who its Gatekeeper is or may need to identify another Gatekeeper due to a failure. This may be done through auto discovery. Auto discovery allows for lower administrative overhead in configuring individual endpoints and additionally allows replacement of an existing Gatekeeper without manually reconfiguring all of the affected endpoints.

The endpoint may multicast (or use other methods as described in Appendix IV/H.225.0) a Gatekeeper Request (GRQ) message, asking "Who is my Gatekeeper?". This is sent to the Gatekeeper's well-known Discovery Multicast Address. One or more Gatekeepers may respond with the Gatekeeper Confirmation (GCF) message indicating "I can be your Gatekeeper" and containing the Transport Address of the Gatekeeper's RAS Channel. If a Gatekeeper does not want the endpoint to register to it, it shall return Gatekeeper Reject (GRJ). See Figure 23. If more than one Gatekeeper responds, the endpoint may choose the Gatekeeper it wants to use. At this point, the endpoint knows which Gatekeeper to register with. The endpoint can now register with that Gatekeeper.

In the event that the endpoint knows the location of the Gatekeeper by some *a priori* means, the endpoint may still choose to unicast the GRQ to the Gatekeeper for the purpose of H.225.0 cryptological exchange.

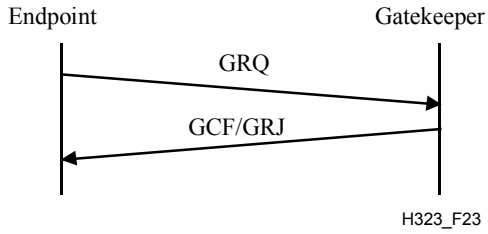


Figure 23/H.323 – Auto discovery

In order to provide redundancy in systems which use a Gatekeeper, the Gatekeeper may indicate alternate Gatekeepers that may be used in the event of a primary Gatekeeper failure. This list of alternate Gatekeepers is provided in the **alternateGatekeeper** field of the GCF and RCF messages.

If no Gatekeeper responds within a timeout, the endpoint may retry the GRQ. An endpoint shall not send a GRQ within 5 s after sending a previous one. If no response is received, the endpoint may use the manual discovery method.

If at any time an endpoint determines it has an invalid registration with its Gatekeeper, it must rediscover its Gatekeeper. The endpoint may assume a registration is invalid if an RRJ is return by a Gatekeeper in response to an RRQ or if no response is received for an RRQ within a timeout.

The GRQ may be repeated periodically (i.e., at endpoint power-up), so the Gatekeeper shall be able to handle multiple requests from the same endpoint.

7.2.2 Endpoint registration

Registration is the process by which an endpoint joins a Zone and informs the Gatekeeper of its Transport Addresses and alias addresses. As part of their configuration process, all endpoints shall register with the Gatekeeper identified through the discovery process. Registration shall occur before any calls are attempted and may occur periodically as necessary (for example, at endpoint power-up).

A Gateway or MCU may register a single Transport Address or multiple Transport Addresses as its call signalling address and may register a single Transport Address or multiple Transport Addresses as its RAS address. The use of multiple Transport Addresses shall indicate a prioritised list of addresses to try when communicating with a given endpoint through either its RAS or Call Signalling Channel.

An endpoint shall send a Registration Request (RRQ) to a Gatekeeper. This is sent to the Gatekeeper's RAS Channel Transport Address. The endpoint has the Network Address of the Gatekeeper from the Gatekeeper discovery process and uses the well-known RAS Channel TSAP Identifier. The Gatekeeper shall respond with either a Registration Confirmation (RCF) or a Registration Reject (RRJ). See Figure 24. An endpoint shall only register with a single Gatekeeper.

The RRQ may be repeated periodically (e.g., at terminal power-up), so the Gatekeeper shall be able to handle multiple requests from the same endpoint. If a Gatekeeper receives an RRQ having the same alias address (or list of alias addresses) and the same Transport Addresses as an active registration, it shall respond with RCF. If a Gatekeeper receives an RRQ having the same alias address (or list of alias addresses) as an active registration and different Transport Addresses, it may confirm the request, if it complies with the Gatekeeper's registration policy. If the request does not comply with the Gatekeeper's registration policy, the Gatekeeper should reject the registration indicating a duplicate or invalid registration. If the Gatekeeper receives an RRQ having the same Transport Addresses as an active registration and a different alias address (or list of alias addresses) and the RRQ is not specified to be an additive RRQ, it should replace the translation table entries. The Gatekeeper may have a method to authenticate these changes.

An endpoint may indicate a backup, redundant, or alternate Transport Addresses using the **alternateEndpoint** structure within the RAS messages. This allows an endpoint to have a secondary network interface or a secondary H.323 endpoint as a backup. The Gatekeeper shall reject ambiguous registrations. The Gatekeeper may reject the registration for other reasons, such as changes in discovery or security issues.

If the endpoint does not include an alias address in the RRQ message, the Gatekeeper may assign one. The Gatekeeper shall return the assigned alias address to the terminal in the RCF message.

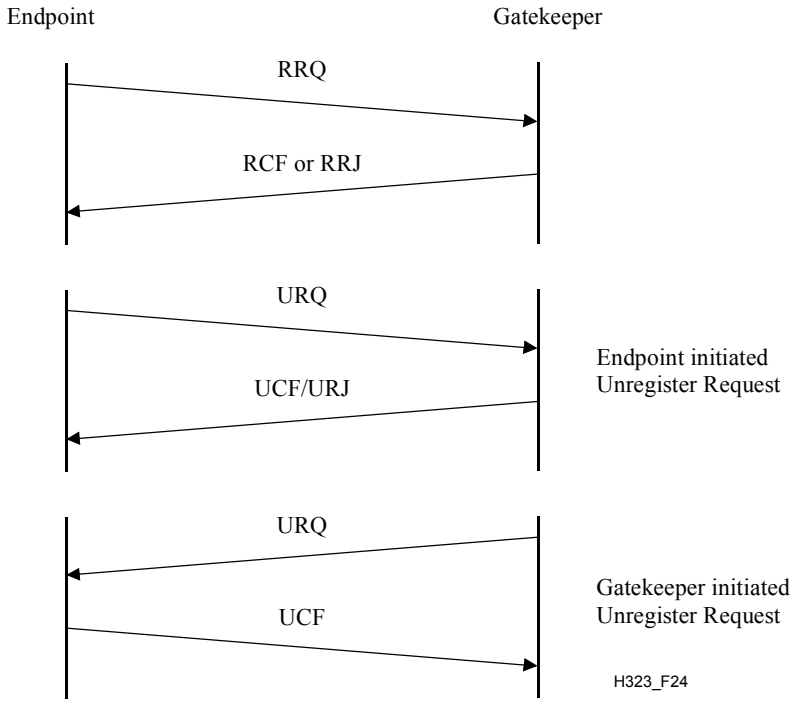


Figure 24/H.323 – Registration

An endpoint may cancel its registration by sending an Unregister Request (URQ) message to the Gatekeeper. This allows an endpoint to change the alias address associated with its Transport Address or vice versa. The Gatekeeper shall respond with either an Unregister Confirmation (UCF) message or an Unregister Reject (URJ) message according to Gatekeeper policy.

If the endpoint sends a URQ message containing a list of alias addresses, the Gatekeeper shall only unregister the listed aliases if it chooses to accept the request. If the endpoint sends a URQ message that does not contain any alias addresses, the Gatekeeper shall unregister all aliases, if any, for the endpoint if it chooses to accept the request.

A Gatekeeper may cancel the registration of an endpoint by sending an Unregister Request (URQ) message to the endpoint. The endpoint shall respond with an Unregister Confirmation (UCF) message. The endpoint shall attempt to re-register with a Gatekeeper prior to initiating any calls. This may require the endpoint to register with a new Gatekeeper.

If the Gatekeeper sends a URQ message containing a list of alias addresses, the endpoint shall assume that only those alias addresses are unregistered. A URQ that contains no aliases shall indicate a request to unregister the endpoint.

An endpoint which is not registered with a Gatekeeper is called an unregistered endpoint. This type of endpoint does not request admission permission from a Gatekeeper and so cannot participate in admissions control, bandwidth control, address translation and other functions performed by the Gatekeeper.

7.2.2.1 Use of lightweight RRQ

An endpoint's registration with a Gatekeeper may have a finite life. An endpoint may request a **timeToLive** in the RRQ message to the Gatekeeper. The Gatekeeper may respond with an RCF containing the same **timeToLive**, a longer **timeToLive**, or a shorter **timeToLive**. If the endpoint cannot accommodate a larger **timeToLive** proposed by the Gatekeeper, the endpoint shall use the largest **timeToLive** value that it can support and that is less than the **timeToLive** proposed by the Gatekeeper. After this time, the registration shall be expired. The **timeToLive** is expressed in

seconds. Prior to the expiration time, the endpoint may send an RRQ message having the **keepAlive** bit set. The keep-alive RRQ may include a minimum amount of information as described in ITU-T Rec. H.225.0. The keep-alive RRQ shall reset the time to live timer in the Gatekeeper, allowing the registration to be extended. After the expiration time, the endpoint must re-register with a Gatekeeper using a full RRQ message.

If the Gatekeeper does not include a **timeToLive** value in the RCF, the registered endpoint shall consider that the Gatekeeper is not supporting the keep-alive mechanism. Endpoints shall not send RRQs with the **keepAlive** field set to Gatekeepers which have indicated that they are not supporting the keep-alive mechanism. A Gatekeeper should not assume that an endpoint supports the keep-alive mechanism if the endpoint does not provide a **timeToLive** value in the RRQ.

Gatekeepers should not treat an RRQ with the **keepAlive** field set as a full registration (i.e., for updating or initializing its translation tables).

Endpoints should consider messaging and processing delays when determining when their registration will expire (i.e., the duration of their own time-to-live timer) at the Gatekeeper.

Expiration of the time-to-live timer in the Gatekeeper results in the expiration of the registration of the endpoint. A Gatekeeper may send a URQ to the endpoint as a notification of such expiration. This allows for loss of synchronization between the time-to-live timers of the Gatekeeper and the endpoint. It also indicates a need for re-registration to endpoints which do not support the keep-alive mechanism.

An endpoint which sends a lightweight RRQ to its Gatekeeper after the time-to-live timer has expired in the Gatekeeper will receive an RRJ response with **rejectReason** of either **fullRegistrationRequired** or **discoveryRequired**, depending on Gatekeeper requirements.

An endpoint which sends an ARQ to its Gatekeeper after the time-to-live timer has expired in the Gatekeeper will receive an ARJ with **rejectReason** of either **callerNotRegistered** or **calledPartyNotRegistered**. An endpoint which initiates a new call through its Gatekeeper after expiration of the Gatekeeper's time-to-live timer will receive a Release Complete message with a reason of **callerNotRegistered** or **calledPartyNotRegistered**.

Disposition of existing calls upon expiration of the time-to-live timer is implementation dependent.

7.2.2.2 Use of additive registrations

Support for additive registrations is optional in both the Gatekeeper and the endpoint. A Gatekeeper that supports additive registrations shall indicate support by including the **supportsAdditiveRegistration** field in the RCF message and shall comply with the procedures set forth in this clause. Additionally, an endpoint shall not use the additive registration procedure described in this clause if the **supportsAdditiveRegistration** field of the RCF is missing.

If the Gatekeeper receives an RRQ with the **additiveRegistration** field included, it shall treat the RRQ as an addition of information to an existing registration for the endpoint specified in the **endpointIdentifier** field. Upon receiving an additive RRQ, the Gatekeeper shall add the alias (or list of aliases) from the **terminalAlias** and **terminalAliasPattern** fields to the existing translation table entries for the endpoint. Also, the Gatekeeper shall add the supported prefixes from the **supportedPrefixes** field of the **terminalType** field to the existing translation table entries for the endpoint. Any previously registered alias addresses or supported prefixes for the endpoint shall remain registered. The Gatekeeper shall replace the endpoint's Call Signalling Addresses and RAS addresses with the values specified in the **callSignalAddress** and **rasAddress** fields, if any are present, and shall replace the endpoint's alternate endpoints with values specified in the **alternateEndpoints** field, if present. The **keepAlive** shall be FALSE if the **additiveRegistration** field is included in the RRQ. However, the receipt of an additive RRQ shall cause the Gatekeeper to restart the endpoint's time to live counter if one is currently running.

An endpoint that sends an additive RRQ to its Gatekeeper when the endpoint is not registered will receive an RRJ response with **rejectReason** of either **fullRegistrationRequired** or **discoveryRequired**, depending on Gatekeeper requirements.

NOTE – As the additive RRQ is not a full registration, the Gatekeeper may ignore fields in the additive RRQ not specifically referenced in this clause.

7.2.3 Endpoint location

An endpoint or Gatekeeper which has an alias address for an endpoint and would like to determine its contact information may issue a Location Request (LRQ) message. This message may be sent to a specific Gatekeeper's RAS Channel TSAP Identifier or may be multicast like the GRQ message to the Gatekeeper's well-known Discovery Multicast Address. The Gatekeeper with which the requested endpoint is registered shall respond with the Location Confirmation (LCF) message containing the contact information of the endpoint or the endpoint's Gatekeeper. Contact information shall include the Call Signalling Channel and RAS Channel addresses to be used to reach the endpoint and optionally additional destination information which can provide dialling information and extension information concerning the requested endpoint.

All Gatekeepers with which the requested endpoint is not registered shall return Location Reject (LRJ) if they received the LRQ on the RAS Channel. Any Gatekeeper with which the requested endpoint is not registered shall not respond to the LRQ, if it received the LRQ on the Discovery Multicast address.

An endpoint or Gatekeeper may include one or more **dialledDigits** or **partyNumber** extensions to which it wishes to connect in the **destinationInfo** field of the LRQ to attempt to locate an available Gateway outside of its zone. A Gatekeeper which receives an LRQ requesting an available Gateway is not obligated to make its Gateways available to such a request.

A Gatekeeper may be aware of the alias address and connection information of endpoints on the SCN. This Gatekeeper could respond to an LRQ requesting information on the SCN endpoint with the connection information necessary to reach that endpoint. This would include the information necessary to address the Gateway as well as the SCN endpoint. Note that the SCN endpoint is not registered with the Gatekeeper in the sense that it exchanges RRQ/RCF messages with the Gatekeeper. The method by which a Gatekeeper becomes aware of the SCN endpoint information is outside the scope of this Recommendation.

7.2.4 Admissions, bandwidth change, status and disengage

The RAS Channel is also used for the transmission of Admissions, Bandwidth Change, Status and Disengage messages. These messages take place between an endpoint and a Gatekeeper and are used to provide admissions control and bandwidth management functions. The detailed use of these messages is described in clause 8.

The Admission Request (ARQ) message specifies the requested Call Bandwidth. This is an upper limit on the aggregate bit rate for all transmitted and received, audio and video channels excluding any RTP headers, RTP payload headers, network headers, and other overhead. Data and control channels are not included in this limit. The Gatekeeper may reduce the requested Call Bandwidth in the Admission Confirm (ACF) message. An endpoint shall assure that the aggregate bit rate, averaged over one second, for all transmitted and received, audio and video channels is at or below the Call Bandwidth. An endpoint or the Gatekeeper may attempt to modify the Call Bandwidth during a call using the Bandwidth Change Request (BRQ) message.

The Admission Confirm Sequence message allows the Gatekeeper to provide a single reply to an ARQ containing alternate routing information, different source information, different tokens, etc. When an endpoint receives an Admission Confirm Sequence message containing more than one ACF inside, it shall process the first ACF in the sequence by attempting to establish the call as described in this Recommendation. In the event that the endpoint is unable to establish the call due

to some unexpected failure, the endpoint may then select the next ACF message in the sequence and re-attempt the call establishment without first consulting the Gatekeeper. Without limiting the definition, "unexpected failures" may include busy circuits; transport routing problems (e.g., "no route to host"); or exhausted gateway resources. It is the endpoint's decision as to whether it wishes to make call attempts to alternate routes in the face of a routing failure.

Endpoints that choose to support the Admission Confirm Sequence message shall indicate this capability by setting the **acfSequences** field in the RRQ message to TRUE. The Gatekeeper shall consider absence of this field as a FALSE value. Gatekeepers shall not send the Admission Confirm Sequence message to an endpoint that has not indicated support for this message in the RRQ. An endpoint may change the value of the **acfSequences** field in subsequent RRQ messages. In the event that the endpoint changes this value from TRUE to FALSE, the endpoint shall be prepared to receive Admission Confirm Sequence messages that might be in transit as a result of having previously advertised support of Admission Confirm Sequence messages.

As it is that the Admission Confirm Sequence is merely a means of providing alternate routing information that could not be provided in an Admission Confirm message, this Recommendation makes no further distinction elsewhere as to the semantic difference between the Admission Confirm message and the Admission Confirm Sequence message. Throughout this Recommendation, "Admission Confirm" or "ACF" refers to either a single Admission Confirm message or an Admission Confirm Sequence message.

7.2.5 Access tokens

An Access Token is a string passed in some RAS messages and the Setup message. The Access Tokens have two uses. First, they can provide privacy by shielding an endpoint's Transport Address and Alias Address information from a calling party. A user may give out only the Access Token for a calling party to use in reaching the endpoint. The Gatekeeper will know the endpoint related to the Access Token from the registration process, so that calls using the Access Token can be routed through the Gatekeeper to the called endpoint. The use of the access token only applies to the Gatekeeper routed call model when attempting to hide the Transport Address from the endpoint.

The second use of the Access Token is in ensuring that calls are routed properly through H.323 entities. An Access Token returned by a Gatekeeper shall be used in any subsequent setup messages sent by the endpoint. This Access Token may be used by a Gateway to assure that the endpoint has permission to use the Gateway resources, or it may be used by a called endpoint to assure that the calling endpoint can signal it directly.

The Access Token may also be distributed by out-of-band methods to assure proper access to Gateways and endpoints in systems that do not have Gatekeepers.

7.2.6 Alternate gatekeeper procedures

For the purposes of ensuring system availability, redundancy, and scalability, the Gatekeeper may provide the RAS signalling function by utilizing multiple physical or logical devices, referred to as Alternate Gatekeepers. If the endpoint supports the Alternate Gatekeeper procedures defined in this clause, it should include the **supportsAltGK** field in the GRQ and RRQ messages.

When an endpoint initiates communication with the Gatekeeper, it may be provided with a list of Alternate Gatekeepers via the GCF message. If the Gatekeeper does not respond to the subsequent RRQ, the endpoint shall attempt to register with the Gatekeeper using the list of Alternate Gatekeepers provided in the GCF. If no Alternate Gatekeeper responds, the endpoint shall reinitiate the Gatekeeper discovery process.

If the endpoint receives a GRJ message containing Alternate Gatekeeper information and does not receive a GCF message, the endpoint shall send GRQ messages to one or more Alternate Gatekeepers in the list of Alternate Gatekeepers received in the GRJ. If multiple GRJ messages are received, the endpoint may select any one GRJ message from which to extract Alternate Gatekeeper

information. If no Alternate Gatekeeper sends a GCF message, the endpoint may attempt to use any new Alternate Gatekeeper lists received for the purpose of Gatekeeper discovery or it may reinitiate the Gatekeeper discovery process.

If the endpoint has not yet registered with the Gatekeeper or has reinitiated the Gatekeeper discovery process, it shall ignore the **needToRegister** field in the Alternate Gatekeeper list and assume that the value is TRUE.

If the endpoint is registered with the Gatekeeper and the Gatekeeper becomes unresponsive, the endpoint shall attempt to communicate with an Alternate Gatekeeper.

The Gatekeeper may explicitly redirect an endpoint to an Alternate Gatekeeper by returning a RAS rejection message with a list of Alternate Gatekeepers. If the **altGKisPermanent** field is set to FALSE in such a redirection, the redirection is considered temporary, as it only applies to a single RAS message.

A Gatekeeper may send a URQ to an endpoint with a list of Alternate Gatekeepers, in which case the endpoint shall respond with a UCF and attempt to communicate with an Alternate Gatekeeper. An endpoint shall not include a list of Alternate Gatekeepers in any URQ message that it sends.

An endpoint shall keep only one list of Alternate Gatekeepers. That list shall be taken from the most recently received list of Alternate Gatekeepers received in any RAS message, with one exception: if the endpoint is temporarily redirected to an Alternate Gatekeeper and the Alternate Gatekeeper returns a rejection message with a list of Alternate Gatekeepers (even if the list is empty), the endpoint shall interpret the rejection as a redirection. The endpoint may ignore the list of Alternate Gatekeepers provided in such a redirection and continue using the list of Alternate Gatekeepers received in the original rejection message.

If the Gatekeeper wishes to clear the endpoint's list of Alternate Gatekeepers, such as when the Gatekeeper is reconfigured to not use Alternate Gatekeepers, it shall return an empty list of Alternate Gatekeepers to the endpoint in the RCF message.

The endpoint shall use the **priority** field to indicate the order in which to communicate with Alternate Gatekeepers. If multiple Alternate Gatekeepers are specified to have the same **priority**, the endpoint may order the Alternate Gatekeepers with the same **priority** value as it chooses.

When an endpoint is redirected to a temporary Alternate Gatekeeper, it shall ignore the **needToRegister** field and assume the value is FALSE and retransmit only the redirected RAS message to a temporary Alternate Gatekeeper. All other RAS messages shall continue to be sent to the Gatekeeper as usual. Note that this does not preclude the Gatekeeper from temporarily redirecting an endpoint to an Alternate Gatekeeper by returning an RRJ to either an RRQ or a lightweight RRQ.

If distinct RAS requests are redirected to temporary Alternate Gatekeepers, each distinct message shall be sent to one and only one temporary Alternate Gatekeeper at a time, although different RAS messages may be sent to different temporary Alternate Gatekeepers simultaneously. If the endpoint determines that a temporary Alternate Gatekeeper is unresponsive, it shall attempt to retransmit the RAS request to another Alternate Gatekeeper. If all Alternate Gatekeepers are unresponsive to a RAS request, the endpoint shall assume that the RAS request is rejected. If the request was an RRQ, the endpoint shall reinitiate the Gatekeeper discovery process.

If the Gatekeeper becomes unresponsive or if the Gatekeeper redirects the endpoint by returning a list of Alternate Gatekeepers with the **altGKisPermanent** field set to TRUE, the endpoint shall attempt to communicate with an Alternate Gatekeeper. The endpoint shall attempt communication with only one Alternate Gatekeeper. Only after the endpoint determines that an Alternate Gatekeeper is unresponsive shall it attempt to communicate with the next Alternate Gatekeeper. If all Alternate Gatekeepers are unresponsive, the endpoint shall reinitiate the Gatekeeper discovery process. If registration is required with an Alternate Gatekeeper, the endpoint shall first attempt to

send an RRQ to the Alternate Gatekeeper, rather than a GRQ. Only if the Gatekeeper returns an RRJ with the reason **discoveryRequired** shall the endpoint send a GRQ to the Alternate Gatekeeper. When permanently transitioning to an Alternate Gatekeeper, the endpoint shall send all further RAS messages to the Alternate Gatekeeper, including outstanding RAS requests that timeout. The endpoint should reset the retry counters for any outstanding RAS messages before transmitting them to the Alternate Gatekeeper for the first time.

If an Alternate Gatekeeper to which an endpoint is redirected returns a rejection message without a list of Alternate Gatekeepers, the endpoint shall accept the message as a rejection to the original request. If the rejection was to an RRQ, the endpoint shall reinitiate the Gatekeeper discovery process. If the Alternate Gatekeeper redirects the endpoint by returning a rejection message with a list of Alternate Gatekeepers, the endpoint shall attempt to send the request to another Alternate Gatekeeper. If all Alternate Gatekeepers redirect the endpoint, the endpoint shall ultimately assume that the request is rejected.

An endpoint shall not send a URQ message when transitioning between Alternate Gatekeepers, even if the **needToRegister** field is TRUE, except in the case where the Gatekeeper sends a URQ with a list of Alternate Gatekeepers.

If an endpoint is redirected to an Alternate Gatekeeper that is specified to be permanent (i.e., the **altGKisPermanent** field is TRUE) or was forced to begin communicating with an Alternate Gatekeeper after its Gatekeeper became unresponsive, it shall assume that the Alternate Gatekeeper is prepared to accept requests relating to existing calls. It shall send all subsequent BRQ, DRQ, and IRR messages relating to existing calls to the Alternate Gatekeeper. Likewise, the Alternate Gatekeeper shall be prepared to handle such messages.

If an endpoint begins communicating with an Alternate Gatekeeper with which registration was not required, including temporary Alternate Gatekeepers, the **gatekeeperIdentifier** field of the URQ, ARQ, BRQ, LRQ, and DRQ messages shall contain the **gatekeeperIdentifier** of the Alternate Gatekeeper from the Alternate Gatekeeper list. This field may not be present when registration is required.

7.2.7 Usage information reporting

An endpoint may have the ability to collect and report call usage information, which may be useful for accounting or billing purposes. A Gatekeeper may request that an endpoint report this information. This feature is intended to interwork with the usage information reporting features of systems that implement Annex G/H.225.0.

Note that this feature is intended for scenarios in which the endpoint from which the usage information is requested is trusted, such as when a gateway and Gatekeeper are administered by the same service provider. That is, it is assumed that the endpoint will accurately report its usage information.

7.2.7.1 Advertising usage information reporting capabilities

An endpoint may advertise to a Gatekeeper its ability to collect and report usage information. It specifies these capabilities in the **usageReportingCapability** field of the RRQ message. If the endpoint has reported its capabilities and these capabilities subsequently change, the endpoint shall send another RRQ specifying its capabilities. Absence of a **usageReportingCapability** field in an RRQ indicates that the endpoint cannot report usage information.

7.2.7.2 Requesting usage information reports

A Gatekeeper may request usage information from an endpoint via the RCF, ACF and IRQ messages. A Gatekeeper should assume that an endpoint that has not advertised the ability to report a particular type of usage information will not report that information, and it should not request that information from the endpoint.

A Gatekeeper may request usage information via the **usageSpec** field of the RCF message. This request is referred to as the "default" **usageSpec**. By including this field, the Gatekeeper is requesting that the endpoint collect and report the specified usage information for all new calls. This request does not apply to calls that are already in progress.

Once a Gatekeeper has delivered a default **usageSpec** via the RCF, it assumes that this request remains in effect until it delivers another default **usageSpec**. If the Gatekeeper does not wish to change a previously delivered default **usageSpec**, it may indicate this by not including the **usageSpec** in when sending an RCF message. In order to change a previously delivered default request for usage information, a Gatekeeper shall send a new **usageSpec** in its next RCF message. In order to request that an endpoint stop reporting usage information, a Gatekeeper shall send a **usageSpec** with no options selected in either the **when** or **required** fields.

A Gatekeeper may request usage information for a particular call via the **usageSpec** field of the ACF message for that call. This request is referred to as the "per-call" **usageSpec**. If provided, this request overrides, for that call, any default usage specification that the Gatekeeper may have provided in an RCF message.

A Gatekeeper may also request usage information for a particular call via the **usageInfoRequested** field of an IRQ message. The response to this request should immediately follow in an IRR message. This request does not affect either the default usage specification sent via the RCF or the per-call usage specification sent via the ACF.

A Gatekeeper that wishes an endpoint to report usage information periodically in unsolicited IRR messages shall indicate this request by selecting the **inIrr** option of the **when** field of the **usageSpec**. It shall also specify either the **irrFrequencyInCall** in the **preGrantedARQ** field of the RCF message, or the **irrFrequency** in the ACF message, as appropriate for a particular call.

A Gatekeeper that requests that usage information be reported at the start of a call or in unsolicited IRR messages (i.e., that selects the **start** or **inIrr** options in the **when** field of the **usageSpec**) should acknowledge IRR messages in order to ensure that the requested usage information is reliably delivered. To indicate that it will acknowledge IRR messages, the Gatekeeper sets the **willRespondToIRR** field of the RCF or ACF message to TRUE.

7.2.7.3 Sending usage information reports

An endpoint may report usage information to a Gatekeeper via the BRQ, IRR and, DRQ and DCF messages. An endpoint may send usage information to a Gatekeeper that has not requested that information. If an endpoint advertises the ability to collect and report a particular type of usage information, and a Gatekeeper requests that information, then the endpoint shall report the requested information. An endpoint shall ignore requests for usage information that are erroneous (such as a request to provide the call end time at the start of a call). An endpoint may ignore a request for usage information that does not fall within the endpoint's advertised reporting capabilities.

If a Gatekeeper sends an endpoint a default **usageSpec** in an RCF message, the endpoint shall set the usage information reporting parameters for all new calls based on this template, unless the Gatekeeper supplies a per-call **usageSpec** for a particular call in an ACF message. If provided, the per-call **usageSpec** overrides the default **usageSpec** for that call. An endpoint may apply an updated default **usageSpec** to existing calls for which no per-call **usageSpec** was provided.

An endpoint shall interpret a **usageSpec** with no options selected in either the **when** or the **required** fields as a request not to report usage information.

When reporting usage information via an IRR message, and the Gatekeeper has indicated via the **willRespondToIRR** field of either the RCF or ACF that it will acknowledge IRRs, an endpoint shall set the **needResponse** field to TRUE and retransmit the information if an acknowledgement is not received. This rule shall apply whether the IRR is solicited or unsolicited.

If the Gatekeeper has requested that usage information be reported at the start of the call (i.e., it selected **start** in the **when** field of the **usageSpec**), and the requested information is within the endpoint's advertised capabilities to report, then the endpoint shall report the requested information immediately after the start of the call. If the endpoint sends a BRQ at this point in time, then it may include the requested usage information in the **usageInformation** field of the BRQ message. Otherwise, the endpoint shall send an unsolicited IRR message with the requested usage information in the per-call **usageInformation** field.

If the Gatekeeper has requested that usage information be reported at the end of the call (i.e., it selected **end** in the **when** field of the **usageSpec**), and the requested information is within the endpoint's advertised capabilities to report, then the endpoint shall report the requested information immediately after the end of the call in the DRQ message (or in the DCF if the call is terminated by the Gatekeeper).

If the Gatekeeper has requested that usage information be reported in unsolicited IRR messages (i.e., it selected **inIrr** in the **when** field of the **usageSpec**), and the requested information is within the endpoint's advertised capabilities to report, then the endpoint shall report the requested information in every unsolicited IRR that it sends.

The endpoint shall apply neither the default nor the per-call **usageSpec** when sending solicited IRR messages (i.e., responses to IRQs). If the Gatekeeper requests usage information via the **usageInfoRequested** field of the IRQ, and it is within the endpoint's advertised capabilities to report this information, then the endpoint shall report the requested information in the per-call **usageInformation** field of the IRR. If the Gatekeeper does not request usage information in the IRR, the endpoint should not include a **usageInformation** field in the response.

7.2.8 Call credit-related capabilities

By utilizing the optional credit-related capabilities, an endpoint can receive a user's credit or debit information from the Gatekeeper before and after the user establishes a call. In turn, the endpoint may relay this information to the end user via an announcement. The endpoint also has the option to limit the user's call duration to an amount of time specified by the Gatekeeper. For instance, the endpoint may disengage the call when the time or money on the user's account is exhausted.

In addition, the Gatekeeper may send balance-related announcements to the endpoint and may indicate a call duration limit to the endpoint.

7.2.8.1 Endpoint advertisement of credit-related capabilities

The endpoint indicates its support for the call credit features via the RRQ. The ability to play or display announcements regarding a caller's balance may be advertised via a new **supportedH248Packages** field. The **supportedH248Packages** field consists of an optional list of **H248PackagesDescriptors** in binary format.

To send a text announcement, the endpoint and Gatekeeper may use the "Display" package (**PackageID** dis, 0x0014), defined in Annex G/H.248. Annex G/H.248 includes facilities to control the location of the text on a terminal display and other functions.

To send the index of either a fixed or a parameterized voice announcement that is locally stored at the endpoint, the endpoint and Gatekeeper may use the "Generic Announcement" package (**PackageID** an, 0x001D) defined in Annex K/H.248.

As an alternative to the use of H.248 packages, the endpoint may indicate via H.225.0 call signalling that it is capable of including the user's balance in a text announcement that it constructs itself. This capability may be indicated via the **canDisplayAmountString** flag.

The endpoint may indicate via the **canEnforceDurationLimit** flag whether it can perform its own call timing.

7.2.8.2 Balance information sent by the gatekeeper to the endpoint

The Gatekeeper may send announcements (which could be either voice or text) to the endpoint via an H.248 "signal" in the **ServiceControlDescriptor** structure in the ACF, SCI, and/or DRQ messages. Alternatively, the Gatekeeper may send a text string to the endpoint in the **amountString** field that indicates the account balance, for example "$10.50", in the appropriate currency. In this case, the endpoint is responsible for embedding the amount string in an announcement (for example, "Current debit card balance: $10.50") that is appropriate for that particular endpoint. Note that ISO 4217 defines standard abbreviations for currency types, such as "USD" for United States dollars. The **amountString** field shall be encoded in Unicode.

A **billingMode** field is also added to allow the Gatekeeper to indicate the billing mode for the call. A mode of **debit** indicates that the call will result in charges against the amount of money available in a user's account. A mode of **credit** indicates that the call will result in charges to be paid by the user at a later time. An endpoint may use this information, for example, to determine the type of announcement to play or display.

The **callDurationLimit** field of the **CallCreditServiceControl** structure indicates the remaining amount of time allowed for a particular call. The **enforceCallDurationLimit** flag indicates whether timing enforcement shall be performed by the endpoint. The **callStartingPoint** field indicates the point in the call that timing shall begin if call duration enforcement is provided by the endpoint.

If the endpoint has advertised that it is capable of enforcing the time limit and the Gatekeeper requests that the endpoint enforce the limit, then the endpoint shall disengage the call when the time limit expires. Timing of the call duration shall begin upon transmission or reception of the Connect message or the Alerting message as indicated by the **callStartingPoint field**.

7.2.9 Alternate transport addresses

An endpoint may indicate support for alternate transport protocols by providing the **alternateTransportAddresses** field in the RRQ message. The Gatekeeper may instruct the endpoint as to which signalling transport protocol to use for making calls by including the **useSpecifiedTransport** field in the RCF or ACF message. The Gatekeeper shall include in the **useSpecifiedTransport** field only those protocols for which the endpoint has indicated its support. The endpoint, upon receipt of the **useSpecifiedTransport** field, shall use the specified transport to establish the call.

The Gatekeeper may give the endpoint a choice of transport protocols to use for call signalling by including the **alternateTransportAddresses** field in the RCF or ACF message without including the **useSpecifiedTransport** field. In this case the endpoint shall either use the protocol specified in the **destCallSignalAddress** field or select among the transports indicated in the **alternateTransportAddresses** field.

The Gatekeeper may also provide the **alternateTransportAddresses** of and endpoint registered with it to an H.323 entity in an LCF message.

7.3 Call signalling channel

The Call Signalling Channel shall be used to carry H.225.0 call control messages. The Call Signalling channel shall be a reliable channel.

In networks that do not contain a Gatekeeper, call signalling messages are passed directly between the calling and called endpoints using the Call Signalling Transport Addresses. In these networks, it is assumed that the calling endpoint knows the Call Signalling Transport Address of the called endpoint and thus can communicate directly.

In networks that do contain a Gatekeeper, the initial admission message exchange takes place between the calling endpoint and the Gatekeeper using the Gatekeeper's RAS Channel Transport Address. Within the initial admissions message exchange, the Gatekeeper indicates in the ACF message whether to send the call signalling directly to the other endpoint or to route it through the Gatekeeper. The call signalling messages are sent to either the endpoint's Call Signalling Transport Address or the Gatekeeper's Call Signalling Transport Address.

The Call Signalling Channel may carry signalling for many concurrent calls, using the Call Reference Value to associate the message with the call. An entity indicates its ability to handle multiple concurrent calls on the same call signalling connection by setting the **multipleCalls** flag to TRUE in messages that it sends on the Call Signalling Channel. An entity may dynamically set the value of the **multipleCalls** field in order to indicate its present ability to support multiple connections along the Call Signalling Channel. If an endpoint wishes to change the value of **multipleCalls** at a time when no other H.225.0 messages are being exchanged across the Call Signalling Channel, it shall transmit the **multipleCalls** field via a Facility message with the CRV set to the Global Call Reference as shown in Figure 4-5/Q.931 and **guid** in the **callIdentifier** field set to all zeros.

An entity that is capable of processing multiple concurrent calls on the Call Signalling Channel may indicate that it will support no additional calls on the signalling channel by sending Release Complete with **newConnectionNeeded** as the **reason**. An entity that receives Release Complete with **newConnectionNeeded** can attempt to connect a new Call Signalling Channel.

An entity may transmit a Status Inquiry message that is not related to a specific call. In such cases, the entity shall set the **callIdentifier** field to all zeros. An entity shall not omit the **Status-UUIE** in the Status message or the **StatusInquiry-UUIE** in the Status Inquiry message when transmitting those messages, but entities shall be prepared to receive messages not containing those message elements in order to maintain backward compatibility.

The Call Signalling Channel may be established prior to the actual need to signal a call, and the channel may remain connected between calls. An entity may indicate this capability by setting the **maintainConnection** flag to TRUE in messages that it sends on the Call Signalling Channel. In addition, an endpoint which has this capability should indicate this when it registers with a Gatekeeper. This will allow a Gatekeeper that utilizes Gatekeeper routing to connect to the endpoint at any point after registration. If the connection drops while no call or signalling is active, neither end shall attempt to open the connection until signalling is needed.

The value of the **maintainConnection** flag sent by an entity over a given Call Signalling Channel shall be the same for every message containing this field for the duration of the Call Signalling Channel. This does not preclude an entity from setting this value to TRUE for one Call Signalling Channel and FALSE for another Call Signalling Channel.

ITU-T Rec. H.225.0 specifies the mandatory Q.931 messages that are used for call signalling in this Recommendation. Clause 8 specifies the procedures for using them.

7.3.1 Call signalling channel routing

Call signalling messages may be passed in two ways. The first method is Gatekeeper routed call signalling (see Figure 25). In this method, call signalling messages are routed through the Gatekeeper between the endpoints. The second method is Direct Endpoint Call Signalling (see Figure 26). In this method, the call signalling messages are passed directly between the endpoints. The choice of which methods is used is made by the Gatekeeper.

Both methods use the same kinds of connections for the same purposes and the same messages. Admission messages are exchanged on RAS channels with the Gatekeeper, followed by an exchange of call signalling messages on a Call Signalling channel. This is then followed by the establishment of the H.245 Control Channel. The actions of the Gatekeeper in response to the

admission messages determine which call model is used; this is not under the control of the endpoint, although the endpoint can specify a preference.

The symmetrical signalling method of Annex D/Q.931 shall be used for all mandatory call signalling procedures. This does not address the role that a Gateway might play on the SCN side using Q.931 or other call signalling protocols.

The Gatekeeper Clouds in Figures 25 through 28 contain one or more Gatekeepers which may or may not communicate with each other. The endpoints may be connected to the same Gatekeeper or to different Gatekeepers.

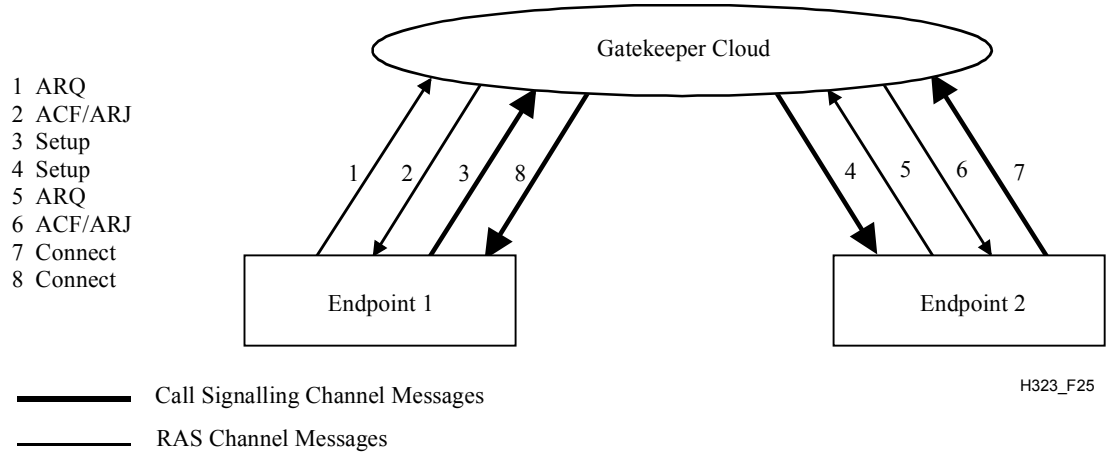


Figure 25/H.323 – Gatekeeper routed call signalling

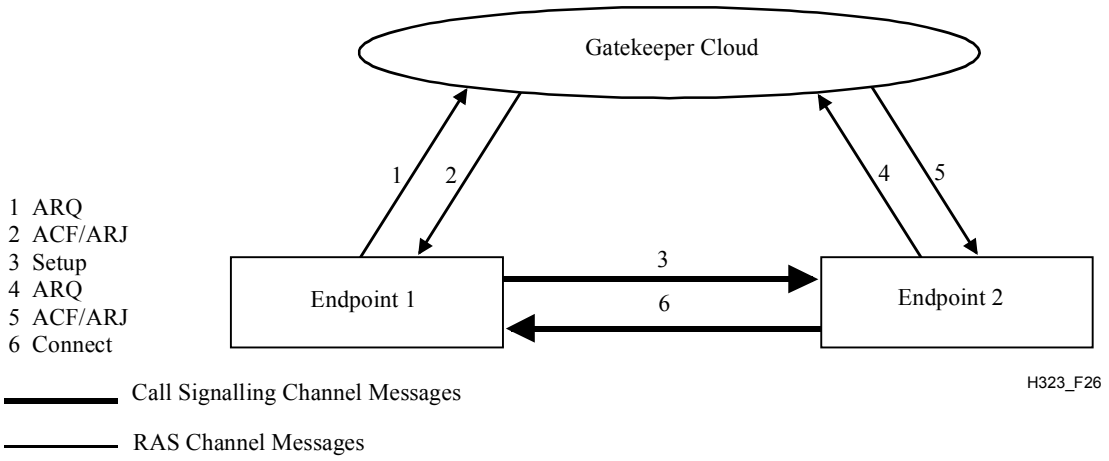


Figure 26/H.323 – Direct endpoint call signalling

7.3.2 Control channel routing

When Gatekeeper routed call signalling is used, there are two methods to route the H.245 Control Channel. In the first method, the H.245 Control Channel is established directly between the endpoints. See Figure 27. This method is for further study. In the second method, the H.245 Control Channel is routed between the endpoints through the Gatekeeper. See Figure 28. This method allows the Gatekeeper to redirect the H.245 Control Channel to an MC when an ad hoc multipoint conference switches from a point-to-point conference to a multipoint conference. The Gatekeeper makes this choice. When Direct Endpoint call signalling is used, the H.245 Control Channel can only be connected directly between the endpoints.

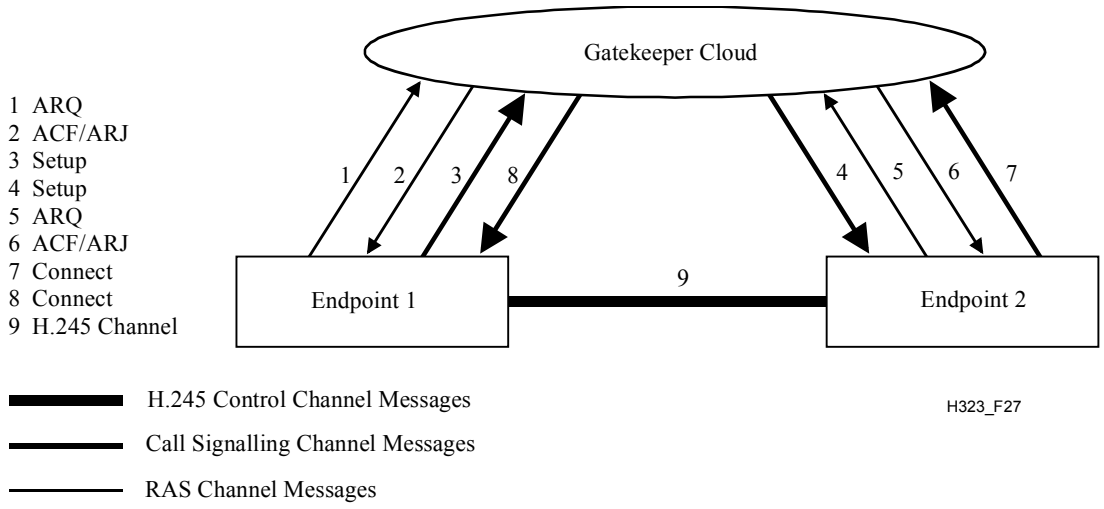


Figure 27/H.323 – Direct H.245 control channel connection between endpoints

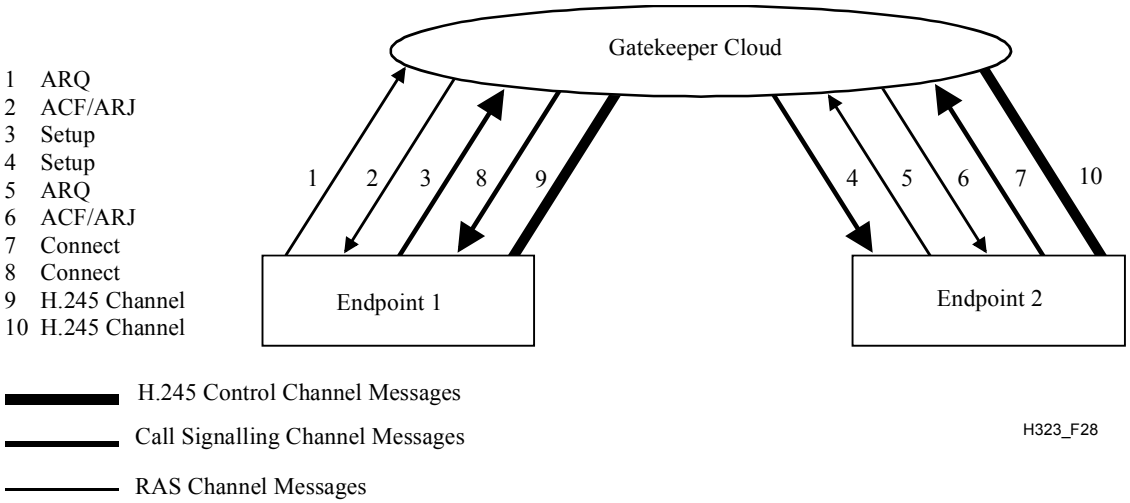


Figure 28/H.323 – Gatekeeper routed H.245 control

7.3.3 Call signalling and control protocol revisions

When a call is routed through a Gatekeeper, Gatekeepers shall use the following rules to determine the H.225.0 or H.245 version number to be indicated in messages originated by an endpoint and routed or forwarded by the Gatekeeper:

a) If the originating endpoint's H.225.0 or H.245 version number is less than or equal to the Gatekeeper's version number, and the Gatekeeper chooses to proxy the functions of an equal or later version number on behalf of the originating endpoint, the routed messages shall reflect the version number of the gatekeeper. Otherwise they shall reflect the version number of the originating endpoint.

b) If the originating endpoint's version number is greater than Gatekeeper's, the routed messages shall reflect the version number of the Gatekeeper.

In all cases, the Gatekeeper may use a single ASN.1 encoding specified by the most recent H.225.0 or H.245 version understood by the Gatekeeper according to these rules.

Since some features in the H.323, such as third party pause and re-routing, require that the signalling entities know exactly what version of the protocol is being used by the other entities in a call and because the **protocolIdentifier** may change after receiving the first call signalling message

and at other times during the call, such as when a call is re-routed to a different entity, entities that rely on version-specific features should determine the version of the other entities in a call by examining the **protocolIdentifier** in the Setup and Connect message at the very least. During a call, a call may be re-routed to a different entity that uses a different version of the protocol. In such a case, entities that rely on version-specific features should again determine the version of the entity to which the call may have been switched. If H.245 signalling is tunnelled, the endpoint may use the call signalling message containing the tunnelled non-empty terminal capability set message in order to determine the version of the remote endpoint. If a separate H.245 Channel is used, an entity may send a Status Inquiry message and determine the protocol version by examining the **protocolIdentifier** in the resulting Status message. In either case, the version of H.245 used by the other entity is signalled in the non-empty capability set message.

It should be noted that H.323 entities prior to version 4 may not provide the **protocolIdentifier** in the Status message, so H.323 entities shall assume that the absence of the **protocolIdentifier** indicates only that the entity is older than version 4.

NOTE – A Gatekeeper may signal its own protocol version when replying to a Setup message (e.g., to send a Call Proceeding message prior to establishing communication with the called party) or when initiating an outbound connection independent of an existing call. Therefore, it is important that an endpoint not rely on the initial message(s) to determine the protocol revision of the remote endpoint.

7.4 Call reference value

All call signalling and RAS messages contain a Call Reference Value (CRV). Refer to ITU-T Rec. H.225.0. There is one CRV for the Call Signalling Channel and an independent CRV for the RAS channel. One CRV is used to associate the call signalling messages. This CRV shall be used in all call signalling messages between two entities (endpoint to Gatekeeper, endpoint-to-endpoint, etc.) related to the same call. A second CRV is used to associate the RAS messages. This CRV shall be used in all RAS messages between two entities related to the same call. New CRVs shall be used for new calls. A second call from an endpoint to invite another endpoint into the same conference shall use new CRVs. The CRV is not the same as the Call ID or the Conference ID (CID). The CRV associates call signalling or RAS messages between two entities within the same call, the Call ID associates all messages between all entities within the same call, and the CID associates all messages between all entities within all calls in the same conference.

The Global Call Reference, as shown Figure 4-5/Q.931 and having the numeric value 0, is used to refer to all calls on the Call Signalling Channel or the RAS channel. When initiating or accepting calls, H.323 entities shall select a CRV value other than Global Call Reference value; the Global Call Reference is reserved for messages which do not pertain to a particular call.

When placing a new call, the calling endpoint shall select a new CRV for the call. The calling endpoint shall use the same CRV on both the RAS channel and the H.225.0 Call Signalling Channel. The called endpoint, however, shall not use the CRV value received in the Setup when communicating on its RAS channel. Instead, the called endpoint shall select a new CRV for use the RAS channel that is unique on that channel without regard to the CRV received in the Setup, though they may happen to be numerically equivalent as a matter of course.

7.5 Call ID

The Call ID is a globally unique non-zero value created by the calling endpoint and passed in various H.225.0 messages. The Call ID identifies the call with which the message is associated. It is used to associate all RAS and Call Signalling messages related to the same call. Unlike CRV, the Call ID does not change within a call. All messages from the calling endpoint to its Gatekeeper, the calling endpoint to the called endpoint, and the called endpoint to its Gatekeeper related to the same call shall contain the same Call ID. The Call ID is encoded as described in ITU-T Rec. H.225.0. In

reference to Figures 29 through 39 in clause 8, all messages within a figure shall have the same Call ID.

When a Version 1 endpoint calls a Version 2 endpoint, it is the responsibility of the Version 2 endpoint to generate a Call ID prior to sending ARQ to its Gatekeeper.

7.6 Conference ID and conference goal

The Conference ID (CID) is a unique non-zero value created by the calling endpoint and passed in various H.225.0 messages. The CID identifies the conference with which the message is associated. Therefore, messages from all endpoints within the same conference will have the same CID. The CID is encoded as specified in ITU-T Rec. H.225.0.

The **conferenceGoal** indicates the intention of the call. Choices are: **create** – to create a new conference, **join** – to join an existing conference, **invite** – to invite a new endpoint into an existing conference, **capability-negotiation** – negotiate capabilities for a later H.332 conference, and **callIndependentSupplementaryService** – transport of supplementary services APDUs.

7.7 Endpoint call capacity

Call capacity indicates an endpoint's acceptance capacity for each type of call the endpoint supports (e.g., voice, T.120 data, H.320, etc.). While any endpoint type may report call capacity through various H.225.0 messages in order to assist a Gatekeeper in routing calls, call capacity information should be reported by Gateways to assist the Gatekeeper with load balancing across Gateways and to help reduce the number of failed call attempts.

The endpoint's maximum and current capacity may be indicated at registration. In addition, the current capacity may also be indicated on a per-call basis. Representing this dynamic capacity requires consideration of these call models:

- Direct call model with per-call admission – In this case, the endpoint may indicate capacity remaining in the ARQ, DRQ, or BRQ messages.
- Direct call model with pre-granted admission – In this case, the endpoint may indicate capacity in RRQ or RAI messages (in the case that the endpoint is a Gateway).
- Gatekeeper routed call model with per-call admission – The endpoint may provide capacity information in an ARQ, DRQ, or BRQ messages.
- Gatekeeper routed call model with pre-granted admission – The endpoint may include capacity information in the call signalling messages, such as Setup or Release Complete. In this case, the originating endpoint may provide its capacity information in a Setup, while the terminating endpoint may provide its capacity information in an Alerting or Connect. Each endpoint may provide updated capacity information using the Release Complete message.

In any case, a Gatekeeper may use the IRQ/IRR exchange to audit an endpoint to potentially discover the call capacity of the endpoint. It should be noted that including capacity information in messages that are already required to be sent to a Gatekeeper, such as an ARQ when not using pre-granted admission or a Setup in a Gatekeeper routed call, rather than sending additional messages for this purpose is preferred. However, if a Gateway receives a Release Complete and is operating in a pre-granted admission mode, it should send an IRR to the Gatekeeper to enable it to maintain more accurate capacity information.

If an endpoint provides call capacity information, it should provide capacity information in an RRQ and should indicate its call capacity reporting capabilities in the RRQ. A Gatekeeper may request via the RCF and IRQ messages that an endpoint provide call capacity information. An endpoint that has indicated the ability to report call capacity shall report its capacity as requested by the gatekeeper. Other than in the initial RRQ, an endpoint should not report maximum call capacity

unless its gatekeeper requests call capacity information in an IRQ message. An endpoint may use capacity information in a BRQ, IRR, or RAI to inform the Gatekeeper of sudden changes, such as that caused by a hardware failure.

An endpoint may signal that it has different call capacities for different supported protocols (i.e., T.120, H.320, H.321, voice, etc.). However, since equipment manufacturers may utilize the same resources for multiple protocols, the Gatekeeper should make no assumptions about how the endpoint's call capacity for one supported protocol may change when the endpoint engages in a call utilizing a different protocol.

A Gateway may signal the call capacity by **group** where the **group** could represent a set of circuits associated with a particular interface or a carrier, for example. This feature allows the Gatekeeper to track the call capacity separately for each group. The **group** may be the same as that reported in the **circuitID** for a particular call.

NOTE – The capacity information reported in any message is of an advisory nature and, due to race conditions, sudden changes in the endpoint, or local allocation of resources, may not be absolutely accurate.

7.8 Caller identification services

7.8.1 Description of services

This clause describes the caller identification services, which includes:

- Calling party number presentation and restriction.
- Connected party number presentation and restriction.
- Called (Alerting) party number presentation and restriction.
- Busy party number presentation and restriction.

7.8.1.1 Calling party address presentation

Calling party address presentation is a feature which provides the alias address of the calling party to the called party. The calling party address may be provided by the calling endpoint or by the Gatekeeper for Gatekeeper routed calls that originate in the packet network. When the call is routed through the Gatekeeper with which the calling endpoint is registered, the Gatekeeper may provide a screening service that assures the address provided is actually that of the calling party. The Gatekeeper may also provide the calling party address when no address is provided by the calling party or when the calling party provides an address other than an address with which the calling party registered.

When a call originates in the switched circuit network and enters the packet network through a Gateway, the Gateway shall pass to the packet network the calling party number information provided from the switched circuit network.

7.8.1.2 Calling party address restriction

Calling party address restriction is a feature which allows the calling endpoint or the calling endpoint's Gatekeeper to restrict presentation of the calling party alias address to the called party. This feature may reside in the endpoint or in the Gatekeeper for Gatekeeper routed calls.

In some cases where calling party address restriction has been indicated, there may exist certain situations where the restriction is overridden (for example, if the called party provides some emergency service).

7.8.1.3 Connected party address presentation

Connected party address presentation is a feature which provides the alias address of the connected or answering party to the calling party. The connected party address may be provided by the connected endpoint or by the Gatekeeper for Gatekeeper routed calls. When the call is routed

through the Gatekeeper with which the connected endpoint is registered, the Gatekeeper may provide a screening service that assures the address provided is actually that of the connected party. The Gatekeeper may also provide the connected party address when no address is provided by the connected party or when the connected party provides an address other than an address with which the connected party registered.

A Gateway shall pass connected party information received from the switched circuit network to the packet network.

7.8.1.4 Connected party address restriction

Connected party address restriction is a feature which allows the connected endpoint or the connected endpoint's Gatekeeper to restrict presentation of the connected party alias address to the calling party. This feature may reside in the endpoint or in the Gatekeeper for Gatekeeper routed calls.

In some cases where connected party address restriction has been indicated, there may exist certain situations where the restriction is overridden (for example, if the calling party provides some emergency service).

7.8.1.5 Called (alerting) party address presentation

Alerting party address presentation is a feature which provides the alias address of the alerting party to the calling party. The alerting party address may be provided by the alerting endpoint or by the Gatekeeper for Gatekeeper routed calls. When the call is routed through the Gatekeeper with which the alerting endpoint is registered, the Gatekeeper may provide a screening service that assures the address provided is actually that of the alerting party. The Gatekeeper may also provide the alerting party address when no address is provided by the alerting party or when the alerting party provides an address other than an address with which the alerting party registered.

7.8.1.6 Called (alerting) party address restriction

Alerting party address restriction is a feature which allows the alerting endpoint or the alerting endpoint's Gatekeeper to restrict presentation of the alerting party alias address to the calling party. This feature may reside in the endpoint or in the Gatekeeper for Gatekeeper routed calls.

7.8.1.7 Busy party address presentation

Busy party address presentation is a feature which provides the alias address of the busy party to the calling party. The busy party address may be provided by the busy endpoint or by the Gatekeeper for Gatekeeper routed calls. When the call is routed through the Gatekeeper with which the busy endpoint is registered, the Gatekeeper may provide a screening service that assures the address provided is actually that of the busy party. The Gatekeeper may also provide the busy party address when no address is provided by the busy party or when the busy party provides an address other than an address with which the busy party registered.

7.8.1.8 Busy party address restriction

Busy party address restriction is a feature which allows the busy endpoint or the busy endpoint's Gatekeeper to restrict presentation of the busy party alias address to the calling party. This feature may reside in the endpoint or in the Gatekeeper for Gatekeeper routed calls.

7.8.2 Messages and information elements

This clause describes the various messages and information elements that allow H.323 devices to provide address presentation and restriction services.

7.8.2.1 Calling party address information

Calling party address information appears in the Setup message.

When address information represents a telephone number, the relevant information may appear in the Calling Party Number IE. This IE contains the caller's number, information about the number, and presentation and screening indicators found in octet 3a. This is the recommended mode of operation for the case where a PSTN Gateway sends a Setup message on the packet network.

Alternatively, calling party information may appear in the **sourceAddress**, **presentationIndicator**, and **screeningIndicator** fields of the Setup message. This mode of operation is required when the **sourceAddress** is not in any form of telephone number (i.e., **sourceAddress** is not a type of **dialledDigits** or **partyNumber**). In accordance with 7.2.2.6/H.225.0, it is also required when the address information is in the form of a telephone number belonging to a Private Numbering Plan.

The **presentationIndicator** field in the Setup message carries information identical to the presentation indicator found in the Calling Party Number IE. The meaning and use of the presentation indicator is defined in ITU-T Rec. Q.951.

The **screeningIndicator** field in the Setup message carries information identical to the screening indicator found in the Calling Party Number IE. The meaning and use of the screening indicator is defined in ITU-T Rec. Q.951.

7.8.2.2 Connected party address information

Connected party address information appears in the Connect message.

When address information represents a telephone number, the relevant information may appear in the Connected Number IE, including the presentation indicator and screening indicator. This is the recommended mode of operation for the case where a PSTN Gateway sends a Connect message on the packet network.

Alternatively, connected party information may appear in the **connectedAddress**, **presentationIndicator**, and **screeningIndicator** fields of the Connect message. This mode of operation is required when **connectedAddress** is not in any form of telephone number (i.e., **connectedAddress** is not type **dialledDigits** or **partyNumber**).

The **presentationIndicator** field in the Connect message carries information identical to the presentation indicator found in the Connected Number IE. The meaning and use of the presentation indicator is defined in ITU-T Rec. Q.951.

The **screeningIndicator** field in the Connect message carries information identical to the screening indicator found in the Connected Number IE. The meaning and use of the screening indicator is defined in ITU-T Rec. Q.951.

7.8.2.3 Called (alerting) party address information

Alerting party address information appears in the Alerting message.

Alerting party information may appear in the **alertingAddress**, **presentationIndicator**, and **screeningIndicator** fields of the Alerting message.

The **presentationIndicator** field in the Alerting message carries information identical to the presentation indicator found in the Connected Number IE. The meaning and use of the presentation indicator is defined in ITU-T Rec. Q.951.

The **screeningIndicator** field in the Alerting message carries information identical to the screening indicator found in the Connected Number IE. The meaning and use of the screening indicator is defined in ITU-T Rec. Q.951.

7.8.2.4 Busy party address information

Busy party address information appears in the Release Complete message.

Busy party information may appear in the **busyAddress**, **presentationIndicator**, and **screeningIndicator** fields of the Release Complete message.

The **presentationIndicator** field in the Release Complete message carries information identical to the presentation indicator found in the Connected Number IE. The meaning and use of the presentation indicator is defined in ITU-T Rec. Q.951.

The **screeningIndicator** field in the Release Complete message carries information identical to the screening indicator found in the Connected Number IE. The meaning and use of the screening indicator is defined in ITU-T Rec. Q.951.

7.8.3 Actions at the originating endpoint

This clause describes the procedural aspects required to provide caller identification services at the originating endpoint.

7.8.3.1 Gateway as originating endpoint

In the case of a Setup message received by a Gateway from the ISDN, the caller's number and presentation information reside in the Calling Party Number IE. The Gateway shall send a Setup message on the packet network with the Calling Party Number IE containing the same information as was found in the Setup message from the SCN with the following exception. If the Numbering Plan Identification field contains value Private Numbering Plan, the digits shall be omitted from the Calling Party Number IE in accordance with 7.2.2.6/H.225.0. In this exception case the Gateway shall place the received caller identification information in the **sourceAddress**, **presentationIndicator** and **screeningIndicator** fields in the Setup message. If the Gateway has the knowledge to send both a PNP Number and an E.164 Number, the Calling Party Number IE shall convey the E.164 Number (and not the "empty" PNP number).

A Gateway in receipt of a Connect message shall copy the Connected Number IE from the Connect message from the packet network to the Connect message to be sent to the ISDN. If the Connected Number IE is not present in the Connect message, the Gateway shall convert **connectedAddress**, **presentationIndicator**, and **screeningIndicator** into a Connected Number IE, if that **connectedAddress** represents some form of telephone number. If **connectedAddress** does not represent some form of telephone number or if the Connected Number IE is not present in the Connect message, the Gateway shall omit the Connected Number IE from the Connect message sent to the ISDN.

A Gateway in receipt of an Alerting message with alerting party information or a Release Complete message with busy party information shall convert the party information to the signalling format of the Gateway's circuit side if the signalling format supports this party information.

7.8.3.2 Terminal or MCU as originating endpoint

For calls originated on the packet network, the originating terminal or MCU may send a Setup message with either the Calling Party Number IE with presentation and screening indicators or with **sourceAddress**, **presentationIndicator**, and **screeningIndicator** fields. In either case, the screening indicator shall indicate "user provided not screened". As an example, if the caller wants to block identification to the called party, the presentation indicator would be set to "presentation restricted", but the caller's number would still appear in the Calling Party Number IE. In Gatekeeper routed cases, the calling party's Gatekeeper may add this information if it is missing or incorrect and the called party's Gatekeeper may remove the caller's identification information if appropriate. The calling party's Gatekeeper or the called party's Gatekeeper may also add or remove address information based on local policy.

A terminal or MCU in receipt of a Connect, Alerting, or Release Complete message should honour the presentation indicator when presenting address information to the user.

7.8.4 Actions at the terminating endpoint

This clause describes the procedural aspects required to provide caller identification services at the terminating endpoint.

7.8.4.1 Gateway as terminating endpoint

A PSTN Gateway in receipt of a Setup message from the packet network shall copy the information found in the Calling Party Number IE from the Setup message to the signalling format supported in the PSTN. For example, this information would be copied to the Calling Party Number IE of the Q.931 Setup message for ISDN. If the Calling Party Number IE is not present in the Setup message, or if the Numbering Plan Identification field contains the value Private Numbering Plan, the Gateway shall form the Calling Party Number IE using the **sourceAddress** (assuming it is one of the telephone number alias types), **presentationIndicator**, and **screeningIndicator** from the Setup message.

The Gateway shall send a Connect message on the packet network with the Connected Number IE containing the same information as was found in the signalling format supported in the telephone network. In the case of a Q.931 Connect message received by a Gateway from the ISDN, connected party information resides in the Connected Number IE.

7.8.4.2 Terminal or MCU as terminating endpoint

A terminal or MCU in receipt of the Setup message should honour the presentation indicator when presenting caller information to the user.

For calls answered on the packet network, the answering terminal or MCU may include in the Connect message either the Connected Number IE or **connectedAddress**, **presentationIndicator**, and **screeningIndicator** fields. In either case, the terminal or MCU shall set the **screeningIndicator** to indicate "user provided not screened". In Gatekeeper routed cases, the answering party's Gatekeeper may add this information if it is missing or incorrect and the calling party's Gatekeeper may remove the answering party's address information if appropriate.

A terminal or MCU may provide address information in the Alerting message, using the **alertingAddress**, **presentationIndicator**, and **screeningIndicator** found in the Alerting message. If the address is provided, the terminal or MCU shall set the **screeningIndicator** to indicate "user provided not screened". In Gatekeeper routed cases, the answering party's Gatekeeper may add this information if it is missing or incorrect and the calling party's Gatekeeper may remove the answering party's address information if appropriate. The answering party's Gatekeeper or the calling party's Gatekeeper may also add or remove address information based on local policy.

A busy terminal or MCU may provide address information in the Release Complete message, using the **busyAddress**, **presentationIndicator**, and **screeningIndicator** found in the Release Complete message. If the address is provided, the terminal or MCU shall set the **screeningIndicator** to indicate "user provided not screened". In Gatekeeper routed cases, the answering party's Gatekeeper may add this information if it is missing or incorrect and the calling party's Gatekeeper may remove the answering party's address information if appropriate.

7.8.5 Actions at a gatekeeper

In Gatekeeper routed scenarios, the Gatekeeper may provide identification information or may provide a screening service. Services that may be provided by a Gatekeeper depend on the type of endpoint served. This clause describes the procedural aspects required to provide caller identification services when the Gatekeeper routes the call signalling.

7.8.5.1 Gateway as originating endpoint

In Gatekeeper routed cases, a Gatekeeper should not modify the information found in the Setup message sent from a Gateway. This assumes that the telephone network has provided correct information.

7.8.5.2 Terminal or MCU as originating endpoint

In Gatekeeper routed cases, a Gatekeeper may provide calling party information when the calling party is not a Gateway. The Gatekeeper may provide a calling party address if the calling party did not provide one or if the Gatekeeper determines the address is not correct. If the Gatekeeper provides an address other than that sent in the Setup message, the Gatekeeper shall set the screening indicator to indicate "network provided". If the Gatekeeper verifies the address information sent in the Setup message, but does not modify the address information, the Gatekeeper shall set the screening indicator to indicate "user provided, verified, and passed". If the Gatekeeper determines that the address information sent in the Setup message is incorrect, but does not modify the address information, the Gatekeeper shall set the screening indicator to indicate "user provided, verified, and failed". The Gatekeeper may set the presentation indicator to provide service to the endpoint. The Gatekeeper may allow the endpoint to override the endpoint's service by specifying a different presentation (for example, restricting presentation for the current call when the endpoint's service is to allow presentation).

7.8.5.3 Gateway as terminating endpoint

In Gatekeeper routed cases, a Gatekeeper should not modify the information found in the Connect message sent from a Gateway. This assumes that the telephone network has provided correct information.

7.8.5.4 Terminal or MCU as terminating endpoint

In Gatekeeper routed cases, a Gatekeeper may provide connected, alerting, or busy party information when the connected, alerting, or busy party is not from a Gateway. The Gatekeeper may provide a connected party (or alerting party, or busy party) address if none was provided by the connected party (or alerting party, or busy party), or if the Gatekeeper determines the address is not correct. If the Gatekeeper provides an address other than that sent in the Connect, Alerting, or Release Complete message, the Gatekeeper shall set the screening indicator to indicate "network provided". If the Gatekeeper verifies the address information sent in the Connect, Alerting, or Release Complete message, but does not modify the address information, the Gatekeeper shall set the screening indicator to indicate "user provided, verified, and passed". If the Gatekeeper determines that the address information sent in the Connect, Alerting, or Release Complete message is incorrect, but does not modify the address information, the Gatekeeper shall set the screening indicator to indicate "user provided, verified, and failed". The Gatekeeper may set the presentation indicator to provide service to the endpoint. The Gatekeeper may allow the endpoint to override the endpoint's service by specifying a different presentation (for example, restricting presentation for the current call when the endpoint's service is to allow presentation).

7.9 Generic extensible framework

The generic extensibility framework allows new features to be readily added to the protocol without affecting the underlying H.225.0 core specification. The extensible framework consists of two parts:

- Carriage of opaque data within H.225.0 messages.
- Negotiation of supported features.

Support of the **generic extensibility framework** is optional.

7.9.1 Format of a GenericData structure

Opaque data may be carried in a sub-set of RAS messages and H.225.0 call signalling messages in the **genericData** field.

The **GenericData** structure primarily consists of an identifier and zero or more parameters, which allows flexible definition of opaque data and features. The **GenericData** structure consists of an **id** to identify the generic data and the **parameters** field to convey the actual parameters.

Each parameter also contains an identifying **id** and a **content** field. The **content** field supports a number of different data types, including **raw**, **text**, **unicode**, **bool**, **number8**, **number16**, **number32**, **id**, **compound**, and **nested**. This allows for flexible definition of generic data and eases implementation. However, it is expected that for generic data that contain a large number of parameters, the **raw** form of **content** shall be used, which will contain ASN.1 data.

7.9.2 Negotiation using the extensible framework – General

The extensible framework provides a common method for feature negotiation that operates over multiple domains and may be managed and configured by different operational entities. Hence, entities do not require *a priori* knowledge of other entities' feature sets to operate successfully.

The mechanism used to negotiate features in both RAS and call signalling uses the **FeatureDescriptor**, which is an alias of a **GenericData** structure as described above. This allows a feature to be identified and have parameters associated with it.

Intermediate signalling entities may – subject to security issues – add their needed, desired and supported features to messages that pass through them. Intermediate entities may remove desired and supported features specified in messages before passing them on. Intermediate entities shall not remove needed feature fields unless they intend to support the features that they are removing. If the intermediate entity does not wish to allow a needed feature, then it shall reject the transaction.

If an intermediate entity elects to support a requested feature signalled in a message, then it should remove the feature request from the message before passing it on. By some means, the intermediate entity should signal back to the requesting entity that the feature is supported. This may be achieved by modifying the response from the remote entity, or by generating its own message.

7.9.3 Negotiation using the extensible framework – RAS

RAS feature negotiation applies to the discovery, registration, and call setup phase. In particular, it applies to the exchange of discovery messages (GRQ, GCF, GRJ), registration messages (RRQ, RCF, RRJ), admission request messages (ARQ, ACF, ARJ), location request messages (LRQ, LCF, LRJ), service control messages (SCI/SCR), and the NonStandardMessage.

In RAS negotiation, entities may specify the set of features that they need for a transaction to be successful, the set of features they desire, and the set of features they support.

7.9.3.1 Processing by the requesting entity

A requesting entity (usually an endpoint) uses the elements in the **FeatureSet** structure to specify the various types of features it requires. It specifies the set of features that it needs using the **neededFeatures** field, the set of features that it desires using the **desiredFeatures** field, and the set of features that it supports in the **supportedFeatures** field. All three of these fields are in the **FeatureSet** structure.

In response to its request, a requesting entity should receive either a confirm or reject message.

If the request is rejected, the responding entity may have included a set of **neededFeatures** that the requesting entity must support in order for the request to be successful. If this is the case and the requesting entity supports the needed features, the requesting entity may reissue a request specifying support for the features needed by the responding entity.

If the request is accepted, special procedures need to be applied to ensure that the negotiation operates in a backwards-compatible manner. This is done by the requesting entity checking that the features that it specified as needed are listed as **supportedFeatures** in the response. If a requesting entity does not observe the features it needs in the response message's **supportedFeatures** field, then it shall assume that the responding entity does not support the features that it needs. If the requesting entity determines that it cannot continue under these circumstances, then it shall undo the operation it was trying to perform (i.e., send a DRQ if it originally sent an ARQ and so forth), so that the state in the responding entity is rolled back.

7.9.3.2 Processing by the responding entity

The responding entity (typically a Gatekeeper) looks at the features specified in the **neededFeatures** field of the request to determine if it can accept the request. It also looks in the **neededFeatures**, **desiredFeatures** and **supportedFeatures** fields to determine whether the features needed by it are supported by the requesting entity.

If the responding entity is a Gatekeeper that sends an LRQ in response to receiving an ARQ, the Gatekeeper shall copy any features that are not provided by the Gatekeeper into the LRQ. In trying to determine whether the necessary set of features are supported, the Gatekeeper shall examine the supported features of the endpoint to which the ARQ may resolve, either locally or in response to an LCF, and the features supported by the Gatekeeper.

If the responding entity determines that the necessary sets of features are supported by both entities, then the responding entity may acknowledge the request. The responding entity lists the set of features that it chooses to support in the **supportedFeatures** field of it reply. If the request is accepted, then all of the **neededFeatures** from the request must be included in the **supportedFeatures** field of the reply. The responding entity may also include **desiredFeatures**.

If the responding entity needs additional features to be supported by the requesting entity, it shall reject the request. If it wishes to declare which features must be supported for the request to be successful, this should be specified using the **neededFeatures** field of the reject message. The responding entity may also include any **desiredFeatures** and **supportedFeatures** in the reject message.

7.9.4 Negotiation using the extensible framework – Call signalling

The following describes the negotiation process for the call signalling channel.

7.9.4.1 Processing by the initiating endpoint

An initiating endpoint may specify the features it needs for a call, the features it desires, and the features it supports. It specifies the set of features that it needs using the **neededFeatures** field in Setup. It also specifies the set of features that it desires using the **desiredFeatures** field and the set of features that it supports using the **supportedFeatures** field.

If the call is rejected, one or more responding entities may have included a set of **neededFeatures** that the initiating endpoint must support in order for the call to be successful. If this is the case, and the initiating endpoint supports the needed features, the initiating endpoint may reinitiate a call specifying support for the features needed by the various entities along the call signalling path.

If the call is accepted, the initiating endpoint shall check that the features that it specified as needed are listed as **supportedFeatures** in the Alerting or Connect message. If an initiating endpoint does not observe the features it needs in the message's **supportedFeatures** field, then it shall assume that the entities along the call signalling path do not support the features that it needs. If the initiating entity determines that it cannot continue under these circumstances, then it shall clear the call using Release Complete.

When an initiating endpoint receives an empty capability set as a result of third-party pause and re-routing, it shall remove any knowledge it has of any remote entities capabilities. When the endpoint receives a non-empty capability set, it shall send its feature set using the **featureSet** field in a Facility message with the **reason** field set to **featureSetUpdate**. In this message, the **replacementFeatureSet** field shall be set to TRUE. When the feature set from the remote endpoint is received in a Facility message, the contents may be interpreted in the same way as above.

7.9.4.2 Processing by intermediate entities

Intermediate entities along the call signalling path, such as Gatekeepers and Border Elements, may also interact with the negotiation process.

Intermediate entities along the signalling path may – subject to security issues – add their needed, desired and supported features to the call signalling messages that pass through them. Intermediate entities may remove desired and supported features specified in messages (including Setup, Alerting and Connect) before passing them on. Intermediate entities shall not remove needed feature fields from a Setup message or a Facility message unless they intend to support the features that they remove. If the intermediate entity does not wish to allow a needed feature, then it shall reject or terminate the call.

If an intermediate entity elects to support a requested feature signalled in a Setup message, then it should remove the feature request from the Setup message before passing it on. The intermediate entity should signal supported features it supports in the Alerting (if sent) or Connect messages along with the destinations supported feature set.

When an intermediate entity receives a **featureSet** parameter in a Facility message with the **replacementFeatureSet** field set to TRUE, it shall modify the features indicated according to its requirements in a similar fashion to how it modifies the features signalled in the Setup, Alerting or Connect messages. It should then pass the message on.

7.9.4.3 Processing by the called endpoint

The called endpoint looks at the features specified in the **neededFeatures** field of the Setup message to determine if it can accept the call. It also looks in the **neededFeatures**, **desiredFeatures** and **supportedFeatures** fields to determine whether the features needed by it are supported by the various entities along the call signalling path.

If the called endpoint determines that the necessary sets of features are supported by the appropriate entities, then the called endpoint may accept the call. The called endpoint lists the set of features that it chooses to support in the **supportedFeatures** field of the Alerting (if sent) and Connect messages. If the call is accepted, then all of the **neededFeatures** from the Setup message must be declared in the **supportedFeatures** field of the Alerting (if sent) or Connect call signalling messages. The called endpoint may also include **desiredFeatures** within the message.

If the called endpoint needs additional features to be supported by the various entities along the call signalling path, it shall reject the call by sending a Release Complete. If it wishes to declare which features must be supported for the call to be successful, this should be specified using the **neededFeatures** field in the Release Complete message. The called endpoint may also include any **desiredFeatures** and **supportedFeatures** in the Release Complete message.

When a called endpoint receives an empty capability set as a result of third-party pause and re-routing, it shall act in the same way as if it had initiated the call. That is, it shall remove any knowledge it has of any remote entities capabilities. When the endpoint later receives a non-empty capability set, it shall send its feature set using the **featureSet** field in a Facility message with the **reason** field set to **featureSetUpdate**. In this message, the **replacementFeatureSet** field shall be set to TRUE. When the feature set from the remote endpoint is received in a Facility message, the contents may be interpreted in the same way as above.

8 Call signalling procedures

The provision of the communication is made in the following steps:

– Phase A: Call setup (see 8.1).
– Phase B: Initial communication and capability exchange (see 8.2).
– Phase C: Establishment of audiovisual communication (see 8.3).
– Phase D: Call services (see 8.4).
– Phase E: Call termination (see 8.5).

8.1 Phase A – Call setup

Call setup takes place using the call control messages defined in ITU-T Rec. H.225.0 according to the call control procedures defined below. Requests for bandwidth reservation should take place at the earliest possible phase.

If both the alias address and the Transport Address are specified, preference shall be given to the alias address.

There is no explicit synchronization or locking between two endpoints during the call setup procedure. This implies that endpoint A can send a Setup message to endpoint B at exactly the same time that endpoint B sends a Setup message to endpoint A. It is up to the application to determine if only one call is desired and to take the appropriate action. This action may be for an endpoint to indicate that it is busy whenever it has an outstanding Setup message. If an endpoint can support more than one simultaneous call, it should indicate that it is busy whenever it receives a Setup message from the same endpoint to which it has an outstanding Setup message.

An endpoint shall be capable of sending the Alerting message. Alerting has the meaning that the called party (user) has been alerted of an incoming call. Alerting shall only be originated by the ultimate called endpoint and then only when it has alerted the user. In the case of interworking through a Gateway, the Gateway shall send Alerting when it receives a ring indication from the SCN. If an endpoint can respond to a Setup message with a Connect, Call Proceeding, or Release Complete within 4 seconds, it is not required to send the Alerting message. An endpoint sending the Setup message can expect to receive either an Alerting, Connect, Call Proceeding, or Release Complete message within 4 seconds after successful transmission.

The Connect message should be sent only if it is certain that the H.245 capability exchange will conclude successfully and a minimum level of communications can take place. This is to maintain the consistency of the meaning of the Connect message between packet based networks and circuit switched networks.

8.1.1 Basic call setup – neither endpoint registered

In the scenario shown in Figure 29 neither endpoint is registered to a Gatekeeper. The two endpoints communicate directly. Endpoint 1 (calling endpoint) sends the Setup (1) message to the well-known Call Signalling Channel TSAP Identifier of Endpoint 2. Endpoint 2 responds with the Connect (4) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

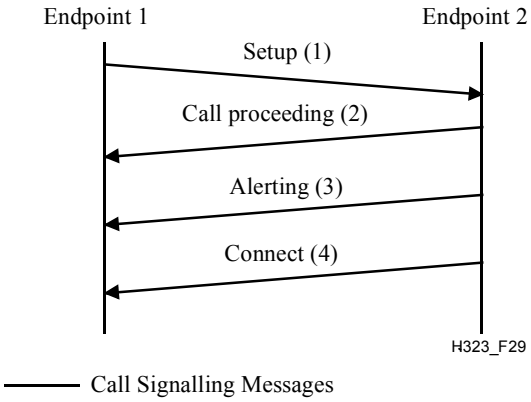


Figure 29/H.323 – Basic call setup, no Gatekeepers

8.1.2 Both endpoints registered to the same gatekeeper

In the scenario shown in Figure 30, both endpoints are registered to the same Gatekeeper, and the Gatekeeper has chosen direct call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with that Gatekeeper. The Gatekeeper shall return the Call Signalling Channel Transport Address of Endpoint 2 (called endpoint) in the ACF. Endpoint 1 then sends the Setup (3) message to Endpoint 2 using that Transport Address. If Endpoint 2 wishes to accept the call, it initiates an ARQ (5)/ACF (6) exchange with the Gatekeeper. It is possible that an ARJ (6) is received by Endpoint 2, in which case it sends Release Complete to Endpoint 1. Endpoint 2 responds with the Connect (8) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

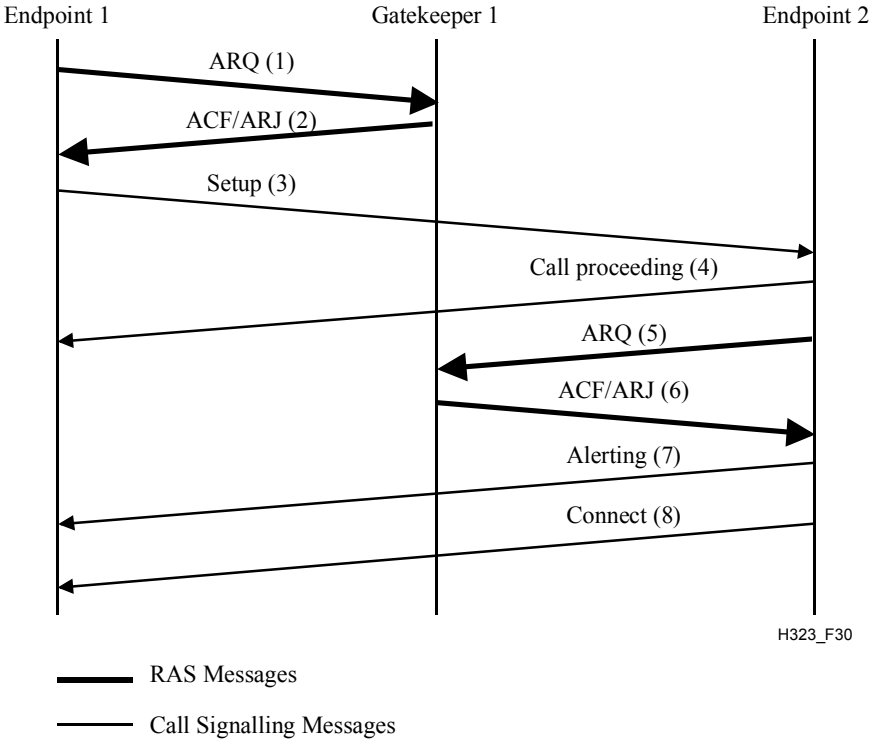


Figure 30/H.323 – Both endpoints registered, same gatekeeper – Direct call signalling

In the scenario shown in Figure 31, both endpoints are registered to the same Gatekeeper, and the Gatekeeper has chosen to route the call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with that Gatekeeper. The Gatekeeper shall return a Call Signalling Channel Transport Address of itself in the ACF. Endpoint 1 then sends the Setup (3) message using that Transport Address. The Gatekeeper then sends the Setup (4) message to Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates an ARQ (6)/ACF (7) exchange with the Gatekeeper. It is possible that an ARJ (7) is received by Endpoint 2, in which case it sends Release Complete to the Gatekeeper. Endpoint 2 responds with the Connect (9) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling. The Gatekeeper sends the Connect (10) message to Endpoint 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper H.245 Control Channel Transport Address, based on whether the Gatekeeper chooses to route the H.245 Control Channel or not.

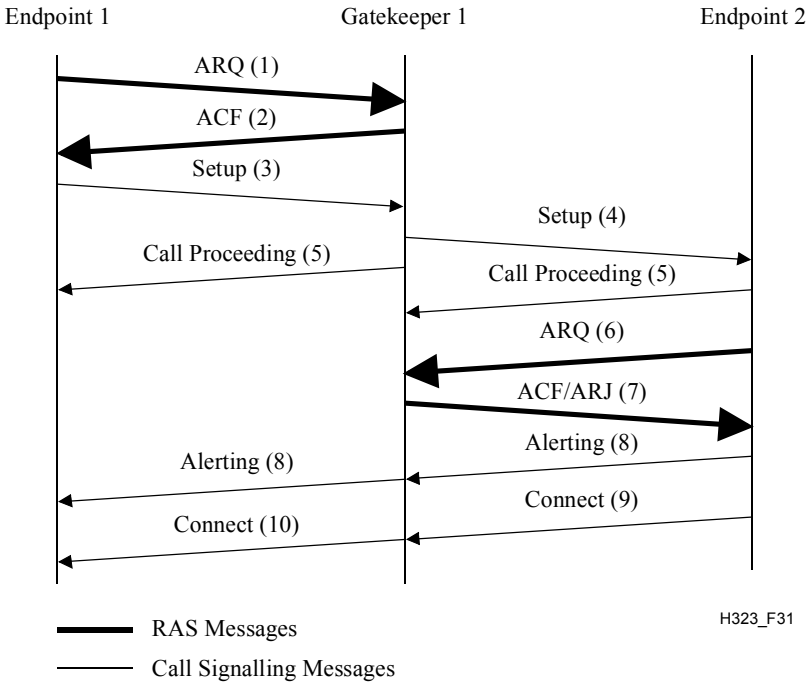


Figure 31/H.323 – Both endpoints registered, same Gatekeeper – Gatekeeper routed call signalling

8.1.3 Only calling endpoint has gatekeeper

In the scenario shown in Figure 32, Endpoint 1 (calling endpoint) is registered with a Gatekeeper, Endpoint 2 (called endpoint) is not registered with a Gatekeeper, and the Gatekeeper has chosen direct call signalling. Endpoint 1 initiates the ARQ (1)/ACF (2) exchange with the Gatekeeper. Endpoint 1 then sends the Setup (3) message to Endpoint 2 using the well-known Call Signalling Channel Transport Address. If Endpoint 2 wishes to accept the call, it responds with the Connect (6) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

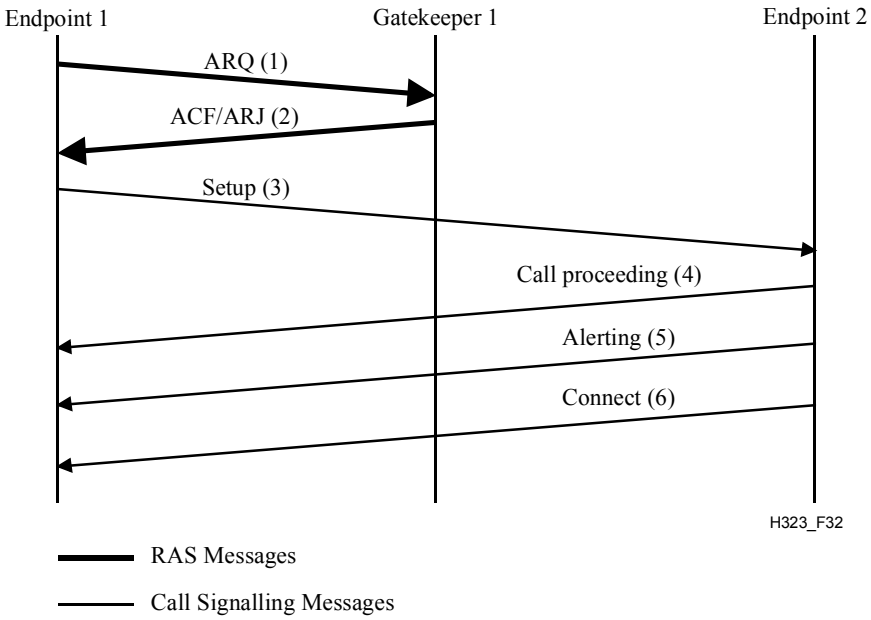


Figure 32/H.323 – Only calling endpoint registered – Direct call signalling

In the scenario shown in Figure 33, Endpoint 1 (calling endpoint) is registered with a Gatekeeper, Endpoint 2 (called endpoint) is not registered with a Gatekeeper, and the Gatekeeper has chosen to route the call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with that Gatekeeper. The Gatekeeper shall return a Call Signalling Channel Transport Address of itself in the ACF (2). Endpoint 1 then sends the Setup (3) message using that Transport Address. The Gatekeeper then sends the Setup (4) message to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it responds with the Connect (7) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling. The Gatekeeper sends the Connect (8) message to Endpoint 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper H.245 Control Channel Transport Address, based on whether the Gatekeeper chooses to route the H.245 Control Channel or not.

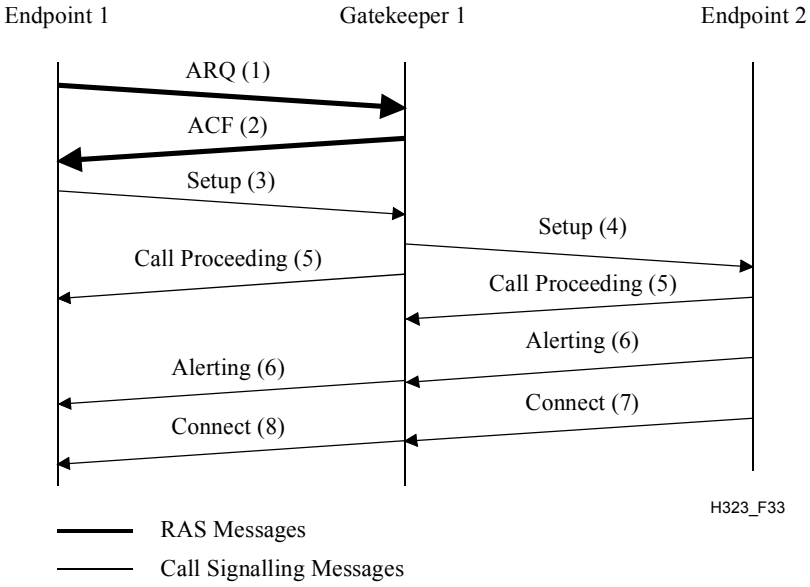


Figure 33/H.323 – Only calling endpoint registered – Gatekeeper routed call signalling

8.1.4 Only called endpoint has gatekeeper

In the scenario shown in Figure 34, Endpoint 1 (calling endpoint) is not registered with a Gatekeeper, Endpoint 2 (called endpoint) is registered with a Gatekeeper, and the Gatekeeper has chosen direct call signalling. Endpoint 1 sends the Setup (1) message to Endpoint 2 using the well-known Call Signalling Channel Transport Address. If Endpoint 2 wishes to accept the call, it initiates an ARQ (3)/ACF (4) exchange with the Gatekeeper. It is possible that an ARJ (4) is received by Endpoint 2, in which case it sends Release Complete to Endpoint 1. Endpoint 2 responds with the Connect (6) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

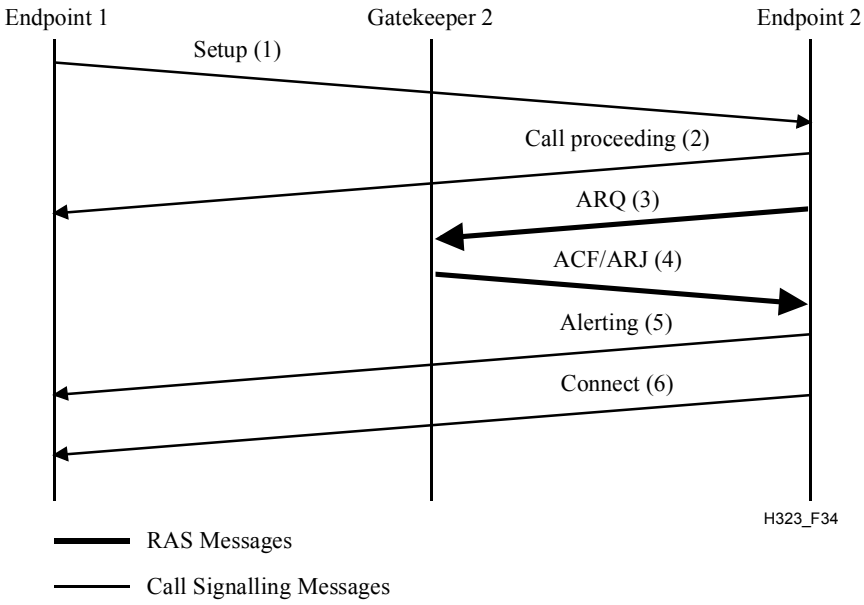


Figure 34/H.323 – Only called endpoint registered – Direct call signalling

In the scenario shown in Figure 35, Endpoint 1 (calling endpoint) is not registered with a Gatekeeper, Endpoint 2 (called endpoint) is registered with a Gatekeeper, and the Gatekeeper has chosen to route the call signalling. Endpoint 1 (calling endpoint) sends a Setup (1) message to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates the ARQ (3)/ACF (4) exchange with that Gatekeeper. If acceptable, the Gatekeeper shall return a Call Signalling Channel Transport Address of itself in the ARJ (4) with a cause code of **routeCallToGatekeeper**. Endpoint 2 replies to Endpoint 1 with a Facility (5) message containing the Call Signalling Transport Address of its Gatekeeper. Endpoint 1 then sends the Release Complete (6) message to Endpoint 2. Endpoint 1 sends a Setup (7) message to the Gatekeeper's Call Signalling Channel Transport Address. The Gatekeeper sends the Setup (8) message to Endpoint 2. Endpoint 2 initiates the ARQ (9)/ACF (10) exchange with that Gatekeeper. Endpoint 2 then responds with the Connect (12) message which contains its H.245 Control Channel Transport Address for use in H.245 signalling. The Gatekeeper sends the Connect (13) message to Endpoint 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper H.245 Control Channel Transport Address, based on whether the Gatekeeper chooses to route the H.245 Control Channel or not.

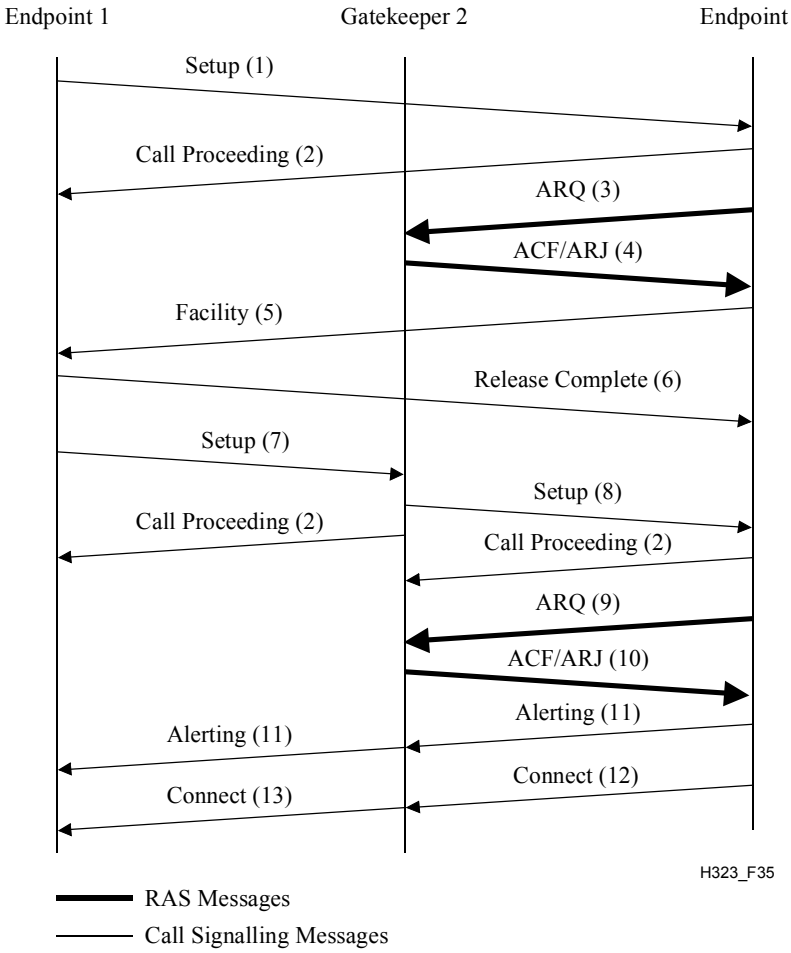


Figure 35/H.323 – Only called endpoint registered – Gatekeeper routed call signalling

8.1.5 Both endpoints registered to different gatekeepers

In the scenario shown in Figure 36, both endpoints are registered to different Gatekeepers, and both Gatekeepers choose direct call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with Gatekeeper 1. Gatekeeper 1 may return the Call Signalling Channel Transport Address of Endpoint 2 (called endpoint) in the ACF if Gatekeeper 1 has a method of communicating with Gatekeeper 2. Endpoint 1 then sends the Setup (3) message to either the Transport Address returned by the Gatekeeper (if available) or to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates an ARQ (5)/ACF (6) exchange with Gatekeeper 2. It is possible that an ARJ (6) is received by Endpoint 2, in which case it sends Release Complete to Endpoint 1. Endpoint 2 responds with the Connect (8) message which contains an H.245 Control Channel Transport Address for use in H.245 signalling.

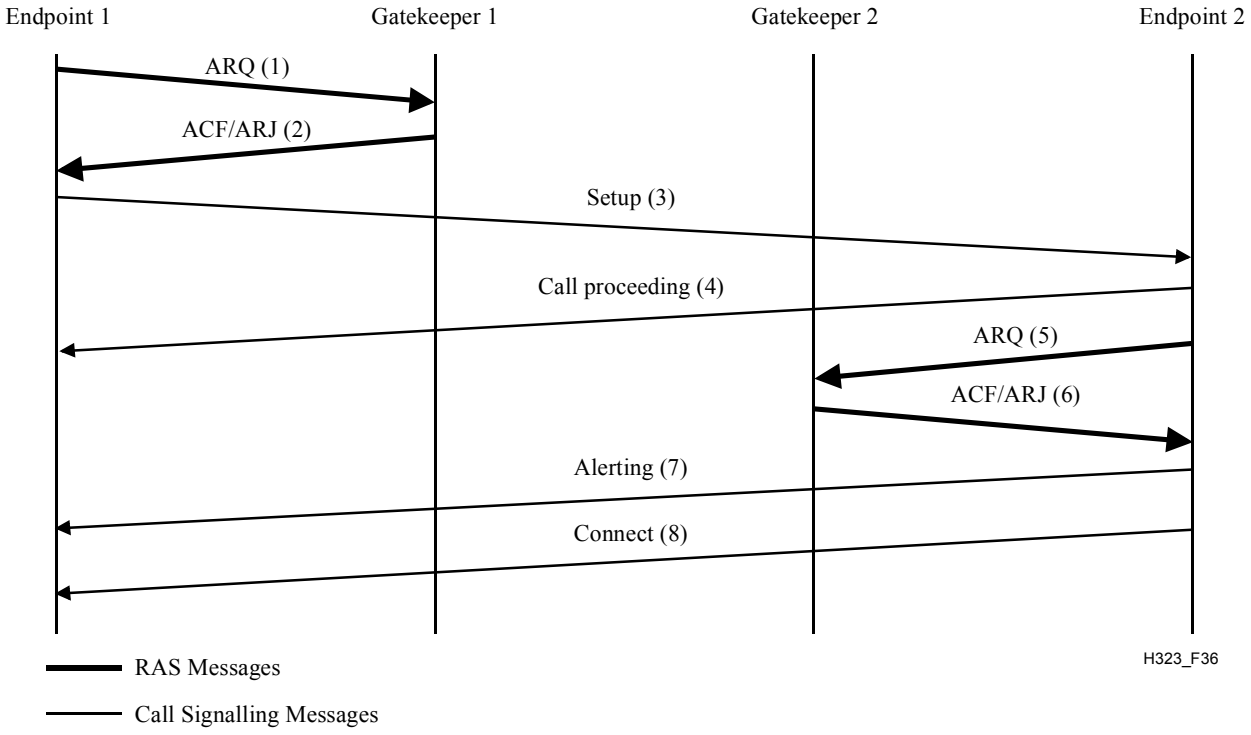


Figure 36/H.323 – Both endpoints registered – Both gatekeepers direct call signalling

In the scenario shown in Figure 37, both endpoints are registered to different Gatekeepers, the calling endpoint's Gatekeeper chooses direct call signalling, and the called endpoint's Gatekeeper chooses to route the call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with Gatekeeper 1. Gatekeeper 1 may return the Call Signalling Channel Transport Address of Endpoint 2 (called endpoint) in the ACF (2) if Gatekeeper 1 has a method of communicating with Gatekeeper 2. Endpoint 1 then sends the Setup (3) message to either the Transport Address returned by the Gatekeeper (if available) or to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates the ARQ (5)/ACF (6) exchange with Gatekeeper 2. If acceptable, Gatekeeper 2 shall return a Call Signalling Channel Transport Address of itself in the ARJ (6) with a cause code of **routeCallToGatekeeper**. Endpoint 2 replies to Endpoint 1 with a Facility (7) message containing the Call Signalling Transport Address of Gatekeeper 2. Endpoint 1 then sends the Release Complete (8) message to Endpoint 2. Endpoint 1 shall send a DRQ (9) to Gatekeeper 1 which responds with DCF (10). Endpoint 1 then initiates a new ARQ (11)/ACF (12) exchange with Gatekeeper 1. Endpoint 1 sends a Setup (13) message to the Gatekeeper's Call Signalling Channel Transport Address. Gatekeeper 2 sends the Setup (14) message to Endpoint 2. Endpoint 2 initiates the ARQ (15)/ACF (16) exchange with Gatekeeper 2. Endpoint 2 then responds with the Connect (18) message which contains its H.245 Control Channel Transport Address for use in H.245 signalling. Gatekeeper 2 sends the Connect (19) message to Endpoint 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper 2 H.245 Control Channel Transport Address, based on whether the Gatekeeper chooses to route the H.245 Control Channel or not.

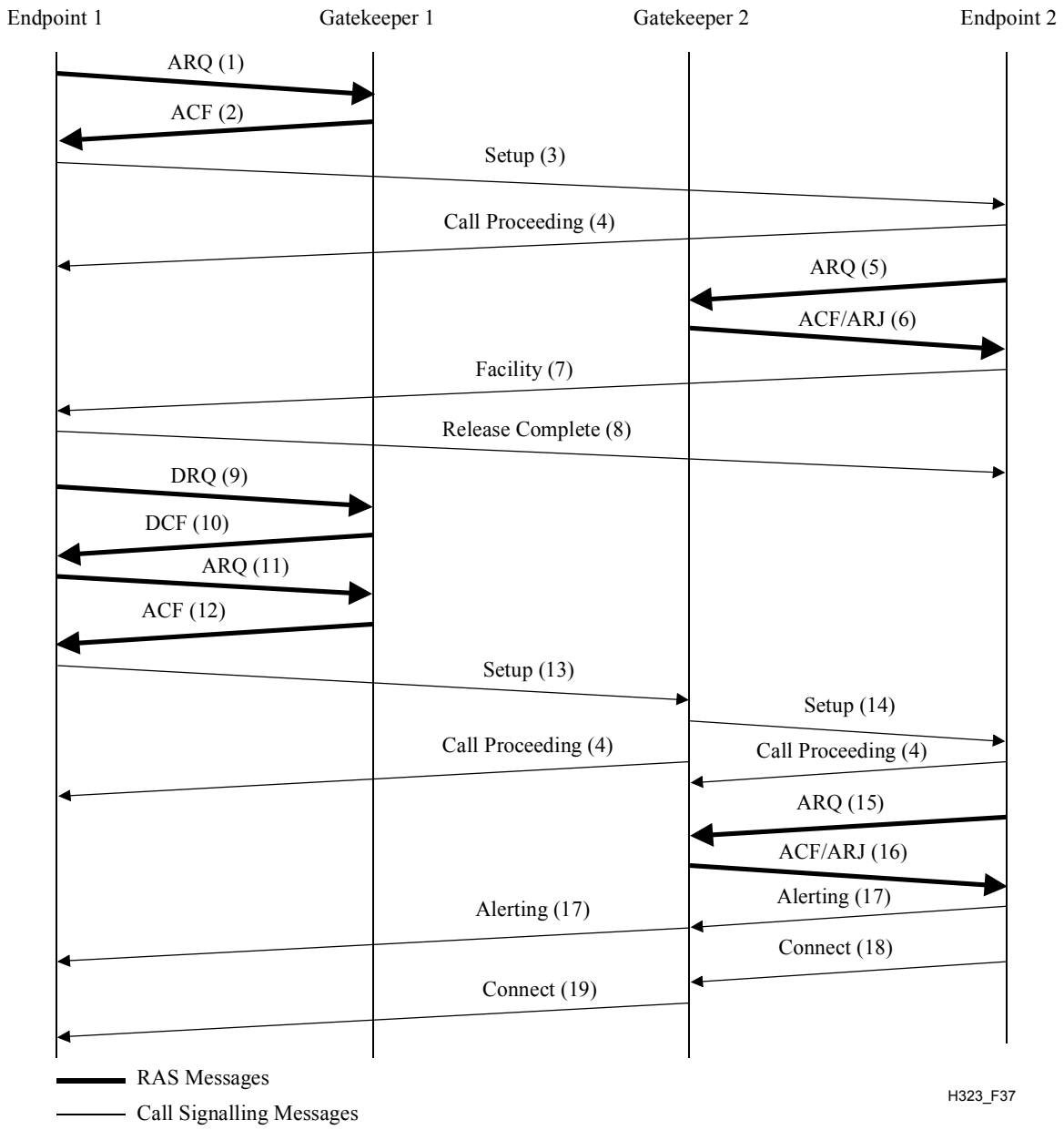


Figure 37/H.323 – Both endpoints registered – Direct/routed call signalling

In the scenario shown in Figure 38, both endpoints are registered to different Gatekeepers, the calling endpoint's Gatekeeper chooses to route the call signalling, and the called endpoint's Gatekeeper chooses direct call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with Gatekeeper 1. Gatekeeper 1 shall return a Call Signalling Channel Transport Address of itself in the ACF (2). Endpoint 1 then sends the Setup (3) message using that Transport Address. Gatekeeper 1 then sends the Setup (4) message containing its Call Signalling Channel Transport Address to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates the ARQ (6)/ACF (7) exchange with Gatekeeper 2. It is possible that an ARJ (7) is received by Endpoint 2, in which case it sends Release Complete to Endpoint 1. Endpoint 2 responds to Gatekeeper 1 with the Connect (9) message which contains its H.245 Control Channel Transport Address for use in H.245 signalling. Gatekeeper 1 sends the Connect (10) message to Endpoint 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper 1 H.245 Control Channel Transport Address, based on whether the Gatekeeper chooses to route the H.245 Control Channel or not.

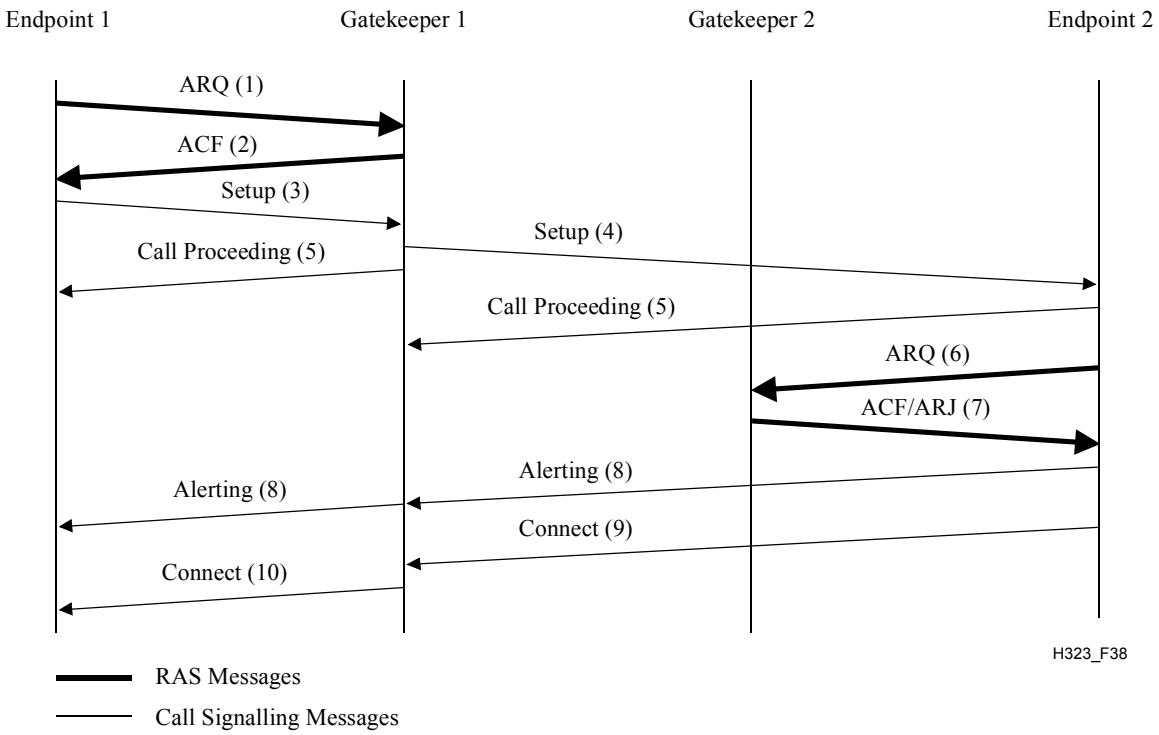


Figure 38/H.323 – Both endpoints registered – Routed/direct call signalling

In the scenario shown in Figure 39, both endpoints are registered to different Gatekeepers, and both Gatekeepers choose to route the call signalling. Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange with Gatekeeper 1. Gatekeeper 1 shall return a Call Signalling Channel Transport Address of itself in the ACF (2). Endpoint 1 then sends the Setup (3) message using that Transport Address. Gatekeeper 1 then sends the Setup (4) message to the well-known Call Signalling Channel Transport Address of Endpoint 2. If Endpoint 2 wishes to accept the call, it initiates the ARQ (6)/ACF (7) exchange with Gatekeeper 2. If acceptable, Gatekeeper 2 shall return a Call Signalling Channel Transport Address of itself in the ARJ (7) with a cause code of **routeCallToGatekeeper**. Endpoint 2 replies to Gatekeeper 1 with a Facility (8) message containing the Call Signalling Transport Address of Gatekeeper 2. Gatekeeper 1 then sends the Release Complete (9) message to Endpoint 2. Gatekeeper 1 sends a Setup (10) message to Gatekeeper 2's Call Signalling Channel Transport Address. Gatekeeper 2 sends the Setup (11) message to Endpoint 2. Endpoint 2 initiates the ARQ (12)/ACF (13) exchange with Gatekeeper 2. Endpoint 2 then responds to Gatekeeper 2 with the Connect (15) message which contains its H.245 Control Channel Transport Address for use in H.245 signalling. Gatekeeper 2 sends the Connect (16) message to Gatekeeper 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper 2 H.245 Control Channel Transport Address, based on whether the Gatekeeper 2 chooses to route the H.245 Control Channel or not. Gatekeeper 1 sends the Connect (17) message to Endpoint 1 which may contain the H.245 Control Channel Transport Address sent by Gatekeeper 2 or a Gatekeeper 1 H.245 Control Channel Transport Address, based on whether the Gatekeeper 1 chooses to route the H.245 Control Channel or not.

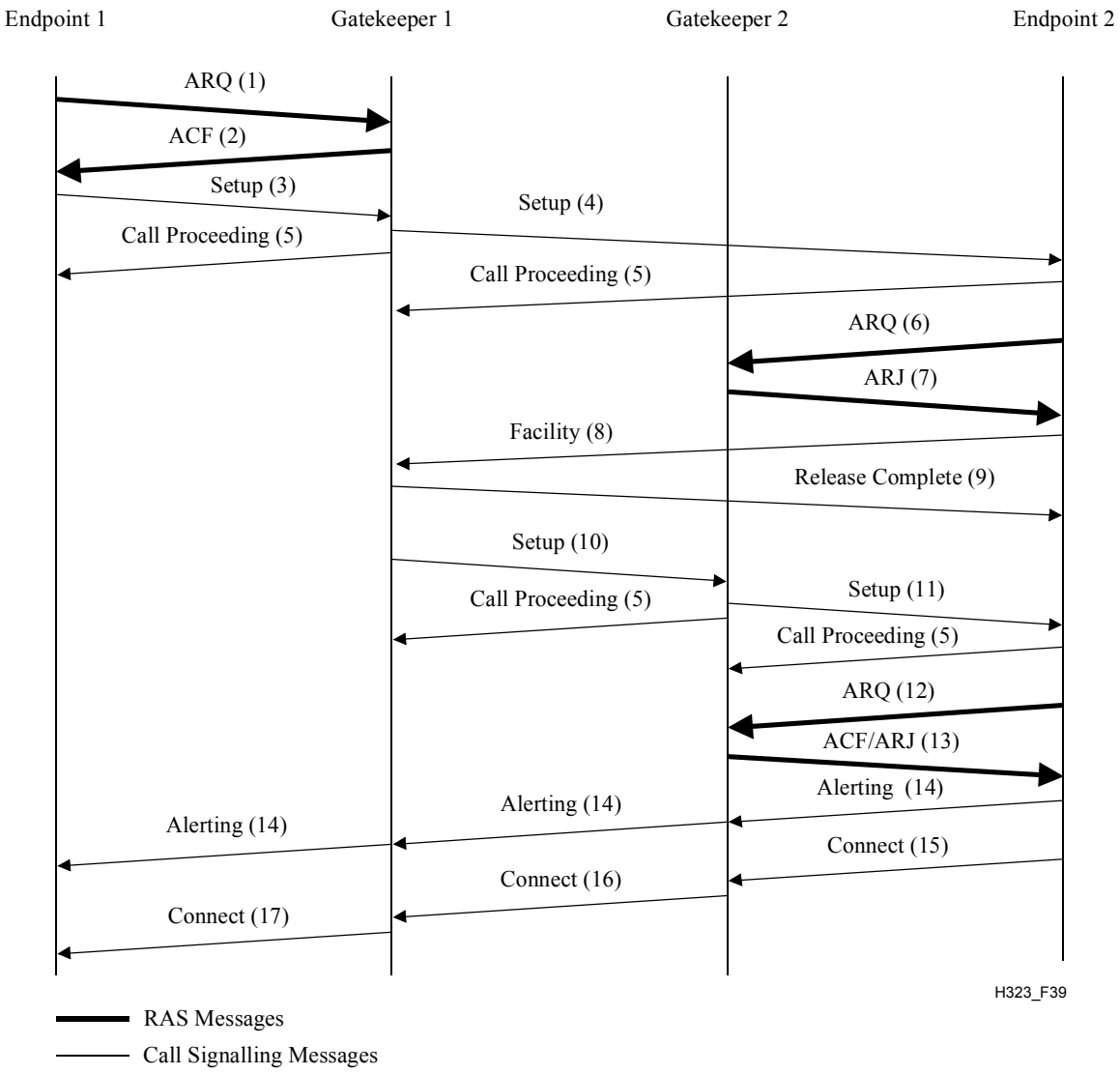


Figure 39/H.323 – Both endpoints registered – Both Gatekeepers routing call signalling

8.1.6 Optional called endpoint signalling

The procedures defined in 8.1.4 and 8.1.5 show that when a called endpoint is registered to a Gatekeeper, a Setup message is initially sent to the called endpoint from the calling endpoint or the calling endpoint's Gatekeeper. If the called endpoint's Gatekeeper wishes to use the Gatekeeper routed call model, it returns its own Call Signalling Channel Transport Address in the ARJ. The called endpoint then uses the Facility message to redirect the call to the called endpoint's Gatekeeper's Call Signalling Transport Address. These procedures assume that the calling endpoint or calling endpoint's Gatekeeper only knows the called endpoints Call Signalling Channel Transport Address. This address may have been received in an LCF sent in response to an LRQ requesting the address of the called endpoint or it may be known through out-of-band methods.

If the called endpoint's Gatekeeper desires a Gatekeeper routed call model, it may return its own Call Signalling Transport Address in the LCF. This will allow the calling endpoint or calling endpoints Gatekeeper to send the Setup message directly to the called endpoints Gatekeeper, thus eliminating the redirection process.

An example of this scenario is shown in Figure 40. In this example, both endpoints are registered to different Gatekeepers, and both Gatekeepers choose to route the call signalling (similar to the case in Figure 39). Endpoint 1 (calling endpoint) sends an ARQ (1) to Gatekeeper 1. Gatekeeper 1 multicasts an LRQ (2) to locate called Endpoint 2. Gatekeeper 2 returns an LCF (3) with the Call

Signalling Channel Transport Address of itself. Thus, Gatekeeper 1 will subsequently send a Setup (6) message to Gatekeeper 2's Call Signalling Channel Transport Address and Gatekeeper 2 will send a Setup (8) message to Endpoint 2. Endpoint 2 initiates the ARQ (9)/ACF (10) exchange with Gatekeeper 2. Endpoint 2 then responds to Gatekeeper 2 with the Connect (12) message which contains its H.245 Control Channel Transport Address for use in H.245 signalling. Gatekeeper 2 sends the Connect (13) message to Gatekeeper 1 which may contain the Endpoint 2 H.245 Control Channel Transport Address or a Gatekeeper 2 H.245 Control Channel Transport Address, based on whether the Gatekeeper 2 chooses to route the H.245 Control Channel or not. Gatekeeper 1 sends the Connect (14) message to Endpoint 1 which may contain the H.245 Control Channel Transport Address sent by Gatekeeper 2 or a Gatekeeper 1 H.245 Control Channel Transport Address, based on whether the Gatekeeper 1 chooses to route the H.245 Control Channel or not.

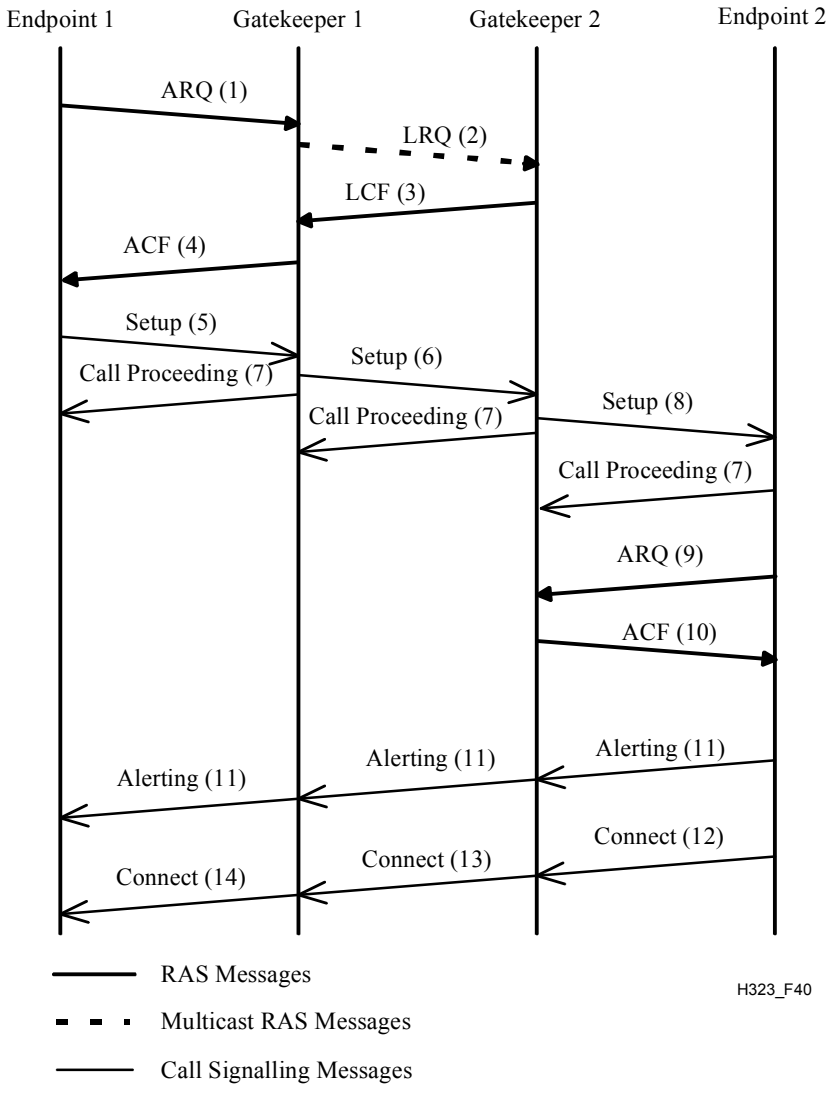


Figure 40/H.323 – Optional called endpoint signalling

8.1.7 Fast connect procedure

H.323 endpoints may establish media channels in a call using either the procedures defined in ITU-T Rec. H.245 or the "Fast Connect" procedure described in this clause. The Fast Connect procedure allows the endpoints to establish a basic point-to-point call with as few as one round-trip message exchange, enabling immediate media stream delivery upon call connection.

The calling endpoint initiates the Fast Connect procedure by sending a Setup message containing the **fastStart** element to the called endpoint. The **fastStart** element consists of a sequence of **OpenLogicalChannel** structures describing media channels which the calling endpoint proposes to send and receive, including all of the parameters necessary to immediately open and begin transferring media on the channels. Details of the content and usage of the **fastStart** element are discussed below.

The called endpoint may refuse to use the Fast Connect procedure, either because it does not implement it or because it intends to invoke features that require use of the procedures defined in ITU-T Rec. H.245. Refusal of the Fast Connect procedure is accomplished by either not returning the **fastStart** element or including the **fastConnectRefused** element in any H.225.0 call signalling message up to and including the Connect message. Note that an endpoint may omit **fastStart** elements in a message prior to Connect, but then later return **fastStart** element in the Connect message thereby accepting the Fast Connect procedure. Refusing the Fast Connect procedure (or not initiating it) requires that H.245 procedures be used for capabilities exchange and opening of media channels.

When the called endpoint desires to proceed with the Fast Connect procedure, it sends a H.225.0 call signalling message (Call Proceeding, Progress, Alerting, or Connect) containing a **fastStart** element selecting from amongst the **OpenLogicalChannel** proposals offered by the calling endpoint. The calling endpoint shall process each of these messages until it determines that Fast Connect is accepted or refused. Although the calling endpoint may receive the **fastStart** element in a Facility message sent by a Gatekeeper, the called endpoint shall not use the Facility message to send **fastStart**. Channels thus accepted are considered opened as though the usual H.245 **openLogicalChannel** and **openLogicalChannelAck** procedure had been followed. The called endpoint shall not include a **fastStart** element in any H.225.0 call signalling message sent after the Connect message and shall not include **fastStart** in any H.225.0 call signalling message unless the Setup message contained a **fastStart** element.

The called endpoint may begin transmitting media (according to the channels opened) immediately after sending a H.225.0 call signalling message containing **fastStart**. The calling endpoint must therefore be prepared to receive media on *any* of the receive channels it proposed in the Setup message, since it is possible for the media to be received prior to the H.225.0 call signalling message indicating precisely which channels will be used. Once a H.225.0 call signalling message containing **fastStart** is received by the calling endpoint, the calling endpoint may discontinue attempting to receive media on the channels for which proposals were not accepted by the called endpoint. Note that national requirements may prohibit called endpoints from transmitting media or limit the nature of the content of the media stream, prior to transmission of a Connect message; it is the responsibility of the endpoint to comply with applicable requirements. If the calling endpoint sets the **mediaWaitForConnect** element to TRUE in the Setup message, then the called endpoint shall not send any media until after the Connect message is sent.

The calling endpoint may begin transmitting media (according to the channels opened) immediately upon receiving a H.225.0 call signalling message containing **fastStart**. Thus, the called endpoint must be prepared to immediately receive media on the channels it accepted in the H.225.0 call signalling message containing **fastStart**. Note that national requirements may prohibit calling endpoints from transmitting media prior to receipt of a Connect message; it is the responsibility of the endpoint to comply with applicable requirements.

NOTE 1 – An entity shall not send an empty **fastStart** element in any message (i.e., a **fastStart** element shall contain at least one **OpenLogicalChannel** proposal). If an endpoint does receive a **fastStart** element that contains no **OpenLogicalChannel** proposals, it shall ignore the **fastStart** element.

NOTE 2 – When an endpoint or a gatekeeper intervening in call signalling receives a **fastStart** element in a Call Proceeding message, it will not be able to relay the Call Proceeding if the Call Proceeding message has already been sent to the originating side. In that case, the **fastStart** element in the Call Proceeding message shall be mapped to a **fastStart** element in a Facility message.

8.1.7.1 Proposal, selection and opening of media channels

The calling endpoint may propose multiple media channels or multiple alternative sets of characteristics for each media channel by encoding multiple **OpenLogicalChannel** structures within the **fastStart** element of the Setup message. Each **OpenLogicalChannel** structure within the **fastStart** element describes exactly one unidirectional media channel or one bidirectional media channel.

In the Setup message, each **OpenLogicalChannel** is a proposal to establish a media channel. **OpenLogicalChannel** proposals are included in the **fastStart** element in order of preference, with the most preferred alternatives listed first in the **fastStart** sequence; proposals to open audio channels shall be listed prior to channels for any other media types. In the H.225.0 call signalling message containing **fastStart** sent in response to Setup, each **OpenLogicalChannel** is an acceptance of a proposed media channel and indicates the channels that are established and can immediately be used for media transmission.

If an offered **dataType** element specifies encryption via the **h235Media** choice, the included **encryptionAuthenticationAndIntegrity** element may include an **encryptionCapability** element containing multiple encryption algorithms (including the NULL algorithm). This construct shall be taken to offer a choice of any one of the specified algorithms for encryption of the associated media capability.

In an **OpenLogicalChannel** that proposes a channel for transmission from the calling endpoint to the called endpoint, the **forwardLogicalChannelParameters** element shall contain parameters specifying the characteristics of the proposed channel, and the **reverseLogicalChannelParameters** element shall be omitted. Each such **OpenLogicalChannel** structure shall have a unique **forwardLogicalChannelNumber** value. Alternative proposals for the same transmit channel shall contain the same **sessionID** value in **H2250LogicalChannelParameters**. The **mediaChannel** element shall be omitted in the proposal; it will be provided by the called endpoint should the proposal be accepted. The other **H2250LogicalChannelParameters** and **dataType** shall be set to correctly describe the transmit capabilities of the calling endpoint associated with this proposed channel. The calling endpoint may choose not to propose any channels for transmission from the calling endpoint to the called endpoint, such as if it desires to use H.245 procedures later to establish such channels.

In the Setup message, each **OpenLogicalChannel** that proposes a unidirectional channel for transmission from the calling endpoint to the called endpoint and that will carry media using RTP shall contain the **mediaControlChannel** element (indicating the reverse RTCP channel) in the **H2250LogicalChannelParameters** element of the **forwardLogicalChannelParameters** structure.

In an **OpenLogicalChannel** that proposes a channel for transmission from the called endpoint to the calling endpoint, the **reverseLogicalChannelParameters** element shall be included and contain parameters specifying the characteristics of the proposed channel. The **forwardLogicalChannelParameters** element must also be included (because it is not optional), with the **dataType** element set to **nullData**, **multiplexParameters** set to **none**, and all optional elements omitted. Alternative proposals for the same receive channel shall contain the same **sessionID** value in **H2250LogicalChannelParameters**. All alternative **OpenLogicalChannel** structures, that propose a channel for transmission from the called endpoint to the calling endpoint, shall contain the same **sessionID** and the same **mediaChannel** value. The other **H2250LogicalChannelParameters** and **dataType** within **reverseLogicalChannelParameters** shall be set to correctly describe the receive capabilities of the calling endpoint associated with this proposed channel. The calling endpoint may choose not to propose any channels for transmission from the called endpoint to the calling endpoint, such as if it desires to use H.245 procedures later to establish such channels.

In the Setup message, each **OpenLogicalChannel** that proposes a unidirectional channel for transmission from the called endpoint to the calling endpoint and that will carry media using RTP shall contain the **mediaControlChannel** element (indicating the RTCP channel going in the same direction) in the **H2250LogicalChannelParameters** element of the **reverseLogicalChannelParameters** structure.

In an **OpenLogicalChannel** that proposes a bidirectional channel between the calling endpoint and the called endpoint, the **forwardLogicalChannelParameters** and **reverseLogicalChannelParameters** element shall contain parameters specifying the characteristics of the proposed channel. Each such **OpenLogicalChannel** structure shall have a unique **forwardLogicalChannelNumber** value. Alternative proposals for the same bidirectional channel shall contain the same **sessionID** value in **H2250LogicalChannelParameters**. The **mediaChannel** element shall be omitted in the proposal; it will be provided in the **reverseLogicalChannelParameters** element by the called endpoint should the proposal be accepted. The other **H2250LogicalChannelParameters** and **dataType** shall be set to correctly describe the transmit capabilities of the calling endpoint associated with this proposed channel.

All **mediaControlChannel** elements inserted by the calling endpoint for the same **sessionID** for both directions shall have the same value.

Upon receipt of a Setup message containing **fastStart**, determining that it is willing to proceed with the Fast Connect procedure, and reaching the point in the connection at which it is ready to begin media transmission, the called endpoint shall choose from amongst the proposed **OpenLogicalChannel** structures containing **reverseLogicalChannelParameters** elements for each media type it wants to transmit, from amongst the proposed **OpenLogicalChannel** structures specifying **forwardLogicalChannelParameters** (and omitting **reverseLogicalChannelParameters**) for each media type it wants to receive, and from amongst the proposed **OpenLogicalChannel** structures containing both **forwardLogicalChannelParameters** and **reverseLogicalChannelParameters** elements for each bidirectional channel it wants to transmit and or receive. If alternative proposals are presented, only one **OpenLogicalChannel** structure shall be selected from amongst each alternative set; alternatives within a set have the same **sessionID**. If multiple encryption algorithms are offered for a channel, the called endpoint must select one and modify the **OpenLogicalChannel** to remove the others. The called endpoint accepts a proposed channel by returning the corresponding **OpenLogicalChannel** structure in any H.225.0 call signalling message sent in response to Setup, up to and including Connect. A called endpoint may choose to repeat the **fastStart** element in all subsequent messages up to and including Connect: the contents of the **fastStart** element shall be the same. Calling endpoints shall react to the first **fastStart** element received in a response message to the Setup message and ignore any subsequent **fastStart** elements. The called endpoint may choose not to open media flow in a particular direction or of a particular media type by not including a corresponding **OpenLogicalChannel** structure in the **fastStart** element of the H.225.0 call signalling response.

When accepting a proposed channel for transmission from called endpoint to calling endpoint, the called endpoint shall return the corresponding **OpenLogicalChannel** structure to the calling endpoint, inserting a unique **forwardLogicalChannelNumber** into the **OpenLogicalChannel** structure and, for channels that will carry media using RTP, a valid **mediaControlChannel** element (indicating the reverse RTCP channel) in the **H2250LogicalChannelParameters** element of the **reverseLogicalChannelParameters** structure. The called endpoint may begin transmitting media on the accepted channel according to the parameters specified in **reverseLogicalChannelParameters** immediately after sending the H.225.0 call signalling response containing **fastStart**, unless **mediaWaitForConnect** was set to TRUE in which case it must wait until after sending the Connect message.

When accepting a proposed channel for transmission from the calling endpoint to the called endpoint, the called endpoint shall return the corresponding **OpenLogicalChannel** structure to the calling endpoint. The called endpoint shall insert valid a **mediaChannel** and, for channels that will carry media using RTP, a **mediaControlChannel** field (indicating the RTCP channel going in the same direction) in the **h2250LogicalChannelParameters** element of the **forwardLogicalChannelParameters** structure. All **mediaControlChannel** elements inserted by the called endpoint for the same **sessionID** for both directions shall have the same value. The called endpoint shall then prepare to immediately receive media flow according to the parameters specified in **forwardLogicalChannelParameters**. The calling endpoint may begin transmitting media on the accepted and opened channels upon receipt of the H.225.0 call signalling response containing **fastStart** and may release any resources allocated to reception on proposed channels that were not accepted.

When accepting a proposed bidirectional channel for transmission between the calling endpoint and the called endpoint, the called endpoint shall return the corresponding **OpenLogicalChannel** structure to the calling endpoint. The called and calling endpoints shall use the value in the **forwardLogicalChannelNumber** element as the logical channel number of the forward and reverse transmission paths of the bidirectional channel. The called endpoint shall insert a valid **mediaChannel** element in the **h2250LogicalChannelParameters** element of the **reverseLogicalChannelParameters** structure. The called and calling endpoints shall receive media flow according to the parameters specified in **forwardLogicalChannelParameters** and the **reverseLogicalChannelParameters**, respectively. The called endpoint shall be prepared to accept a connection for the bidirectional channel prior to returning the **fastStart** element. The calling endpoint may begin transmitting media on the accepted channels upon receipt of the H.225.0 call signalling response containing **fastStart** and may release any resources allocated for proposed channels that were not accepted.

NOTE – The called endpoint is only allowed to alter fields in a proposed **OpenLogicalChannel** structure as specified in this clause. An endpoint is not allowed, for example, to alter the number of frames per packet or other characteristics of the proposed channel not specifically stated in this clause. If the calling endpoint wants to increase the likelihood that the Fast Connect can be accepted, it should include multiple proposals with different alternative parameters. This rule does not preclude an endpoint from including **encryptionSync** in the returned **OpenLogicalChannel**.

8.1.7.2 Switching to H.245 procedures

After establishment of a call using the Fast Connect procedure, either endpoint may determine that it is necessary to invoke call features that require the use of H.245 procedures. Either endpoint may initiate the use of H.245 procedures at any point during the call using tunnelling as described in 8.2.1 (if **h245Tunnelling** remains enabled). An H.323 Version 4 or higher entity that uses Fast Connect in a call shall use H.245 tunnelling when an H.245 Control Channel is required and shall always set the **h245Tunnelling** field to TRUE. The process for switching to a separate H.245 connection is described in 8.2.3 and may be used by Version 3 or older entities or by newer H.323 entities when communicating with Version 3 or older entities for the purpose of maintaining backward compatibility.

When a call is established using the Fast Connect procedure, both endpoints shall keep the H.225.0 Call Signalling Channel open until either the call is terminated or, for compatibility with older endpoints, until a separate H.245 connection is established.

When H.245 procedures are activated, all mandatory procedures of H.245 that normally occur upon initiation of an H.245 connection shall be completed prior to initiation of any additional H.245 procedures. The media channels that were established in the Fast Connect procedure are "inherited" as though they had been opened using normal H.245 **openLogicalChannel** and **openLogicalChannelAck** procedures.

If the calling endpoint utilizes Fast Connect to initiate a call, it shall not open the H.245 Control Channel using the normal H.245 tunnelling or via a separate H.245 connection until the called endpoint has returned **fastStart**, **fastConnectRefused**, **h245Address**, or the Connect message. Note that older H.323 endpoints may open the H.245 Control Channel even before receiving one of these message elements or message, in spite of the fact that it initiated a Fast Connect call. While this behaviour was strongly discouraged in previous publications and is now forbidden, endpoints need to be aware of this older behaviour. If an endpoint opens the H.245 Control Channel before receiving the aforementioned message elements or message, the endpoint shall assume that Fast Connect is terminated and shall not send a **fastStart** element.

However, an endpoint may exchange the **terminalCapabilitySet** message and the **masterSlaveDetermination** message in the Setup message as described in 8.2.4. Such an exchange constitutes the opening of the H.245 Control Channel, but does not preclude either endpoint from proceeding with Fast Connect.

The called endpoint shall not initiate H.245 before returning **fastConnectRefused**, **fastStart**, or the Connect message. A called endpoint that returns the **h245Address** element in any message up to and including the Connect message, and which has not already explicitly accepted or rejected Fast Connect, shall also return either **fastStart** or **fastConnectRefused** in the same message. Note that older endpoints may not return **fastStart** or **fastConnectRefused**. For backward compatibility with older endpoints, H.323 endpoints may assume that Fast Connect is refused if the called endpoint sends the **h245Address** element or opens the H.245 Control Channel without simultaneously or previously sending **fastStart** or **fastConnectRefused**.

Note that in the case where a separate H.245 connection is opened from the called endpoint to the calling endpoint that supplied its **h245Address** in the Setup message, a race condition exists: the calling endpoint may detect the opening of the H.245 Control Channel from the called endpoint before it receives the **fastStart** element. For this reason, it is recommended that if an endpoint accepts Fast Connect and initiates a separate connection for H.245, it should introduce a delay between sending the H.225.0 message containing the **fastStart** element and the initiation of the separate H.245 connection. In the event that the called endpoint fails to introduce a delay, the calling endpoint should still be prepared for a possible late arrival of the **fastStart** element in this scenario. Older endpoints may assume that Fast Connect is refused if the H.245 Control Channel is opened prior to receiving the **fastStart** element.

8.1.7.3 Terminating a call

If a call connected using the Fast Connect procedure continues to completion without initiation of H.245 procedures, then the call may be terminated by either endpoint sending a H.225.0 call signalling Release Complete message. If H.245 procedures are initiated during the call, then the call is terminated as described in 8.5.

If a separate H.245 connection has not been established and the H.225.0 Call Signalling Channel is terminated, the call shall also be terminated.

8.1.7.4 In-band and out-of-band tones and announcements

Tones and announcements can be locally generated or passed in-band from the terminating endpoint.

On completing call setup, the endpoint on the terminating side shall decide if it will provide in-band tones or if locally generated tones at the originating side shall be used. Note that other type of indication can replace locally generated tones and announcement in some systems (visual indications on a screen for example). For the purpose of this clause, they will be referred to as locally generated tones and announcements. Locally generated tones, provided at the originating side, are the default. The terminating side may wish to provide in-band-generated tones and announcements, for example when the terminating endpoint is a gateway to an analogue network.

To instruct the originating side not to generate locally generated tones, such as ringback or busy, the terminating side shall open the media channel by responding to the Fast Connect request and sending a Progress indicator information element with progress descriptor #1, *Call is not end-to-end ISDN; further call progress information may be available in-band*, or #8, *In-band information or an appropriate pattern is now available* in a Call Proceeding, Progress or Alerting message, or in a Connect message if an Alerting message was not sent. The response to the Fast Connect message shall be done before or at the same time the Progress indicator is sent (i.e., up to and including the same message the Progress indicator is sent). The terminating side can provide in-band tones or announcements (such as ringback or busy) as soon as the progress descriptor has been sent and the media channel has been opened. Note that the Progress indicator should be in an Alerting message only if the endpoint is being alerted. If another in-band tone, such as busy or re-order tone is provided, the Progress indicator should not be in an Alerting. When no appropriate call setup message is available, a Progress message can be used to carry the Progress indicator.

NOTE – When an endpoint or a Gatekeeper intervening in call signalling receives a Progress indicator information element in a Call Proceeding message, it will not be able to relay the Call Proceeding if the Call Proceeding message has already been sent to the originating side. In that case, the Progress indicator information element in the Call Proceeding message shall be mapped to a Progress indicator information element in a Progress message.

If the terminating side does not wish to provide far-end tones and announcements, it shall not send a Progress indicator information element with progress descriptor #1 or #8. To instruct the originating side that locally generated alerting shall be applied, the Alerting message shall be sent.

Upon receipt of an Alerting message, the originating side shall provide locally generated tones and announcement unless both the following conditions are true:

1) A media channel is available for "listening". The **fastStart** element could have been received in any message up to and including Alerting message.

2) A Progress indicator information element with progress descriptor #1, *Call is not end-to-end ISDN; further call progress information may be available in-band*, or #8, *In-band information or an appropriate pattern is now available*, was received in any message up to and including the Alerting message.

Upon receipt of a Release Complete message including a Cause information element, the originating side shall generate a tone or provide an indication appropriate to the received cause value. For example, if cause value #17, *User busy*, is received, the originating shall generate busy tone or provide an indication of user busy.

When locally generated tones and announcements are used, the Signal information element can optionally also be present to include more information about the type of signal to be provided.

8.1.8 Call setup via gateways

8.1.8.1 Gateway in-bound call setup

When an external terminal calls a network endpoint via the Gateway, call setup between the Gateway and the network endpoint proceeds the same as the endpoint-to-endpoint call setup. The Gateway may need to issue a Call Proceeding message to the external terminal while establishing the call on the network.

A Gateway which cannot directly route an incoming SCN call to an H.323 endpoint shall be able to accept two-stage dialling. For Gateways to H.320 networks (also H.321, H.322 and H.310 in H.321 mode), the Gateway shall accept SBE numbers from the H.320 terminal. Optionally, Gateways to H.320 networks may support the TCS-4 and IIS BAS codes to retrieve the H.323 dialling information after a H.320 call has been established. For Gateways to H.310 native mode and H.324 networks, the Gateway shall accept H.245 **userInputIndication** messages from the H.324 terminal. In these two cases, support of DTMF is optional. For Gateways to speech-only

terminals, the Gateway shall accept DTMF numbers from the speech-only terminal. These numbers will indicate a second stage dialling number to access the individual endpoint on the network.

8.1.8.2 Gateway out-bound call setup

When a network endpoint calls an external terminal via the Gateway, call setup between the network endpoint and the Gateway proceeds the same as the endpoint-to-endpoint call setup. The Gateway will receive the destination **dialledDigits** or **partyNumber** (**e164Number** or **privateNumber**) in the Setup message. It will then use this address to place the out-bound call. The Gateway may return Call Proceeding messages to the network endpoint while establishing the outgoing call.

A Gateway should send a Call Proceeding message after it receives the Setup message (or after it receives ACF) if it expects more than 4 seconds to elapse before it can respond with Alerting, Connect, or Release Complete.

The Progress Indicator information element is used to indicate that inter-networking is occurring. The Gateway shall issue a Progress indicator information element within the Alerting, Call Proceeding or Connect messages. This information may also be sent in a Progress message.

The network endpoint shall send all **dialledDigits** or **partyNumber** addresses that it is calling in the Setup message. For example, a six B-channel call on the ISDN will require six **dialledDigits** or **partyNumber** addresses in the Setup message. The Gateway shall respond to the Setup message with a Connect or Release Complete message as well as Alerting, Call Proceeding, or Progress messages. Failure of the SCN call shall be reported to the network endpoint in the Release Complete message. The use of multiple CRV values and multiple Setup messages is for further study. Addition of channels on the SCN during a call is for further study.

A network endpoint that is registered with a Gatekeeper should request sufficient call bandwidth in the ARQ message for the aggregate of all SCN calls. If sufficient call bandwidth was not requested in the ARQ message, the procedures of 8.4.1, Bandwidth Changes, shall be followed in order to obtain additional call bandwidth.

The Gateway may advance to Phase B after placing the first call on the SCN. Additional calls for the additional SCN **dialledDigits** or **partyNumber** numbers may be placed after the capability exchange with the Gateway and establishment of audio communications with the SCN endpoint.

8.1.9 Call setup with an MCU

For Centralized Multipoint Conferences, all endpoints exchange call signalling with the MCU. Call setup between an endpoint and the MCU proceeds the same as the endpoint-to-endpoint call setup scenarios of 8.1.1 through 8.1.5. The MCU may be the called endpoint or the calling endpoint.

In a Centralized Multipoint Conference, the H.245 Control Channel is opened between the endpoints and the MC within the MCU. The audio, video, and data channels are opened between the endpoints and the MP within the MCU. In a Decentralized Multipoint Conference, the H.245 Control Channel is open between the endpoint and the MC (there may be many such H.245 Control Channels, one for each call). The Audio and Video Channels should be multicast to all endpoints in the conference. The Data Channel shall be opened with the Data MP.

In an ad hoc Multipoint Conference where the endpoints do not contain an MC and the Gatekeeper would like to provide an Ad Hoc multipoint service for the endpoints, the H.245 Control Channel may be routed through the Gatekeeper. Initially, the H.245 Control Channel would be routed between the endpoints through the Gatekeeper. When the conference switches to multipoint, the Gatekeeper may connect the endpoints to an MC associated with the Gatekeeper.

In an ad hoc Multipoint Conference where one or both of the endpoints contains an MC, the normal call setup procedures defined in 8.1.1 through 8.1.5 are used. These procedures may apply even if an endpoint that contains an MC is actually a MCU. The master-slave determination procedure is used to determine which MC will be the Active MC for the conference.

8.1.10 Call forwarding

An endpoint wishing to forward a call to another endpoint may issue a Facility message indicating the address of the new endpoint. The endpoint receiving this Facility indication should send a Release Complete and then restart the Phase A procedures with the new endpoint.

8.1.11 Broadcast call setup

Call setup for loosely controlled Broadcast and Broadcast Panel conferences shall follow the procedures defined in ITU-T Rec. H.332.

8.1.12 Overlapped sending

H.323 entities can optionally support overlap sending. If a Gatekeeper is present, and overlap sending is being used, endpoints should send an ARQ message to the Gatekeeper each time some new addressing information is input. The endpoint shall place the total cumulative addressing information into the **destinationInfo** field each time an ARQ message is sent. If there is insufficient addressing information in the ARQ, the Gatekeeper should respond with an ARJ with the **reason** set to **incompleteAddress**. This indicates that the endpoint should send another ARQ when more addressing information is available. When a Gatekeeper has sufficient addressing information to assign a suitable **destCallSignalAddress**, it shall return an ACF. Note that this does not necessarily mean that the addressing information is complete. If the Gatekeeper sends an ARJ with **AdmissionRejectReason** set to something other than **incompleteAddress**, the call setup process shall be aborted.

When an endpoint has a suitable **destCallSignalAddress**, it shall send a Setup message with the **canOverlapSend** field assigned according to whether it is capable of supporting the overlap sending procedures. If a remote entity receives a Setup message with an incomplete address and the **canOverlapSend** field set to TRUE, it should initiate overlap sending procedures by returning the Setup Acknowledge message. Additional addressing information should be sent using Information messages. If the address is incomplete and the **canOverlapSend** field set to FALSE, the remote entity should send Release Complete. Note that Gateways should not transfer Setup Acknowledge messages from the SCN to H.323 endpoints that have not indicated that they can support overlap sending procedures as the desired result may not be achieved.

8.1.13 Call setup to conference alias

Alias addresses (see 7.1.3) may be used to represent a conference at an MC. The procedures in the preceding subclauses apply, except as noted here.

8.1.13.1 Joining to a conference alias, with no gatekeeper

Endpoint 1 (calling endpoint) sends the Setup (1) message (see Figure 29) to the well-known Call Signalling Channel TSAP Identifier of Endpoint 2 (the MC). The Setup message includes the following fields:

```
destinationAddress          = conferenceAlias
destCallSignalAddress       = MC(U) transport address
conferenceID                = 0 (since the CID is unknown)
conferenceGoal              = join
```

Endpoint 2 responds with the Connect (4) message, which contains:

```
h245Address                 = Transport Address for H.245 signalling
conferenceID                = CID for the conference
```

8.1.13.2 Joining to a conference alias, with gatekeeper

Endpoint 1 (calling endpoint) initiates the ARQ (1)/ACF (2) exchange (reference Figure 30) with the Gatekeeper. The ARQ contains:

```
destinationInfo             = conferenceAlias
callIdentifier              = some value N
conferenceID                = 0 (since the CID is unknown)
```

The Gatekeeper shall return the Call Signalling Channel Transport Address of Endpoint 2 (called endpoint, containing the MC) in the ACF. Endpoint 1 then sends the Setup (3) message to Endpoint 2 using that Transport Address and the following fields:

```
destinationAddress          = conferenceAlias
destCallSignalAddress       = address supplied by ACF
conferenceID                = 0
conferenceGoal              = join
```

Ultimately, Endpoint 2 returns a Connect message with following fields:

```
h245Address                 = Transport Address for H.245 signalling
conferenceID                = CID for the conference
```

Endpoint 1 completes the call by informing its Gatekeeper of the correct CID. Endpoint 1 sends an IRR to the Gatekeeper with the following fields:

```
callIdentifier              = same value N as used in the first ARQ
conferenceID                = original CID from endpoint 1
substituteConferenceIDs     = CID from endpoint 2
```

8.1.13.3 Create or invite with a conference alias

Endpoint 1 (calling endpoint) may send a Setup message to Endpoint 2. The Setup message includes the following fields:

```
destinationAddress          = conferenceAlias
destCallSignalAddress       = MC(U) transport address
conferenceID                = CID of the conference
conferenceGoal              = create or invite
```

Endpoint 2 responds with the Connect message, which contains:

```
h245Address                 = Transport Address for H.245 signalling
conferenceID                = CID for the conference
```

8.1.13.4 Consideration for version 1 endpoints

When an H.323 entity (endpoint or MCU) receives a Setup message from a Version 1 entity and the **destinationAddress** matches one of its conferences aliases, then it shall ignore the **conferenceGoal** and treat the Setup request as a join request.

When a Gatekeeper receives an ARQ, from a Version 1 entity and the **destinationInfo** matches one of its conferences aliases, then it shall ignore the **conferenceID** field. Likewise, when an H.323 entity receives a Setup message from a Version 1 entity and the **destinationAddress** matches one of its conferences aliases, then it shall ignore the **conferenceID**.

These provisions allow a Version 1 endpoint to call a conference Alias.

8.1.14 Gatekeeper modification of destination addresses

An endpoint shall set the **canMapAlias** field to TRUE to indicate its ability to accept modified destination information from a Gatekeeper. The endpoint shall use the destination information returned in ACF or LCF instead of the destination information passed in the ARQ or LRQ. For an ingress Gateway, the destination information that appears in the ACF will be used in the Setup message that is sent on the packet network. For an egress Gateway, the destination information that appears in the ACF will be used to address a destination in the GSTN (for example, appearing in the Setup message sent to the ISDN).

In Gatekeeper routed cases, the Gatekeeper may modify destination addresses in the Setup message it receives before sending out a corresponding Setup message.

NOTE – H.323 systems prior to Version 4 were not required to set the **canMapAlias** field to TRUE.

8.1.15 Indicating desired protocols

When an endpoint places a call, it may indicate in various H.225.0 messages those protocols it desires to utilize during the course of a call, such as fax, H.320, T.120, etc., in the **desiredProtocols** field. If the endpoint provides a list of desired protocols to its Gatekeeper or if an entity sends an LRQ message to a Gatekeeper with a list of desired protocols, the Gatekeeper should attempt to locate an endpoint that can provide support for the desired protocols. If the Gatekeeper finds no endpoint that supports any of the desired protocols, the Gatekeeper shall still resolve the address so that the call may continue.

The calling endpoint may examine the **EndpointType** of the destination endpoint to determine exactly what protocols the remote endpoint possesses.

8.1.16 Gatekeeper requested tones and announcements

A Gatekeeper may request a Gateway to play a tone or an announcement for a variety of call events. These call events could be "pre-call" events (something that happens before the terminating gateway is signalled, such as prompting the caller for a destination number or an account code), "mid-call" events (something that happens in the middle of a call, such as providing an announcement to alert the parties on the call that the call will end in a few minutes), or "end-call" events (something that happens at the end of the call, such as a farewell message). In all cases, the Gatekeeper may use a **H248SignalsDescriptor** to describe the prompt the gateway should use.

The following pre-call events are supported:

- Prompting for a destination – In what is often called two-stage dialling, the caller dials one number to reach the Gateway and is prompted to dial the true destination number. Although a Gateway may have a general policy to always provide the prompt, in some circumstances it may make sense to allow the Gateway to consult the Gatekeeper. This "consult" operation is simply the ARQ with the called number as **destinationInfo**. If the Gatekeeper decides that a true destination number is required, the Gatekeeper may instruct the Gateway to prompt the caller, collect the additional digits, and consult the Gatekeeper with the destination. The Gatekeeper uses the ARJ with a **serviceControl** element and an **AdmissionRejectReason** of **collectDestination**. The **serviceControl** element has a **ServiceControlDescriptor** of type **signal** (which contains the **H248SignalsDescriptor**) and a **reason** of **open**. The **AdmissionRejectReason** of **collectDestination** instructs the Gateway to place the collected true destination into the **destinationInfo** of a new ARQ.

- Prompting for an authorization code, account code, or PIN – In this case, the Gatekeeper replies to the ARQ with an ARJ containing a **serviceControl** element and an **AdmissionRejectReason** of **collectPIN**. The **serviceControl** element has a **ServiceControlDescriptor** of type **signal** (which contains the **H248SignalsDescriptor**) and a reason of open. The **AdmissionRejectReason** of **collectPIN** instructs the Gateway to place the collected PIN (or authorization code or account code) into a token or **cryptoToken** of a new ARQ.
- Prompting for both a destination and a PIN – This is simply a serial operation of the first two cases.
- Rejecting a call – A Gatekeeper may choose to reject a call, but provide some feedback to the user (for example, providing a network busy tone or announcement if there are no available facilities for a destination). In this case, the ARJ would contain an **AdmissionRejectReason** that reflects the condition, but not **collectPIN** or **collectDestination**.

A Gatekeeper may initiate a mid-call signal by using the SCI message. The **serviceControl** element has a **ServiceControlDescriptor** of type **signal** (which contains the H.248 **H248SignalsDescriptor**) and a **reason** of **open**. The signal may be stopped by sending the **ServiceControlIndication** message, but with a **ServiceControlDescriptor** containing a **reason** of **close**. A Gateway should respond to the SCI message with a SCR with an appropriate **result**.

A Gatekeeper may initiate an end-call signal in a DRQ (for the direct endpoint routing case) or a Release Complete (for the Gatekeeper routed case) with a **serviceControl** element. The **serviceControl** element has a **ServiceControlDescriptor** of type **signal** (which contains the H.248 **H248SignalsDescriptor**) and a **reason** of **open**. The signal may be stopped by sending the **ServiceControlIndication** message, but with a **ServiceControlDescriptor** containing a **reason** of **close**.

8.2 Phase B – Initial communication and capability exchange

Once both sides have exchanged call setup messages from Phase A, the endpoints shall, if they plan to use H.245, establish the H.245 Control Channel. The procedures of ITU-T Rec. H.245 are used over the H.245 Control Channel for the capability exchange and to open the media channels.

NOTE – Optionally, the H.245 Control Channel may be set up by the called endpoint on receipt of Setup and by the calling endpoint on receipt of Alerting or Call Proceeding. In the event that Connect does not arrive or an endpoint sends Release Complete, the H.245 Control Channel shall be closed.

Endpoints shall support the capabilities exchange procedure of H.245 as described in 6.2.8.1.

Endpoint system capabilities are exchanged by transmission of the H.245 **terminalCapabilitySet** message. This capability message shall be the first H.245 message sent unless the endpoint is indicating that it understands the **parallelH245Control** field (see 8.2.4). If prior to successful completion of terminal capability exchange, any other procedure fails (i.e., rejected, not understood, not supported), then the initiating endpoint should initiate and successfully complete terminal capability exchange before attempting any other procedure. An endpoint which receives a **terminalCapabilitySet** message from a peer prior to initiating capabilities exchange shall respond as required by 6.2.8.1 and should initiate and successfully complete capabilities exchange with that peer prior to initiating any other procedure.

Endpoints shall support the master-slave determination procedure of H.245 as described in 6.2.8.4. In cases where both endpoints in a call have MC capability, the master-slave determination is used for determining which MC will be the Active MC for the conference. The Active MC may then send the **mcLocationIndication** message. The procedure also provides master-slave determination for opening bidirectional channels for data.

Master-slave determination shall be advanced (by sending either **MasterSlaveDetermination** or **MasterSlaveDeterminationAck** as appropriate) in the first H.245 message after Terminal Capability Exchange has been initiated.

If the initial capability exchange or master-slave determination procedures fail, these should be retried at least two additional times before the endpoint abandons the connection attempt and proceeds to Phase E.

Following successful completion of the requirements of Phase B, the endpoints shall proceed directly to the desired operating mode, normally Phase C.

8.2.1 Encapsulation of H.245 messages within H.225.0 Call Signalling messages

In order to conserve resources, synchronize call signalling and control, and reduce call setup time, it may be desirable to convey H.245 messages within the H.225.0 call signalling Call Signalling Channel instead of establishing a separate H.245 channel. This process, known as "encapsulation" or "tunnelling" of H.245 messages, is accomplished by utilizing the **h245Control** element of **h323-uu-pdu** on the Call Signalling Channel, copying an encoded H.245 message as an octet string.

When tunnelling is active, one or more H.245 messages can be encapsulated in any H.225.0 call signalling message. If tunnelling is being utilized and there is no need for transmission of a H.225.0 call signalling message at the time an H.245 message must be transmitted, then a Facility message shall be sent with the **reason** set to **transportedInformation**. (Note that H.323 systems prior to version 4 used a Facility message with **h323-message-body** set to **empty**.)

A calling entity capable of and willing to use H.245 encapsulation shall set the **h245Tunnelling** element to TRUE in the Setup message and any subsequent H.225.0 call signalling messages it sends so long as it desires tunnelling to remain active. A called entity capable of and willing to use H.245 encapsulation shall set the **h245Tunnelling** element to TRUE in the first H.225.0 call signalling message sent in response to Setup and in every subsequent H.225.0 call signalling message it sends so long as it desires tunnelling to remain active. The called entity shall not set **h245Tunnelling** to TRUE in any H.225.0 call signalling response (and tunnelling remains disabled) unless it was TRUE in the Setup message to which it is responding. If the called entity does not yet know if H.245 tunnelling can be supported, it shall include the **provisionalRespToH245Tunnelling** flag. This may happen, for example, when a Gatekeeper is responding to a calling entity with a message such as Call Proceeding before the called endpoint responds to the **h245Tunnelling** flag. The **provisionalRespToH245Tunnelling** flag effectively eliminates the meaning of the **h245Tunnelling** flag in a message and the flag shall thus be ignored by the receiving endpoint.

If **h245Tunnelling** is not set to TRUE in any H.225.0 call signalling message that does not include the **provisionalRespToH245Tunnelling** flag, then tunnelling is disabled from that point for the duration of the call and a separate H.245 connection shall be established when and if H.245 procedures are invoked.

The calling entity may include tunnelled H.245 messages in the Setup message; it must also set the **h245Tunnelling** element to TRUE. If the called entity does not set **h245Tunnelling** to TRUE and the **provisionalRespToH245Tunnelling** flag is absent in the first H.225.0 call signalling message sent in response to Setup, then the calling entity shall assume that the H.245 messages it had encapsulated in Setup were ignored by the called entity and repeat them, as necessary, after the separate H.245 channel is established. The called entity, if it sets **h245Tunnelling** to TRUE, may also include encapsulated H.245 messages in the first and subsequent H.225.0 call signalling messages.

The calling endpoint shall not include both a **fastStart** element and encapsulated H.245 messages in **h245Control** in the same Setup message, since the presence of the encapsulated H.245 messages would override the Fast Connect procedure. A calling endpoint may, however, include both a **fastStart** element and set **h245Tunnelling** to TRUE within the same Setup message; likewise, a called endpoint may include **fastStart** and set **h245Tunnelling** to TRUE within the same H.225.0 call signalling response. In this case, the Fast Connect procedures are followed, and the H.245 connection remains "unestablished" until actual transmission of the first tunnelled H.245 message or opening of the separate H.245 connection.

When H.245 encapsulation is being used, both endpoints shall keep the H.225.0 Call Signalling Channel open until either the call is terminated or a separate H.245 connection is established.

When an endpoint receives an **h245control** element encapsulating more than one H.245 PDU, the encapsulated H.245 PDUs shall be processed (i.e., provided to higher layers) sequentially by order of increasing offset from the beginning of the H.225.0 message.

H.323 Version 4 and higher entities shall indicate support for H.245 tunnelling as described in this clause by setting the **h245Tunnelling** field to TRUE in all messages containing this field.

8.2.2 Tunnelling through intermediate signalling entities

Entities in the signalling path such as Gatekeepers may perform functions such as divert on no-reply or other advanced call control that results in representing to an endpoint a call state that is different from the actual call state at the other endpoint. Such intermediate entities shall ensure that H.245 messages encapsulated in H.225.0 call signalling messages are forwarded to the other endpoint even if the H.225.0 call signalling message in which the H.245 message is encapsulated would be consumed and not forwarded to the other endpoint. This is accomplished by transferring the encapsulated H.245 message into a Facility message with the **reason** set to **transportedInformation**. (Note that H.323 systems prior to version 4 used the Facility message with the **h323-message-body** set to **empty**.) For example, if a Gatekeeper has already sent a Connect message to a calling endpoint and later receives a Connect message from a called endpoint that contains an encapsulated H.245 message, it must forward the H.245 message using a Facility message.

Entities in the signalling path shall also use the Facility message or the Progress message to convey any new information (such as Q.931 information elements, CallProceeding-UUIE fields, tunnelled non-H.323 protocols, and encapsulated H.245 messages) received in a Call Proceeding message to the other endpoint if the entity has already sent a Call Proceeding message. This will allow the entity, for example, to transmit the **fastStart** element to facilitate proper establishment of a Fast Connect call and/or a Progress Indicator to indicate the presence of in-band tones and announcements. When using the Facility message to carrying such information extracted from the Call Proceeding message, the **reason** in the Facility should be set to **forwardedElements**.

8.2.3 Switching to a separate H.245 connection

When H.245 encapsulation or Fast Connect is being used, either endpoint may choose to switch to using the separate H.245 connection at any time. In order to facilitate initiation of the separate H.245 connection by either endpoint, each endpoint may include **h245Address** in any H.225.0 call signalling message it sends during the call. If, at the time an endpoint deems it necessary to initiate the separate H.245 connection, it finds that it has not yet received the **h245Address** of the other endpoint, the endpoint shall transmit a Facility message with a **FacilityReason** of **startH245** and provide its H.245 address in the **h245Address** element. An endpoint receiving a Facility message with a **facilityReason** of **startH245** which has not already independently initiated the separate H.245 channel shall open the H.245 channel using the **h245Address** specified. Use of the separate H.245 connection is initiated by opening the H.245 TCP connection and accepted by acknowledgement of the H.245 TCP connection.

If tunnelling was being used, the endpoint initiating the separate H.245 connection shall not send any further tunnelled H.245 messages on the Call Signalling Channel and shall send no H.245 messages on the separate H.245 connection until the establishment of the TCP connection is acknowledged. The endpoint acknowledging opening of the separate H.245 connection shall not send any further tunnelled H.245 messages on the Call Signalling Channel after acknowledging the opening of the separate H.245 connection. Because of the possibility that H.245 messages have already been sent and are in transit when the separate H.245 channel is initiated, endpoints shall continue to receive and correctly process tunnelled H.245 messages until a H.225.0 call signalling message is received with the **h245Tunnelling** flag set to FALSE; responses to such "late" tunnelled H.245 messages or acknowledgement of such messages shall be sent on the separate H.245 connection after it is established. Once a separate H.245 connection has been established, it is not possible to switch back to using tunnelling.

In the event that both endpoints simultaneously initiate the separate H.245 connection, the endpoint with the numerically smaller **h245Address** shall close the TCP connection it opened and use the connection opened by the other endpoint. For purposes of comparing the numeric values of **h245Address**, each octet of the address shall be individually compared beginning with the first octet of the OCTET STRING and continuing through the OCTET STRING left to right until unequal numeric octet values are found. Comparison shall first be performed on the network-layer address element of **h245Address** and, if found to be equal, then on the transport (port) address element.

8.2.4 Initiating H.245 tunnelling in parallel with fast connect

As detailed in 8.2, the first two H.245 messages sent by an endpoint on the H.245 Control Channel are the **terminalCapabilitySet** message and the **masterSlaveDetermination** message. Even when Fast Connect is being utilized, there are advantages to exchanging those messages as quickly as possible. In particular, an entity may need to know as early as possible whether DTMF is supported in **UserInputIndication** or with RTP payload types (as described in 10.5) by the other entity. Additionally, if Fast Connect is refused, there are obvious advantages to having already transmitted these messages, as there are fewer messages to exchange in order to open logical channels.

Therefore, to expedite the exchange of capabilities and overall call setup, an entity may include the H.245 **terminalCapabilitySet** message and the **masterSlaveDetermination** message in the Setup message by including those messages in the **parallelH245Control** field of the Setup message. Unlike the **h245Control** field, the calling entity may send these messages in the Setup message along with the **fastStart** element. The calling entity shall set the **h245Tunnelling** field to TRUE when including the **parallelH245Control** field.

NOTE – A calling entity should not include the **parallelH245Control** field without also including the **fastStart** field, since H.245 tunnelling in the context of a call that does not utilize the Fast Connect procedures should be handled according to 8.2.1.

To indicate that the called entity understands the **parallelH245Control** field, the first H.245 message that the called entity sends shall be the **terminalCapabilitySetAck** message tunnelled in the H.225.0 Call Signalling Channel. This response message should be sent by the called entity at the same time that **fastConnectRefused** or **fastStart** is sent to the calling entity. Note that if an endpoint does not indicate that it understands the **parallelH245Control** field, it shall abide by 8.2 and send **terminalCapabiltySet** and not **terminalCapabilitySetAck** as the first H.245 message. The called entity shall set the **h245Tunnelling** field to TRUE if it understands the **parallelH245Control** field. Figure 41 shows the message exchanges of a Fast Connect call between two endpoints that understand the **parallelH245Control** field.

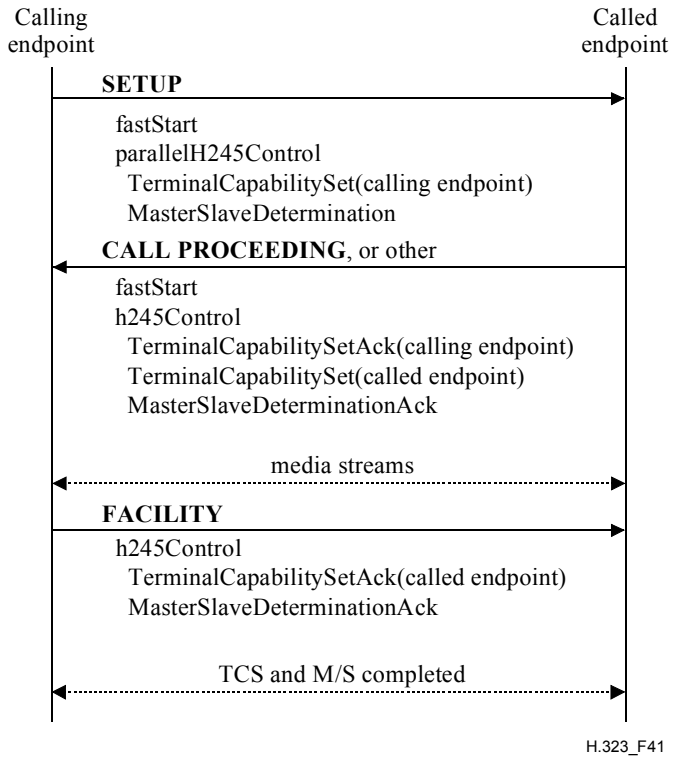


Figure 41/H.323 – Successful initiation of H.245 in parallel with fast connect

The calling entity shall recognize that the **parallelH245Control** field was not understood when either it receives a Connect message and still has not received a response to the initial **terminalCapabilitySet** message, the first H.245 message received from the called entity is not a tunnelled **terminalCapabilitySetAck** message, or **fastStart** or **fastConnectRefused** is received and no response has been received for the **terminalCapabilitySet** message. Figure 42 shows a message exchange between an endpoint that sends the **parallelH245Control** field and a called endpoint that does not understand the field.

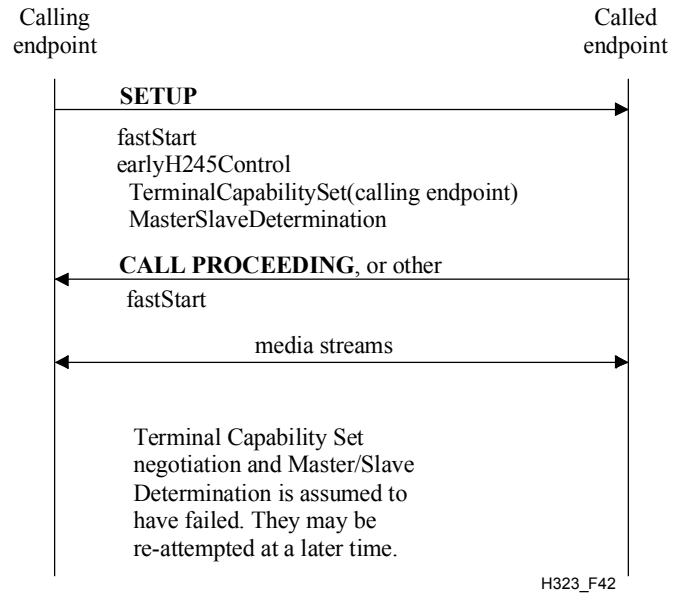


Figure 42/H.323 – Unsuccessful initiation of H.245 in parallel with fast connect

8.3 Phase C – Establishment of audiovisual communication

Following the exchange of capabilities and master-slave determination, the procedures of ITU-T Rec. H.245 shall then be used to open logical channels for the various information streams. The audio and video streams, which are transmitted in the logical channels setup in H.245, are transported over dynamic TSAP Identifiers using an unreliable protocol (see ITU-T Rec. H.225.0). Data communications which is transmitted in the logical channels setup in H.245, are transported using a reliable protocol (see ITU-T Rec. H.225.0).

The **openLogicalChannelAck** message returns, or the **reverseLogicalChannelParameters** of the **openLogicalChannel** request contains, the Transport Address that the receiving endpoint has assigned to that logical channel. The transmitting channel shall send the information stream associated with the logical channel to that Transport Address.

Following the opening of logical channels for audio and video, one **h2250MaximumSkewIndication** message shall be sent by the transmitter for each associated audio and video pair.

8.3.1 Mode changes

During a session, the procedures for changing channel structure, capability, receive mode, etc., shall be carried out as defined in ITU-T Rec. H.245. Appendix V/H.245 contains a procedure for changing modes on a logical channel which may minimize the disruption of the audio.

8.3.2 Exchange of video by mutual agreement

The indication **videoIndicateReadyToActivate** is defined in ITU-T Rec. H.245. Its use is optional, but when used the procedure shall be as follows.

Endpoint 1 has been set so that video is not transmitted unless and until Endpoint 2 has also indicated readiness to transmit video. Endpoint 1 shall send the indication **videoIndicateReadyToActivate** when the initial capability exchange has been completed, but shall not transmit a video signal until it has received either **videoIndicateReadyToActivate** or incoming video from Endpoint 2.

An endpoint which has not been set in this optional way is not obliged to wait until receipt of **videoIndicateReadyToActivate** or video before initiating its video transmission.

8.3.3 Media stream address distribution

In unicast, the endpoint shall open logical channels to the MCU or other endpoint. Addresses are passed in the **openLogicalChannel** and **openLogicalChannelAck**.

In multicast, the multicast addresses are assigned by the MC and distributed to the endpoints in the **communicationModeCommand**. It is the responsibility of the MC to allocate and assign unique multicast addresses. The endpoint shall signal an **openLogicalChannel** to the MC with the assigned multicast address. The MC shall forward the **openLogicalChannel** to each receiving endpoint. In cases where media from multiple endpoints are transmitted on a single session (e.g., single multicast address), the MC shall open a logical channel to each endpoint receiving media from an endpoint in the conference.

In cases where an endpoint joins a conference after the initial **communicationModeCommand** has been transmitted, it is the responsibility of the MC to send an updated **communicationModeCommand** to the new endpoint and to open the appropriate logical channels for media sourced from the new endpoint. In cases where an endpoint leaves the conference after the initial **communicationModeCommand** has been transmitted, it is the responsibility of the MC to close the appropriate logical channels which were being sourced from the endpoint which left the conference.

In multi-unicast, the endpoint must open logical channels to each of the other endpoints. The **openLogicalChannel** is sent to the MC and shall contain the terminal number of the endpoint for which the channel is intended. The endpoint can match a **openLogicalChannelAck** by the **forwardLogicalChannelNumber**.

8.3.4 Correlation of media streams in multipoint conferences

The following method shall be used to associate a logical channel with an RTP stream within a multipoint conference. The media stream source endpoint sends the **openLogicalChannel** message to the MC. In cases where the source would like to indicate a destination for the **openLogicalChannel**, the source endpoint should place the **terminalLabel** of the destination endpoint in the destination field of the **h2250LogicalChannelParameters**. The source endpoint shall also place its own **terminalLabel** in the **source** field of **h2250LogicalChannelParameters**. Note that in the multicast model, the absence of a **destination** indicates that the stream is applicable to all endpoints.

If a source endpoint has been assigned a **terminalLabel** by an MC, the source endpoint shall use an SSRC that contains the lowest byte of its **terminalLabel** as the lowest byte of its SSRC.

The destination endpoint may associate the logical channel number with the RTP stream source by comparing the **openLogicalChannel.h2250LogicalChannelParameters.source** field with the lowest byte of the SSRC in the RTP header.

It is possible for SSRC collisions when an H.323 endpoint is in an H.332 conference. The endpoint detecting the collision shall follow the procedures in RTP for SSRC collision resolution.

8.3.5 Communication mode command procedures

The H.245 **communicationModeCommand** is sent by an H.323 MC to specify the communication mode for each media type: unicast or multicast. This command may cause a switch between a centralized and decentralized conference and therefore may involve closing all existing logical channels and opening new ones.

The **communicationModeCommand** specifies all the sessions in the conference. For each session, the following data are specified: the RTP session identifier, the associated RTP session ID if applicable, a terminal label if applicable, a description of the session, the **dataType** of the sessions (e.g., G.711), and a unicast or multicast address for the media and media control channels as appropriate for the conference configuration and type.

The **communicationModeCommand** conveys the transmit modes which conference endpoints are to use in a conference. The command does not convey receive modes, as they are specified by **openLogicalChannel** commands which are sent from the MC to the endpoints.

It is presumed that the **communicationModeCommand** is defining the modes of a conference and is therefore sent after the **multipointConference** indication which notifies an endpoint that it must comply with the commands of the MC. Endpoints should wait for a **communicationModeCommand** before opening logical channels when they have received a **multipointConference** indication.

Endpoints receiving a **communicationModeCommand** use the **terminalLabel** field of each table entry to determine if the entry is applicable for its own processing. Entries which do not contain a **terminalLabel** apply to all endpoints in the conference. Entries which contain **terminalLabels** are commands to specific endpoints which match the **terminalLabel** in the entry. For example, when audio streams from all endpoints are placed on one multicast address (one session), the table entry for the audio mode, media address, and media control address will not contain a **terminalLabel**. When the table entry commands an endpoint to send its video to a multicast address, the MC will include that endpoint's **terminalLabel**.

The **communicationModeCommand** can be used to instruct endpoints in a conference (or a point-to-point call) to change modes by indicating a new mode for a **mediaChannel** that is already in use. It can also be used to tell an endpoint to transmit the media stream to a new address by indicating the mode currently in use, but with new **mediaChannel**. Similarly, an endpoint that receives a **communicationModeCommand** indicating the mode currently in use and no **mediaChannel** should close the appropriate channel and the attempt to reopen using the **openLogicalChannel-openLogicalChannelAck** sequence, where the **openLogicalChannelAck** contains the address to which the endpoint will send the media.

Appendix I contains examples of the **communicationModeTable** entries for various cases.

8.4 Phase D – Call services

8.4.1 Bandwidth changes

Call bandwidth is initially established and approved by the Gatekeeper during the admissions exchange. An endpoint shall assure that the aggregate for all transmitted and received audio and video channels, excluding any RTP headers, RTP payload headers, network headers, and other overhead, is within this bandwidth. Data and control channels are not included in this limit.

At any time during a conference, the endpoints or Gatekeeper may request an increase or decrease in the call bandwidth. An endpoint may change the bit rate of a logical channel without requesting a bandwidth change from the Gatekeeper if the aggregate bit rate of all transmitted and received channels does not exceed the current call bandwidth. If the change will result in a aggregate bit rate that exceeds the current call bandwidth, the endpoint shall request a change in the call bandwidth from its Gatekeeper and await confirmation prior to actually increasing any bit rate. A bandwidth change request is recommended when an endpoint will use a reduced bandwidth for an extended period of time, thus freeing up bandwidth for other calls.

An endpoint wishing to change its call bandwidth sends a Bandwidth Change Request (BRQ) message (1) to the Gatekeeper. The Gatekeeper determines if the request is acceptable. The criteria for this determination is outside the scope of this Recommendation. If the Gatekeeper determines

that the request is not acceptable, it returns a Bandwidth Change Reject (BRJ) message (2) to endpoint. If the Gatekeeper determines that the request is acceptable, it returns a Bandwidth Change Confirm (BCF) message (2).

If Endpoint 1 wishes to increase its transmitted bit rate on a logical channel, it first determines if the call bandwidth will be exceeded. See Figure 43. If it will, Endpoint 1 shall request a bandwidth change (1 and 2) from Gatekeeper 1. When the call bandwidth is sufficient to support the change, Endpoint 1 sends a **closeLogicalChannel** (3) message to close the logical channel. It then reopens the logical channel using the **openLogicalChannel** (4) specifying the new bit rate. If the receiving endpoint wishes to accept the channel with the new bit rate, it must first assure that its call bandwidth is not exceeded by the change. If it is, the endpoint shall request a call bandwidth change (5 and 6) with its Gatekeeper. When the call bandwidth is sufficient to support the channel, the endpoint replies with an **openLogicalChannelAck** (7); otherwise, it responds with an **openLogicalChannelReject** indicating unacceptable bit rate.

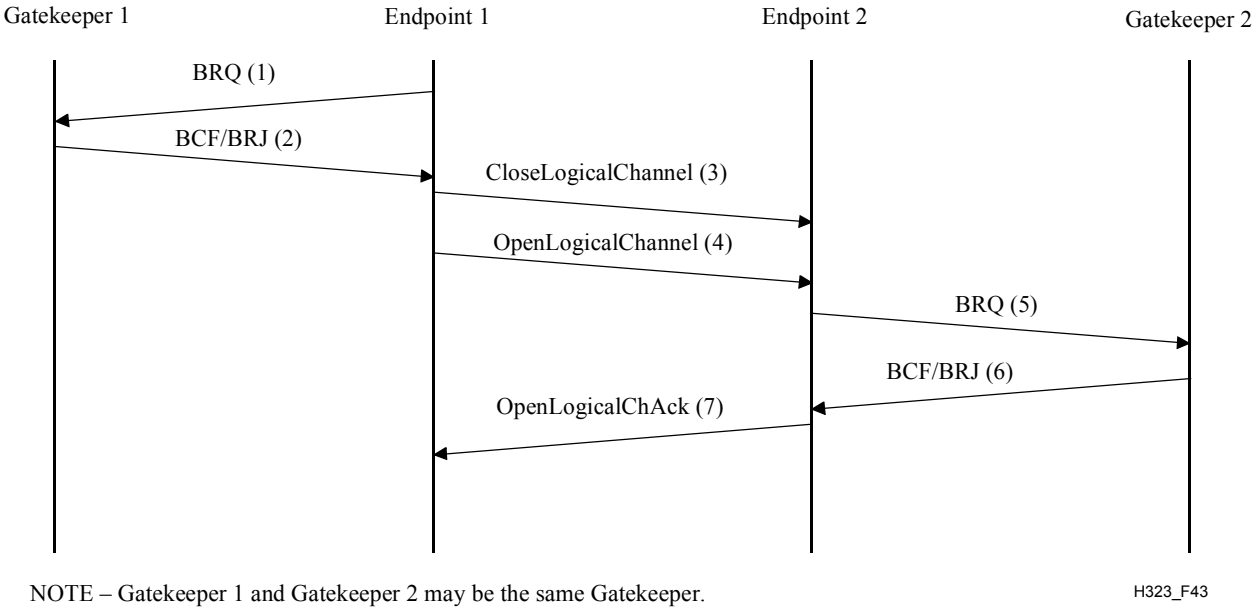
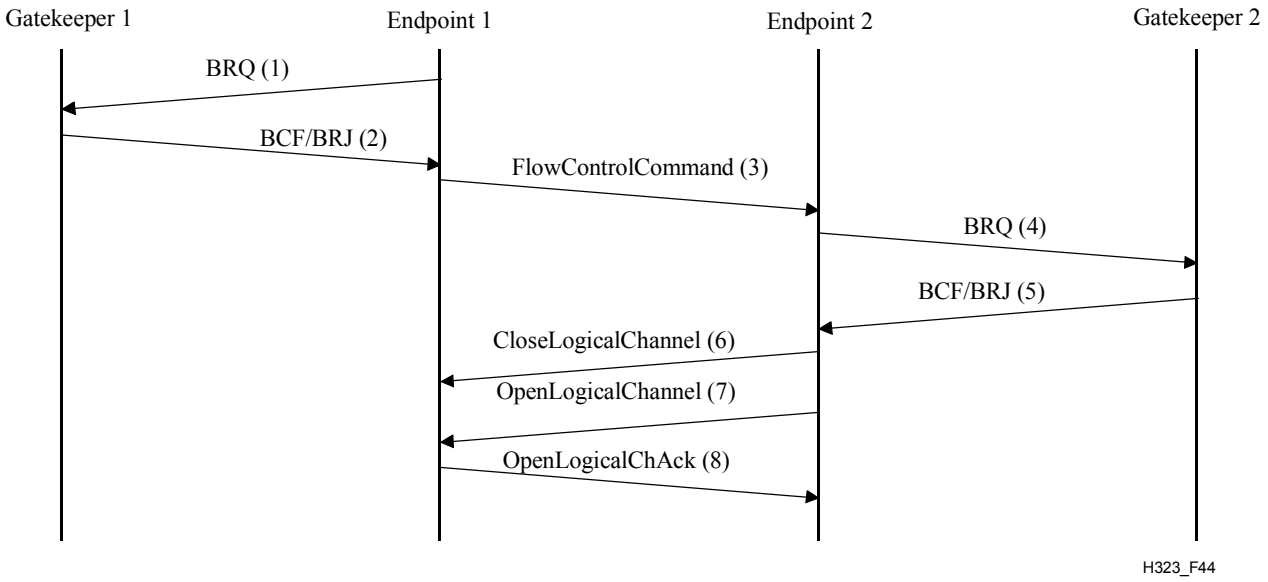


Figure 43/H.323 – Bandwidth change request – Transmitter change

If Endpoint 1 wishes to increase its transmitted bit rate on a logical channel from Endpoint 2, which it previously flow controlled to a lower bit rate, Endpoint 1 first determines if the call bandwidth will be exceeded. See Figure 44. If it will, Endpoint 1 shall request a bandwidth change from Gatekeeper 1. When the call bandwidth is sufficient to support the change, Endpoint 1 sends a **flowControlCommand** (3) to indicate the new upper limit on bit rate for the channel. If Endpoint 2 decides to increase the bit rate on the channel, it must first assure that its call bandwidth is not exceeded by the change. If it is, Endpoint 2 shall request a call bandwidth change (4 and 5) with its Gatekeeper. When the call bandwidth is sufficient to support the channel, Endpoint 2 will send the **closeLogicalChannel** (6) message to close the logical channel. It then reopens the logical channel using the **openLogicalChannel** (7) specifying the new bit rate. Endpoint 1 should then accept the channel with the new bit rate, and it replies with an **openLogicalChannelAck** (8).



NOTE – Gatekeeper 1 and Gatekeeper 2 may be the same Gatekeeper.

Figure 44/H.323 – Bandwidth change request – Receiver change

A Gatekeeper wishing to change the transmitted bit rate of Endpoint 1 sends a BRQ message to Endpoint 1. If the request is for a decrease in bit rate and the endpoint has the ability to support the requested bit rate, Endpoint 1 shall comply by reducing its aggregate bit rate and returning a BCF. If Endpoint 1 cannot support the requested bit rate, the endpoint may return a BRJ. Endpoint 1 may initiate the appropriate H.245 signalling to inform Endpoint 2 that bit rates have changed. This will allow Endpoint 2 to inform its Gatekeeper of the change. If the request is for an increase, the endpoint may increase its bit rate when desired and allowed by the Gatekeeper.

If the Gatekeeper wishes to increase the bandwidth used by the endpoint, the endpoint may return a BCF to indicate acceptance of the new higher bit rate or a BRJ to indicate that it rejects the additional bandwidth. The endpoint should only accept the higher bit rate if the endpoint is prepared to utilize the additional bandwidth.

The endpoint shall send a BRQ message to the Gatekeeper whenever bandwidth utilization decreases below that which was specified in the original ARQ or the last BRQ or BCF message. The endpoint shall also send a BRQ message to the Gatekeeper whenever logical channel signalling results in the addition or removal of a unique multicast stream to or from the endpoint.

Bandwidth information may be used by a Gatekeeper to better manage bandwidth usage on the network. It should be noted that precise bandwidth management requires the Gatekeeper to understand the network topology, which is outside the scope of this Recommendation. In addition, the bandwidth usage by the endpoint may actually be different than that which was reported due to the use of silence suppression, variable bit-rate codecs, or other factors. An endpoint shall not repeatedly send BRQ messages to its Gatekeeper when actual bandwidth utilization fluctuates due to these factors. Rather, the endpoint should request necessary bandwidth based on the set of open logical channels and should not consider periods of silence or other factors as a decrease in bandwidth.

8.4.2 Status

In order for the Gatekeeper to determine if an endpoint is turned off or has otherwise entered a failure mode, the Gatekeeper may use the Information Request (IRQ)/Information Request Response (IRR) message sequence (see ITU-T Rec. H.225.0) to poll the endpoints at an interval decided by the manufacturer. The Gatekeeper may request information for a single call or for all

active calls. Except when requesting additional IRR segments, the polling interval to request information for a particular call or all calls shall be greater than 10 s. However, the Gatekeeper may send IRQ messages that contain unique **callReferenceValue** values without regard to the polling period. This message may also be used by a diagnostic device as described in 11.2.

When an endpoint transmits an IRR message, it shall include the **perCallInfo** field in order to provide details about calls to the Gatekeeper. If the Gatekeeper requests status for all calls and no calls are active or for a single call that is no longer active or for which the endpoint has no information, the endpoint shall return an IRR message with the **invalidCall** field included and shall omit the **perCallInfo** field from the IRR.

If the Gatekeeper wants to receive call details for all of the active calls on an endpoint, it may send an IRQ message with the **callReferenceValue** field set to 0. The Gatekeeper should include the **segmentedResponseSupported** field to allow requests for all calls to be segmented if necessary. If the **segmentedResponseSupported** field is included, the endpoint shall return all or part of the call information in the **perCallInfo** field in a single IRR message. If segmentation is not allowed, but not all call details can be included in the IRR message, the endpoint shall include the **incomplete** field in the IRR message. If segmentation is allowed, the endpoint may return one or multiple IRR messages in response to the IRQ message. If one IRR message containing all call detail information is returned, the **irrStatus** element shall not be present. If the response is segmented into multiple IRR messages, the endpoint shall send the first IRR message and include the **segment** field. If the Gatekeeper wishes to receive the next segment, it shall transmit another IRQ message that includes the **segmentedResponseSupported** field, has the **callReferenceValue** set to 0, and has the **nextSegmentRequested** field set to the value of the next segment that the Gatekeeper expects to receive. If the Gatekeeper wishes to receive additional segments, it shall send the next IRQ message within 5 seconds after receiving the previous IRR message. If the endpoint receives a request for additional segments after 5 seconds (plus locally determined appropriate time for network delay), it may return an IRR message with the **incomplete** field included. When receiving an IRQ message from the Gatekeeper requesting the next segment within the allotted time, the endpoint shall transmit the next IRR message containing the next segment of call information. Note that if an IRR message is lost, the Gatekeeper may retransmit a request for the previously transmitted segment. Therefore, the endpoint shall be prepared to transmit the previous or next segment. If no additional segments are available or when the endpoint transmits the last segment of a series of IRR messages, the endpoint shall return an IRR message that includes the **complete** field. The Gatekeeper shall not transmit a different IRQ message to the endpoint requesting all call detail information until the last segment of information is transmitted or until the 10-second polling period has elapsed.

NOTE 1 – Since calls may begin or end after sending the first IRR message segment in response to an IRQ message requesting call details for all calls, the endpoint may or may not choose to include such calls when sending subsequent IRR message segments. The decision to report such calls when sending subsequent IRR segments is left to the manufacturer.

NOTE 2 – In order to improve performance and achieve better scalability, a Gatekeeper should limit the frequency at which it requests call details for all calls. Requesting call details for all calls is beneficial when an endpoint initially registers with the Gatekeeper, for example. However, repeatedly requesting such information – especially from very large-scale Gateways or MCUs – may lead to unacceptable performance degradation.

The Gatekeeper may want an endpoint to periodically send an unsolicited IRR message. The Gatekeeper may indicate this to the endpoint by specifying the rate that this IRR is sent within the **irrFrequency** field of the Admission Confirm (ACF) message. An endpoint receiving this **irrFrequency** rate shall send an IRR message at that rate for the duration of the call. While this rate is in effect, the Gatekeeper may still send IRQ messages to the endpoint which shall respond as described above.

An endpoint may want some of the unsolicited IRRs to be delivered reliably. The Gatekeeper can enable this by using the **willRespondToIRR** field in the RCF or ACF that it can acknowledge unsolicited IRRs. In this case, the endpoint may explicitly request the Gatekeeper to send an acknowledgment for the IRR. The Gatekeeper shall respond to such an IRR message by sending either an acknowledgment (IACK) or a negative acknowledgment (INAK). If the Gatekeeper did not announce that it will acknowledge IRRs, or if the endpoint did not request such an acknowledgment, no response shall follow the IRR.

During the duration of a call, an endpoint or Gatekeeper may periodically request call status from another endpoint. The requesting endpoint or Gatekeeper issues a Status Enquiry message. The Endpoint receiving the Status Enquiry message shall respond with a Status message indicating the current call state. This procedure may be used by the Gatekeeper in order to periodically check if a call is still active. Endpoints shall be able to accept any valid state values received in the Status message, including those which it may not be capable of entering. Note that this is an H.225.0 message sent on the Call Signalling Channel and should not be confused with IRR which is a RAS message sent on the RAS Channel.

The Gatekeeper may want to receive copies of certain H.225.0 call signalling PDUs when they are received or sent by an endpoint. An endpoint indicates its capability to send these PDUs by setting the **willSupplyUUIEs** in the ARQ or RRQ message sent to the Gatekeeper. The Gatekeeper indicates the list of PDU types it wishes to receive copies of, in the **uuiesRequested** field in the ACF or RCF. It also indicates if it wants copies when the PDUs are sent or received. An endpoint indicating this capability and receiving this list, shall send an IRR to the Gatekeeper each time it receives/sends the type of PDU requested.

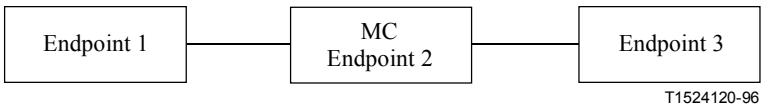
8.4.3 Ad hoc conference expansion

The following procedures are optional for terminals and Gateways and mandatory for MCs.

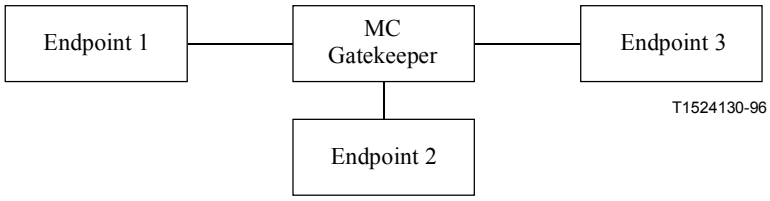
When a user places a call, the intent of the call is often not known to the calling endpoint. The user may wish to simply create a conference for itself and the called endpoint, the user may wish to join some conference at the called entity, or the user may wish to get a list of conferences that the called entity can provide. Using the procedures of this clause the conferences can be expanded from point-to-point calls into Ad Hoc Multipoint conferences.

An Ad Hoc Multipoint conference is one that can be expanded from a point-to-point conference involving an MC to a multipoint conference. First, a point-to-point conference is created between two endpoints (Endpoint 1 and Endpoint 2). At least one endpoint or the Gatekeeper must contain an MC. Once the point-to-point conference has been created, the conference may be expanded to multipoint conference in two different ways. The first way is when any endpoint in the conference invites another endpoint (Endpoint 3) into the conference by calling that endpoint through the MC. The second way is for an endpoint (Endpoint 3) to join an existing conference by calling an endpoint in the conference.

Ad Hoc Conference expansion can take place when using either the direct call signalling model or the Gatekeeper routed call signalling model. The H.245 Control Channel topology for the direct call signalling model appears as:



The H.245 Control Channel topology for the Gatekeeper routed call signalling model appears as:



In either case an MC must be present in the conference at the time of expansion to any number greater than 2 endpoints. Note that in the Gatekeeper routed model, the MC may be located in the Gatekeeper and/or one of the endpoints.

The procedures required to create a point-to-point conference and then expand the conference through invite and join, for each call model, is covered in the following subclauses. Procedures for the calling endpoint to discover a list of conferences that the called entity can provide are also covered.

It should be noted that the call is ended by a failure of the entity that is providing the MC.

8.4.3.1 Direct endpoint call signalling – conference create

Endpoint 1 creates a conference with Endpoint 2 as follows:

A1) Endpoint 1 sends a Setup message to Endpoint 2 containing a globally unique CID = N and **conferenceGoal** = **create** according to the procedure in 8.1.

A2) Endpoint 2 has the following options:

- A2a) If it wants to join the conference, it sends a Connect message with CID = N to Endpoint 1. In this case it is either:
  1) not participating in another conference; or
  2) it is participating in another conference, it is capable of participating in multiple conferences at the same time, and the received CID = N does not match the CID of any of the conferences in which it is currently participating.
- A2b) If it is in another conference with CID = M and can participate in only one conference at a time it either:
  1) rejects the call by sending Release Complete indicating in-conference; or
  2) it can request Endpoint 1 to join the conference with CID = M by sending a Facility message indicating **routeCallToMC** with the Call Signalling Channel Transport Address of the endpoint containing the MC and CID = M of the conference. The handling of the Facility message by Endpoint 1 is described in 8.4.3.7.
- A2c) If it does not wish to join this conference, it rejects the call by sending Release Complete indicating that the destination is busy.
- A2d) If Endpoint 2 is an MC(U) that hosts multiple conferences and wishes to provide Endpoint 1 with a choice of conferences to join, it can send a Facility message indicating **conferenceListChoice** and a list of conferences that Endpoint 1 may choose from. The list of conferences is sent as part of the Facility-UUIE. For backward compatibility, with Version 1 endpoints, conference lists are only provided if the **protocolIdentifier** in Endpoint 1's Setup message indicates that it is Version 2 or above.

Upon receipt of this **conferenceListChoice** Facility message, Endpoint 1 may join a conference from the list of conferences by sending a new Setup message to the MC(U) on the Call Signalling Channel which contains the selected CID and which has **conferenceGoal** = **join**. If Endpoint 1 chooses not to join any of the listed conferences, it shall send a Release Complete message to the MC(U).

A3) If Endpoint 2 enters the conference, Endpoint 1 uses the Transport Address of the Control Channel provided in the Connect message to open the Control Channel with Endpoint 2.

A4) The H.245 messages are then exchanged as described below:

A4a) **terminalCapabilitySet** messages are exchanged between the endpoints to determine the version number of the H.245 used in order to parse the remaining received messages correctly.

A4b) Using H.245 master-slave determination procedure, it is determined that Endpoint 2 is the master. In the Gatekeeper-Routed model, the master could be in an MC collocated with the Gatekeeper. If the master has an MC, it becomes the Active MC. It may then send the **mcLocationIndication** to the other endpoint(s). The MC may be active in the conference now or when the user initiates the multipoint conference function, at the choice of the manufacturer.

A4c) The master may send the **terminalNumberAssign** message to the endpoints. The endpoints shall use the 8-bit terminal number and not use the 8-bit MCU number from the 16-bit number assigned as the low 8 bits of the SSRC field in the RTP header. These low 8 bits in SSRC then identify the streams from a particular endpoint.

A4d) Since the capabilities of the receiver are known from the **terminalCapabilitySet** message, the transmitter opens the logical channels. It shall send one **h2250MaximumSkewIndication** for each pair of audio and video transmitted.

8.4.3.2 Direct endpoint call signalling – conference invite

There are two cases of the conference invite. First, the endpoint which contains the Active MC wishes to invite another endpoint into the conference. Second, an endpoint which does not contain the Active MC wishes to invite another endpoint into the conference.

1) After a point-to-point conference has been established using procedures A1) to A4) in 8.4.3.1, an endpoint (Endpoint 2) containing the Active MC wishing to add another endpoint to the conference shall use the following procedure:

B1) Endpoint 2 sends a Setup message to Endpoint 3 with CID = N and **conferenceGoal** = **invite** according to the procedures in 8.1. See Figure 45.

B2) Endpoint 3 has the following options:

B2a) If it wishes to accept the invitation to join the conference, it sends a Connect message with CID = N to Endpoint 2.

B2b) If it wishes to reject the invitation to join the conference, it sends a Release Complete message to Endpoint 2 indicating that the destination is busy.

B2c) If it is in another conference with CID = M, it can request Endpoint 2 to join the conference with CID = M by sending a Facility message indicating **routeCallToMC** with the Call Signalling Channel Transport Address of the endpoint containing the MC and CID = M of the conference. The handling of the Facility message by Endpoint 2 is described in 8.4.3.7.

B2d) If the received CID matches the CID of a conference that Endpoint 3 is currently participating in, it shall reject the call by sending Release Complete indicating that it is already in the conference.

B3) If Endpoint 3 accepts the invitation, Endpoint 2 uses the Transport Address of the Control Channel provided in the Connect message to open the Control Channel with Endpoint 3.

B4) The H.245 messages are then exchanged as described below:

C1) **terminalCapabilitySet** messages are exchanged between the MC and Endpoint 3.

C2) Using H.245 master-slave determination procedure, it is determined that Endpoint 2 is already the Active MC. The MC may then send the **mcLocationIndication** to the Endpoint 3.

C3) The MC shall send **multipointConference** at this time to all the three endpoints.

C4) The MC may send the **terminalNumberAssign** message to Endpoint 3. If received, the endpoints shall use the 8-bit terminal number and not use the 8-bit MCU number from the 16-bit number assigned as the low 8 bits of the SSRC field in the RTP header. These low 8 bits in SSRC then identify the streams from a particular endpoint.

C5) An endpoint can get the list of the other endpoints in the conference by sending the **terminalListRequest** message to the MC. The MC responds with the **terminalListResponse**.

C6) Whenever a new endpoint joins the conference, the MC sends the **terminalNumberAssign** message to Endpoint 4 and **terminalJoinedConference** message to Endpoints 1, 2 and 3.

C7) Whenever an endpoint leaves the conference, the MC sends **terminalLeftConference** to the remaining endpoints.

C8) The MC shall send the **communicationModeCommand** to all the endpoints in the conference.

C9) Endpoint 1 and Endpoint 2 will close their logical channels that were created during the point-to-point conference if they are inconsistent with the information contained in the **communicationModeCommand**.

C10) The logical channels can now be opened between the MC and the endpoints.

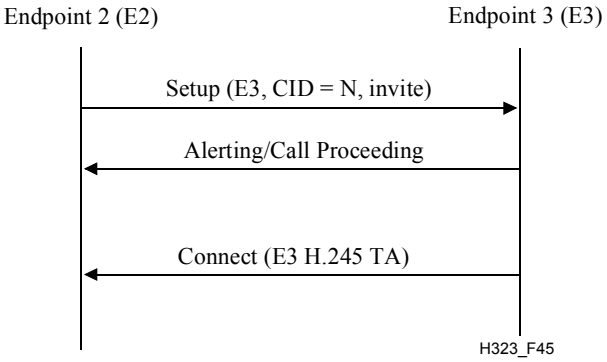


Figure 45/H.323 – MC invite signalling

2) After a point-to-point conference has been established using procedures A1) to A4) in 8.4.3.1, an endpoint (Endpoint 1) that does not contain the Active MC wishing to add another endpoint to the conference shall use the following procedure:

B1) Endpoint 1 sends a Setup message to the MC (Endpoint 2) with a new CRV indicating a call to Endpoint 3 by providing the Transport Address of Endpoint 3, CID = N, and **conferenceGoal** = **invite**. See Figure 46.

B2) Endpoint 2 sends a Setup message to Endpoint 3 with CID = N and **conferenceGoal = invite** according to the procedures in 8.1.

B3) During call signalling with Endpoint 3, Endpoint 2 shall pass Call Signalling messages received from Endpoint 3, including Connect, to Endpoint 1 (the original inviter).

B4) Endpoint 3 has the same options, described previously, of either accepting or rejecting the invitation.

B5) At some time after the completion of the call setup procedure between Endpoint 2 and Endpoint 3, Endpoint 2 shall send a Release Complete message to Endpoint 1.

B6) If Endpoint 3 accepts the invitation, Endpoint 2 uses the Transport Address of the Control Channel provided in the Connect message to open the Control Channel with Endpoint 3.

B7) The H.245 messages are then exchanged as previously described in procedures C1) to C10).

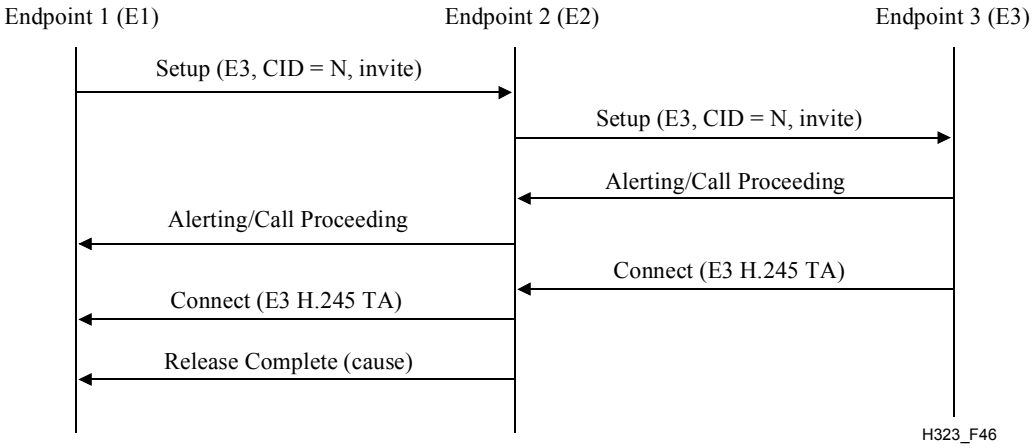


Figure 46/H.323 – Non-MC invite signalling

8.4.3.3 Direct endpoint call signalling – conference join

There are two cases of the conference join. First, an endpoint calls the endpoint which contains the Active MC. Second, an endpoint calls an endpoint which is not the Active MC.

After a point-to-point conference has been established using procedures A1) to A4) in 8.4.3.1, an endpoint (Endpoint 3) wishing to join a conference may attempt to connect with the endpoint containing the Active MC in the conference. In this case, the following procedure shall be used:

B1) Endpoint 3 sends a Setup message to Endpoint 2 with CID = N and **conferenceGoal = join** according to the procedures in 8.1. See Figure 47.

B2) If the CID matches the CID of an active conference in the MC, Endpoint 2 (MC) has the following options:

B2a) If it decides that Endpoint 3 should be allowed to join the conference, it sends the Connect message with CID = N.

B2b) If it decides that Endpoint 3 should not be allowed to join the conference, it sends the Release Complete message indicating that the destination is busy.

B3) If the CID does not match the CID of an active conference in the MC, Endpoint 2 shall send Release Complete indicating a bad CID.

B4) If Endpoint 2 allows the join, Endpoint 2 opens the Control Channel with Endpoint 3.

B5) The H.245 messages are then exchanged as previously described in procedures C1) to C10).

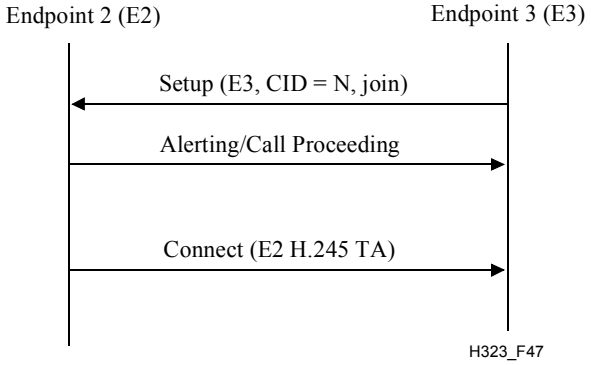


Figure 47/H.323 – MC join signalling

After a point-to-point conference has been established using procedures A1) to A4), an endpoint (Endpoint 3) wishing to join a conference may attempt to connect with an endpoint that does not contain the Active MC in the conference. In this case, the following procedure shall be used:

B1) Endpoint 3 sends a Setup message to Endpoint 1 with CID = N and **conferenceGoal** = **join** according to the procedures in 8.1. See Figure 48.

B2) Endpoint 1 returns a Facility message indicating **routeCallToMC** with the Call Signalling Channel Transport Address of Endpoint 2 (containing the Active MC) and the CID = N of the conference.

B3) Endpoint 3 then sends a Setup message to Endpoint 2 (MC) with CID = N and **conferenceGoal** = **join** as described in the previous conference join procedure.

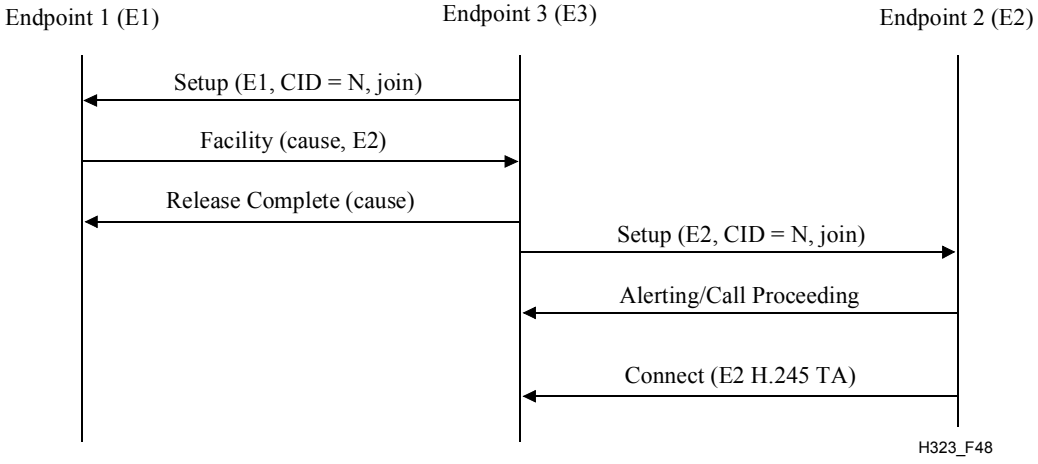


Figure 48/H.323 – Non-MC join signalling

8.4.3.4 Gatekeeper routed call signalling – conference create

In cases where the Gatekeeper routes the Call Signalling Channel and the H.245 Control Channel, the Gatekeeper may contain (or have access to) an MC or MCU. Procedures A1) to A4) are used to establish the point-to-point call.

If the MC(U) hosts multiple conferences and wishes to provide Endpoint 1 with a choice of conferences to join, it can send a Facility message indicating **conferenceListChoice** and a list of conferences that Endpoint 1 may choose from. The list of conferences is sent as part of the Facility-UUIE. For backward compatibility, with Version 1 endpoints, conference lists are only

provided if the **protocolIdentifier** in Endpoint 1's Setup message indicates that it is Version 2 or above.

Upon receipt of this **conferenceListChoice** Facility message, Endpoint 1 may join a conference from the list of conferences by sending a new Setup message to the MC(U) on the Call Signalling Channel which contains the selected CID and which has **conferenceGoal** = **join**. If Endpoint 1 chooses not to join any of the listed conferences, it shall send a Release Complete message to the MC(U).

During master-slave determination [A4b], if the Gatekeeper's **terminalType** is greater than the **terminalType** received in the **masterSlaveDetermination** message, the Gatekeeper may attempt to become master for the call. In this case, the Gatekeeper shall immediately send a **masterSlaveDeterminationAck** message to the source of the Master-Slave Determination message indicating that it is a slave, and the Gatekeeper performs Master-Slave Determination with the destination entity as defined in 6.2.8.4. If the Gatekeeper wins that Master-Slave Determination, the MC associated with the Gatekeeper shall be the Active MC. If the Gatekeepers **terminalType** is not greater than the **terminalType** of the endpoint or the Gatekeeper decides not to replace the endpoint's **terminalType** with its own, the Gatekeeper shall not modify the **terminalType** value and it shall transparently relay all messages of that Master-Slave Determination procedure.

8.4.3.5 Gatekeeper routed call signalling – conference invite

After a point-to-point conference has been established using procedures A1) to A4) as modified above, an endpoint (Endpoint 1 or 2) that does not contain the Active MC wishing to add another endpoint to the conference shall use the following procedure:

B1) Endpoint 1 sends a Setup message through the Gatekeeper directed to Endpoint 3 with a new CRV, CID = N and **conferenceGoal** = **invite**. See Figure 49.

B2) The Gatekeeper (MC) sends a Setup message to Endpoint 3 with CID = N and **conferenceGoal** = **invite** according to the procedures in 8.1.

B3) During call signalling with Endpoint 3, the Gatekeeper shall pass Call Signalling messages received from Endpoint 3, including Connect, to Endpoint 1 (the original inviter).

B4) Endpoint 3 has the same options, described previously, of either accepting or rejecting the invitation.

B5) At some time after the completion of the call setup procedure between the Gatekeeper and Endpoint 3, the Gatekeeper shall send a Release Complete message to Endpoint 1.

B6) If Endpoint 3 accepts the invitation, the Gatekeeper uses the Transport Address of the Control Channel provided in the Connect message to open the Control Channel with Endpoint 3.

B7) The H.245 messages are then exchanged as previously described in procedures C1) to C10) with the Gatekeeper taking part in all master-slave determination procedures as the Active MC (C2). At this time, the Control Channels from the endpoints should be connected to the MC, and the MC should be in control of the conference.

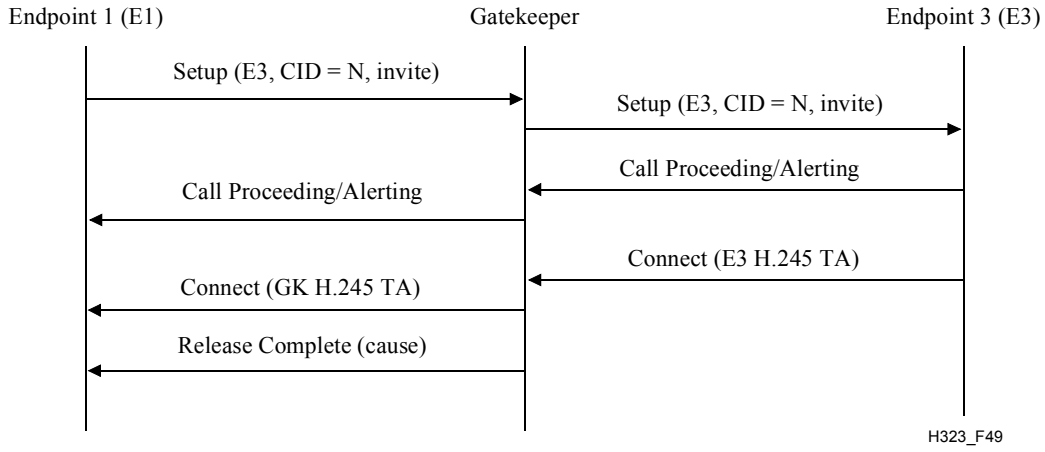


Figure 49/H.323 – Gatekeeper routed invite signalling

8.4.3.6 Gatekeeper routed call model – conference join

After a point-to-point conference has been established using procedures A1) to A4) as modified above, an endpoint (Endpoint 3), wishing to join a conference may attempt to connect with an endpoint that does not contain the Active MC in the conference. In this case, the following procedure shall be used:

B1) Endpoint 3 sends a Setup message through the Gatekeeper directed to Endpoint 1 with CID = N and **conferenceGoal = join** according to the procedures in 8.1. See Figure 50.

B2) If the CID matches the CID of an active conference in the MC, the Gatekeeper (MC) has the following options:

- B2a) If it decides that Endpoint 3 should be allowed to join the conference, it sends the Connect message with CID = N to Endpoint 3.
- B2b) If it decides that Endpoint 3 should not be allowed to join the conference, it sends the Release Complete message indicating that the destination is busy.
- B2c) The Gatekeeper may forward the Setup message to Endpoint 1. Endpoint 1 may respond with a Facility message indicating **routeCallToMC** or it may respond with a Release Complete.

B3) If the CID does not match the CID of an active conference in the MC, the Gatekeeper shall send Release Complete indicating a bad CID.

B4) If the Gatekeeper allows the join, the Gatekeeper uses the Transport Address of the Control Channel provided in the Setup message to open the Control Channel with Endpoint 3.

B5) The H.245 messages are then exchanged as previously described in procedures C1) to C10) with the Gatekeeper taking part in all master-slave determination procedures as the Active MC (C2). At this time, the Control Channels from the endpoints should be connected to the MC, and the MC should be in control of the conference.

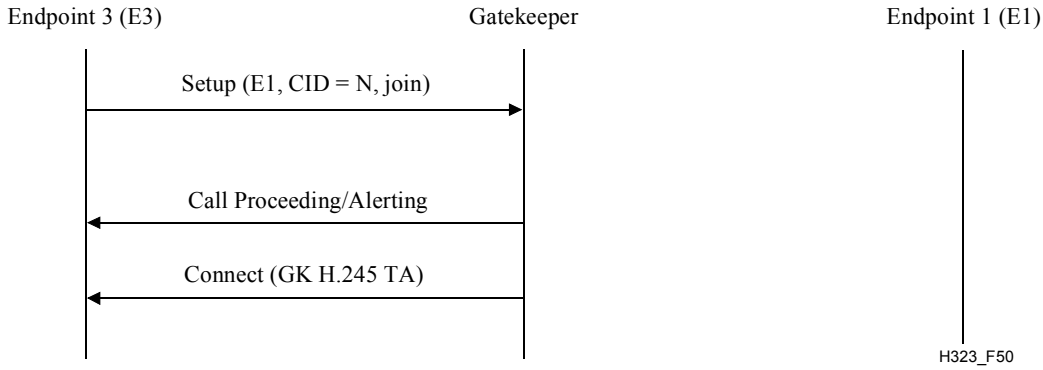


Figure 50/H.323 – Gatekeeper routed join signalling

8.4.3.7 Handling of the facility message

Upon receiving a Facility message indicating **routeCallToMC** with the Call Signalling Channel Transport Address of the endpoint containing the MC and CID of a conference, an endpoint may release the current call and attempt to join the indicated conference according to the procedures in 8.4.3.3 or in 8.4.3.6.

An endpoint may receive such a Facility message either as a direct reply to its Setup message or during the active phase of a call.

8.4.3.8 Conference out of consultation

This clause defines the procedures for an endpoint (endpoint A) requesting an ad-hoc conference with two or more other endpoints (remote endpoints B, C, etc.) with which endpoint A already has active calls. This typically applies – but is not limited to – ad-hoc conference being requested out of a consultation condition.

NOTE 1 – "Consultation condition" refers to a situation where endpoint A has an active call with endpoint C (consultation call) while having one or more endpoints on hold, i.e., held call(s). An endpoint may be put on hold by using the procedures of ITU-T Rec. H.450.4 [36], 8.4.6, or by local procedures.

Endpoint A has the capability of "merging" the independent calls to multiple endpoints into a single conference either at the endpoint A (as described in scenario 1 below) or by forming the conference on a separate MCU (as described in scenario 2 below).

NOTE 2 – Procedures in this clause relate only to the calls at an endpoint that are to be joined into a conference out of consultation. An endpoint may have additional calls that do not participate in the conference and to which this clause will not apply.

8.4.3.8.1 Scenario 1: Conference provided by endpoint

If endpoint A has the capabilities, it may "merge" the held call and the consulted call into a conference resulting in a three-way conversation between A, B and C. For this scenario endpoint A must have an MC. Both the centralized and the decentralized conferencing models are possible. If the centralized model is to be used (i.e., if the terminal provides the media mixing/switching), endpoint A shall have an MP.

An endpoint with MC and MP is actually an MCU and should use **terminalType** 170, 180 or 190 as appropriate for master-slave determination.

The following scenarios are possible:

1a) If endpoint A is the master of both calls to B and C, it may simply retrieve the held call into the conference with C and declare itself as the Active MC on both calls through master-slave negotiation.

1b) If endpoint A is a slave on one or more of the calls but no call on which it is the slave has an Active MC, endpoint A should reinitiate master-slave determination on all calls in which it is slave, using the **terminalType** 240, as specified in Table 1 for an Active MC. If it ends this procedure as master on all calls, it should act as in 1a) above; if it is slave in one or more calls, endpoint A should act as directed in 1c) below.

1c) If one or more of the calls in which endpoint A is participating is already a call in which endpoint A is not the Active MC, procedures for cascading MCUs shall be followed.

Once a conference is established within endpoint A, a further endpoint D – that is being consulted by endpoint A – may be invited into the existing conference as described in 8.4.3.2 and 8.4.3.5.

8.4.3.8.2 Scenario 2: Conference provided by MCU

If endpoint A has access to an MCU, the following procedure may be used to accomplish conference out of consultation:

2a) Endpoint A establishes a new call to the MCU using a Setup message with **conferenceGoal = create** and CID = N.

2b) Endpoint A drops its call with endpoint C using a Release Complete message with **reason** set to **replaceWithConferenceInvite** including argument CID = N.

2c) Endpoint A sends a Setup message to the MCU with **conferenceGoal = invite**, CID = N, and sufficient information for the MCU to make a call to endpoint C (see also 8.4.3.2).

2d) Steps 2b) and 2c) shall be repeated with "endpoint C" replaced by "endpoint B". Note that there is no requirement to retrieve the call to B from hold before inviting it to the conference.

2e) For exchange of H.245 conference related messages refer to 8.4.3.2 of H.323 steps C1)-C10).

Alternative mechanisms to steps 2b), 2c) and 2d) are:

1) H.450.2 [35] Call Transfer (with endpoint A acting as "transferring" endpoint, endpoints B and C acting as "transferred" endpoints and the MC/MCU acting as the "transferred-to" endpoint. The Facility message containing **callTransferInitiate Invoke APDU** shall also contain element CID = N.

2) H.225.0 "Facility re-route to MC" mechanism (sending an H.225.0 Facility message to endpoints B and C containing CID = N, **facilityReason = routeCallToMC** and the address of the MCU) if H.450.2 is not supported.

These alternative mechanisms are recommended if the remote endpoint is located within the SCN.

An endpoint (e.g., endpoint A) may split from the conference (e.g., by putting its call to the MCU on hold). Endpoint A may then consult with a further endpoint D that may subsequently be invited to the existing conference by using the procedures as described in 2b) and 2c) above with "endpoint C" replaced by "endpoint D". Alternative mechanisms as described above by means of using H.450.2 Call Transfer or H.225.0 "Facility re-route to MC" may be used instead.

8.4.4 Supplementary services

Support for Supplementary Services is optional. The H.450.x series of Recommendations describes a method of providing Supplementary Services in the H.323 environment.

8.4.5 Multipoint cascading

In order to cascade MCs, a call must be established between the entities containing the MCs. This call is established according to the procedures defined in 8.1 and 8.4.3. Once the call is established and the H.245 Control Channel is opened, the Active MC (determines according to the master/slave procedures in 6.2.8.4) may activate the MC in a connected entity. This is done by using the H.245 **remoteMC** message. The following results shall occur in response to the **remoteMC** message:

Calling entity	Called entity	Conference goal	RemoteMC Sender	RemoteMC Selection	Result
Active MC	Inactive MC	**create**	Calling entity	**masterActivate**	Called MC accepts request and becomes the master MC
Active MC	Inactive MC	**invite**	Calling entity	**slaveActivate**	Called MC accepts request and becomes a slave MC
Active MC	Inactive MC	**join**	N/A	N/A	Not allowed
Inactive MC	Active MC	**create**	N/A	N/A	Not allowed
Inactive MC	Active MC	**invite**	N/A	N/A	Not allowed
Inactive MC	Active MC	**join**	Called entity	**slaveActivate**	Calling MC accepts request and becomes a slave MC

Once the cascaded conference is established, either the master or slave MCs may invite other endpoints into the conference. There shall only be one master MC in a conference. A slave MC shall only be cascaded to a master MC. Slave MCs shall not be cascaded to other slave MCs. This allows only dumb-bell or star cascaded configurations.

The slave MC shall identify the cascaded conference using the CID established by the master when the conference was created.

The slave MC shall accept and act upon **communicationsModeCommand** messages from the master MC. The slave MC shall forward these messages to its locally connected endpoints. The slave MC may receive **requestMode** messages from its locally connected endpoints. It should forward these to the master MC. The slave MC shall not send **communicationsModeCommand** messages to the master MC.

The master MC should follow the procedures in 8.4.3.2, C3) through C10), in order to establish a common operating mode with the slave MC. Based on this information, each MC is responsible for opening logical channels for media distribution between its locally connected endpoints and endpoints designated by the master MC.

In addition to inviting new endpoints into the conference, an MC which supports multiple conferences may directly move endpoints into another conference without tearing down the existing connection. If this is done, the MC should send the **substituteCID** message to these endpoints. Endpoints which receive a **substituteCID** message during a call shall continue to use the conference ID (CID) used in the previous RAS messages (e.g., ARQ, BRQ, etc.) when conversing with its Gatekeeper for the duration of that particular call.

Terminal numbering and chair control functions may follow the procedures defined in ITU-T Rec. H.243. The use of T.120 for controlling MC cascading is for further study. The use of T.120 in cascaded connections is described in the T.120 series of Recommendations.

When a master sends a **remoteMC** Request with the selection **deActivate**, the slave MC should remove all endpoints from the conference.

8.4.6 Third party initiated pause and re-routing

For the purpose of this clause, an empty capability set is defined as a **terminalCapabilitySet** message that contains only a sequence number and a protocol identifier.

To allow Gatekeepers to re-route connections from endpoints that do not support supplementary services, endpoints shall respond to the reception of an empty capability set as defined in this clause. This feature allows "network" elements such as PBXs, call centers, and IVR systems to re-route connections independently of supplementary services and facilitates pre-connect announcements. It can also be used to delay H.245 media establishment when features such as Gatekeeper based user location are being used. It is also highly recommended that Version 1 endpoints support this feature.

On reception of an empty capability set, an endpoint shall enter a "transmitter side paused" state. On entering this state, the endpoint shall stop transmitting on established logical channels and shall close all logical channels that it previously opened, including bidirectional logical channels. It shall close these channels in the usual way by sending the **closeLogicalChannel** message. The endpoint shall not request the remote endpoint to close logical channels, either unidirectional or bidirectional, that the remote endpoint opened. The endpoint shall send the **terminalCapabilitySetAck** message in the usual way: the message may be sent before stopping transmission and so shall not be interpreted as an indication that transmission has stopped.

While in the "transmitter side paused" state, an endpoint shall not initiate the opening of any logical channels, but shall accept the opening and closing of logical channels from the remote end based on the usual rules and shall continue to receive media on open logical channels opened by the remote endpoint. This allows endpoints to receive announcements (e.g., pre-connect call progress) where the announcing entity does not wish to receive media from the endpoint. A **terminalCapabilitySet** message may be sent whenever an endpoint's capabilities change, including when the endpoint is in the "transmitter side paused" state. This allows communication to be established between two endpoints that initially do not declare any capabilities.

An endpoint in "transmitter side paused" state may also put the other endpoint in the call into a "transmitter side paused" state by transmitting an empty capability set message. Upon reception of the empty capability set message, the receiver shall adhere to the procedures defined in this clause.

An endpoint shall leave the "transmitter side paused" state on reception of any **terminalCapabilitySet** message, other than an empty capability set. On leaving this state, an endpoint shall reset its H.245 state to that which it was in just after the H.245 transport connection was made at call establishment time (i.e., the beginning of phase B), but shall preserve state information relating to any logical channels that are open. This puts the endpoint in a known H.245 state after the pause. This allows an endpoint to be connected to a different endpoint when it is released from the paused state.

After leaving the "transmitter side paused" state, an endpoint shall proceed with normal H.245 procedures: it shall take part in master/slave determination signalling and may proceed with normal open logical channel signalling procedures. When an MC leaves the "transmitter side paused" state, it shall act as if a new endpoint has entered the conference.

If an endpoint in a "transmitter side paused" state had also transmitted an empty capability set in order to put the other end in "transmitter side paused" state, it shall assume that it is still in a paused state until it receives a non-empty capability set from the other side when it releases the other endpoint from the paused state. The paused endpoint shall be prepared to receive OLCs from the other endpoint.

Unless its capabilities have changed, an endpoint need not resend a capability set as the Gatekeeper will have supplied this to the remote endpoint to remove any paused state in the remote endpoint. This option of not sending a capability set enables faster reconnection. If the first

terminalCapabilitySet message sent by an endpoint after leaving the "transmitter side paused" state differs from the capability set that the Gatekeeper provided to the remote endpoint, the Gatekeeper shall signal the remote endpoint to remove capabilities which were not indicated by the initiating endpoint.

NOTE 1 – An endpoint should take care with the capabilities it sends at this time. In particular, an endpoint shall send all capabilities it wants to advertise and not a small addition to previously signalled capabilities. In addition, if the endpoint has so many capabilities that it requires more than one **terminalCapabilitySet** to signal them, there may be a window of time when the gatekeeper has removed the capabilities described in second and subsequent **terminalCapabilitySet** messages.

NOTE 2 – A non-empty capability set shall not be sent to an endpoint until all of its transmit logical channels have been closed. A switching entity should also send an H.450 redirection indication Facility message if the endpoint is being re-routed.

8.5 Phase E – Call termination

Either endpoint or an intermediate call signalling entity may terminate a call. Call termination shall be accomplished by either Procedure A or Procedure B:

Procedure A:

A-1) It should discontinue transmission of video at the end of a complete picture, when applicable.

A-2) It should discontinue transmission of data, when applicable.

A-3) It should discontinue transmission of audio, when applicable.

A-4) It shall transmit a Release Complete message and close the H.225.0 call signalling channel and, if open separately, the H.245 Control Channel without sending any H.245 message. Note that closing the media channels is implied.

A-5) Endpoints shall clear the call by using the procedures defined in 8.5.1 or 8.5.2.

Procedure B:

B-1) It should discontinue transmission of video at the end of a complete picture and then close all logical channels for video, when applicable.

B-2) It should discontinue transmission of data and then close all logical channels for data, when applicable.

B-3) It should discontinue transmission of audio and then close all logical channels for audio, when applicable.

B-4) It shall transmit the H.245 **endSessionCommand** message in the H.245 Control Channel, indicating to the far end that it wishes to disconnect the call and then discontinue H.245 message transmission.

B-5) It shall then wait to receive the **endSessionCommand** message from the other endpoint and then shall close the H.245 Control Channel.

B-6) It shall transmit a Release Complete message and close the H.225.0 call signalling channel.

B-7) Endpoints shall clear the call by using the procedures defined in 8.5.1 or 8.5.2.

An endpoint receiving **endSessionCommand** without first having transmitted it shall carry out steps B-1) to B-7) above, except that in step B-5), it shall not wait for the **endSessionCommand** from the first endpoint.

Terminating a call may not terminate a conference; a conference may be explicitly terminated using an H.245 message (**dropConference**). In this case, the endpoints shall wait for the MC to terminate the calls as described above.

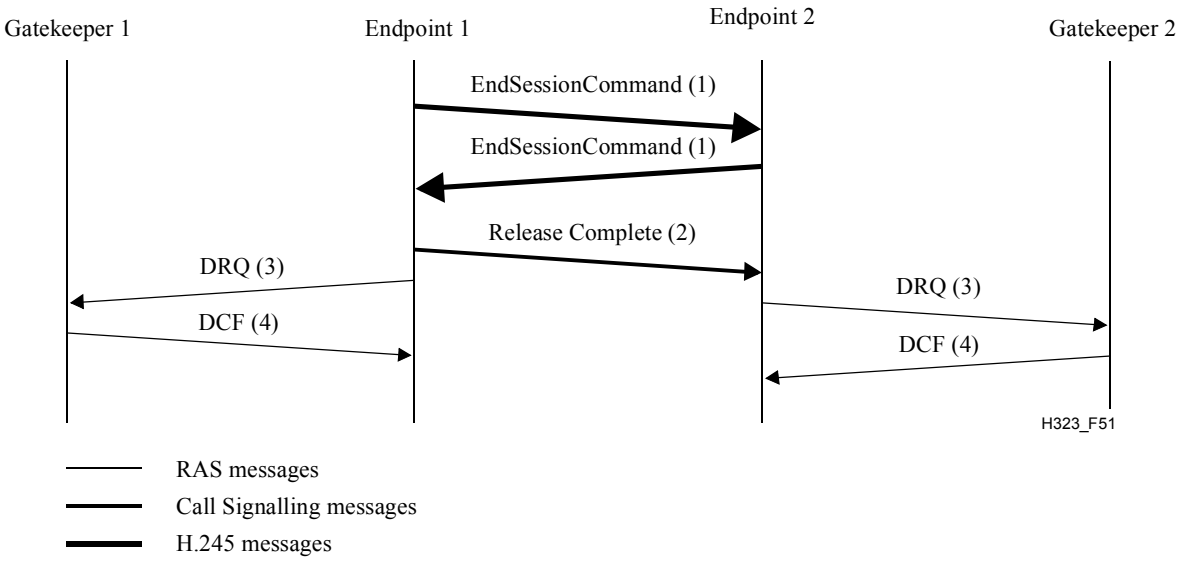
8.5.1 Call clearing without a gatekeeper

In networks that do not contain a Gatekeeper, after steps A-1) to A-5) or B-1) to B-6) above, the call is terminated. No further action is required.

8.5.2 Call clearing with a gatekeeper

In networks that contain a Gatekeeper, the Gatekeeper needs to know about the release of bandwidth. After performing steps A-1) to A-5 or B-1) to B-6) above, each endpoint shall transmit an H.225.0 Disengage Request (DRQ) message (3) to its Gatekeeper. The Gatekeeper shall respond with a Disengage Confirm (DCF) message (4). After sending the DRQ message, the endpoints shall not send further unsolicited IRR messages to the Gatekeeper. See Figure 51. At this point, the call is terminated. Figure 51 shows the direct call model; a similar procedure is followed for the Gatekeeper routed model.

The DRQ and DCF messages shall be sent on the RAS Channel.



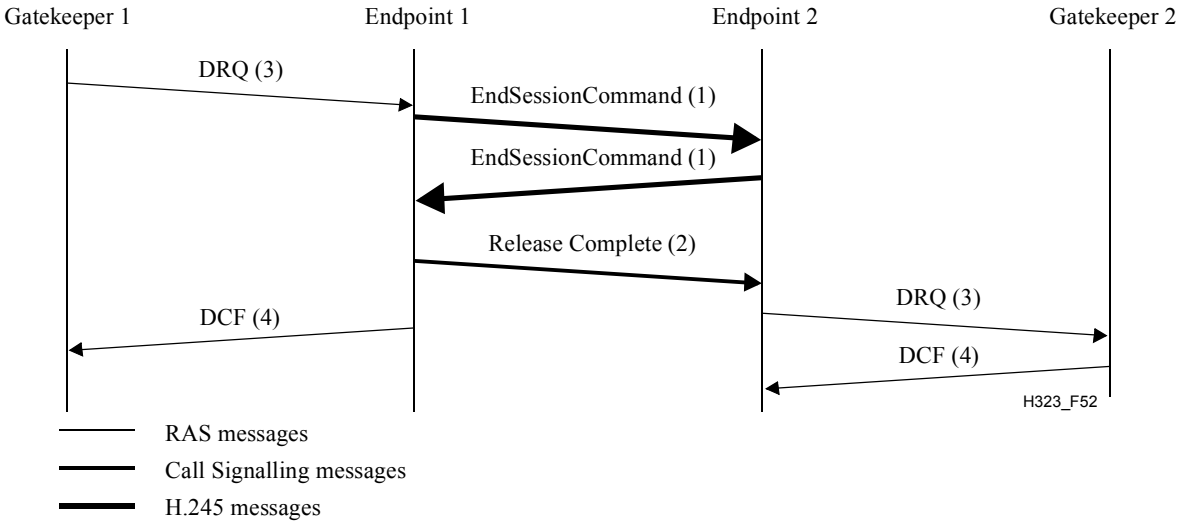
NOTE – Gatekeeper 1 and Gatekeeper 2 may be the same Gatekeeper.

Figure 51/H.323 – Endpoint initiated call clearing (Procedure B)

8.5.3 Call clearing by gatekeeper

The Gatekeeper may terminate call by sending a DRQ to an endpoint. See Figure 52. The endpoint shall immediately follow steps A-1) through A-5) or B-1) through B-6) from above and then reply to the Gatekeeper with DCF. The other endpoint, upon receiving **endSessionCommand**, shall follow the procedure described above. Figure 52 shows the direct call model; a similar procedure is followed for the Gatekeeper routed model.

If the conference is a multipoint conference, the Gatekeeper should send a DRQ to each endpoint in the conference, in order to close the entire conference.



NOTE – Gatekeeper 1 and Gatekeeper 2 may be the same Gatekeeper.

Figure 52/H.323 – Gatekeeper initiated call clearing

8.6 Protocol failure handling

The underlying reliable protocol of the H.245 Control Channel uses appropriate effort to deliver or receive data on the channel before reporting a protocol failure. Therefore, if a protocol failure is reported on the channel, the H.245 Control Channel, and all associated logical channels shall be closed. This shall be done following the procedures of Phase E, as if the other endpoint had issued the H.245 **endSessionCommand**. This includes transmission of the DRQ message to the Gatekeeper and termination of the Call Signalling Channel. In the case where the MC detects failure in a multipoint conference, the MC shall send **terminalLeftConference** messages to the remaining terminals. It is up to the implementation whether or not to try to re-establish the call without user intervention. In any case, this would appear to the other endpoint (and the Gatekeeper) as a new call.

The Call Signalling Channel also uses an underlying reliable protocol. Depending on the routing of the Call Signalling Channel, either the Gatekeeper or an endpoint may detect the protocol failure. If the Gatekeeper detects the failure, it shall attempt to re-establish the Call Control Channel. This implies that the endpoint shall always have the ability to establish a channel on its Call Signalling Channel Transport Address. Failure of the Call Signalling channel shall not change the call state. After re-establishment of the Call Signalling Channel, the Gatekeeper may send a Status message to request the call state of the endpoint to assure that they are in synchronization.

If the endpoint detects the failure, the endpoint may choose to terminate the call as described in Phase E, or it may attempt to re-establish the Call Signalling Channel as described above.

If, during a call, an endpoint wants to determine if the other endpoint is still functioning and connected, it may send the H.245 **roundTripDelayRequest**. Since H.245 Control Channel is carried on a reliable channel, this will result in a response from the other endpoint or an error from the transport interface. In the latter case, the procedures described above shall be used. An endpoint in a multipoint conference may use the same mechanism; however, it will learn only whether it still has a connection to the MC. Note that it is possible for an endpoint to have an error-free connection with the MC but still be receiving no audio or video from the rest of the terminals in the conference.

NOTE – The requirement to close the H.245 Control Channel and all associated logical channels does not apply to equipment that is capable of recovering the H.245 control channel.

9 Interoperation with other terminal types

Interoperation with other terminals shall be accomplished through the Gateway. See 6.3 and ITU-T Rec. H.246.

9.1 Speech-only terminals

Interoperation with speech-only terminals (telephony) over the ISDN or GSTN can be provided by:

1) using a H.323-ISDN speech Gateway;
2) using a H.323-GSTN speech Gateway.

The Gateway should consider the following issues:

– Audio code conversion:
  - ISDN: if desired, since ISDN uses G.711.
  - GSTN: from analogue to G.711.
– Bit stream conversion:
  - ISDN: H.225.0 to/from unframed.
  - GSTN: generate H.225.0.
– Control conversion (generate H.245).
– Call Control Signalling conversion.
– DTMF tone conversion to/from H.245 **userInputIndication** message and RTP payload types (as per 10.5).

9.2 Visual telephone terminals over the ISDN (ITU-T Rec. H.320)

Interoperation with visual telephone terminals over the ISDN (ITU-T Rec. H.320) can be provided by:

– using a H.323-H.320 Gateway.

The Gateway should consider the following issues:

– Video format conversion. (If desired, H.261 is mandatory for both terminal types.)
– Audio code conversion. (If desired, G.711 is mandatory for both terminal types.)
– Data protocol conversion.
– Bit stream conversion. (H.225.0 to/from H.221.)
– Control conversion. (H.245 to/from H.242.)
– Call Control Signalling conversion.
– SBE Number conversion to/from H.245 **userInputIndication** message and RTP payload types (as per 10.5).

9.3 Visual telephone terminals over GSTN (ITU-T Rec. H.324)

Interoperation with visual telephone terminals over the GSTN (ITU-T Rec. H.324) can be provided by two methods:

1) using a H.323-H.324 Gateway;
2) using a H.323-H.320 Gateway, assuming that there exists an H.320-H.324 Gateway in the circuit switched network.

The Gateway should consider the following issues:

– Video format conversion. (If desired, H.261 is mandatory for both terminal types.)
– Data protocol conversion.
– Audio code conversion. (G.711 is mandatory for H.323 terminal, G.723.1 is mandatory for H.324 terminal.)
– Bit stream conversion. (H.225.0 to/from H.223.)
– Call Control Signalling conversion.

9.4 Visual telephone terminals over mobile radio (ITU-T Rec. H.324/M – Annex C/H.324)

For further study.

9.5 Visual telephone terminals over ATM (H.321 and H.310 RAST)

Interoperation with visual telephone terminals over ATM networks (H.321 and H.310 RAST terminals operating in H.320/H.321 interworking mode) can be provided by two methods:

1) using a H.323-H.321 Gateway;
2) using a H.323-H.320 Gateway, assuming that there exists an I.580 ISDN/ATM Interworking Unit in the network.

The Gateway should consider the following issues:

– Video format conversion. (If desired, H.261 is mandatory for both terminal types.)
– Data protocol conversion.
– Audio code conversion. (If desired, G.711 is mandatory for both terminal types.)
– Bit stream conversion. (H.225.0 to/from H.221.)
– Control conversion. (H.245 to/from H.242.)
– Call Control Signalling conversion.

9.6 Visual telephone terminals over guaranteed quality of service LANs (ITU-T Rec. H.322)

Interoperation with visual telephone terminals over Guaranteed Quality of Service LANs (ITU-T Rec. H.322) can be provided by:

– using a H.323-H.320 Gateway, assuming that there exists a GQOS LAN-ISDN Gateway in the network.

The Gateway should consider the following issues:

– Video format conversion. (If desired, H.261 is mandatory for both terminal types.)
– Data protocol conversion.
– Audio code conversion. (If desired, G.711 is mandatory for both terminal types.)
– Bit stream conversion. (H.225.0 to/from H.221.)
– Control conversion. (H.245 to/from H.242.)
– Call Control Signalling conversion.

9.7 Simultaneous voice and data terminals over GSTN (ITU-T Rec. V.70)

Interoperation with Simultaneous Voice and Data Terminals over GSTN (ITU-T Rec. V.70) can be provided by:

– using a H.323-V.70 Gateway.

The Gateway should consider the following issues:

– Audio code conversion. (G.711 to/from Annex A/G.729.)
– Data protocol conversion.
– Bit stream conversion. (H.225.0 to/from V.76/V.75.)
– Control conversion. (Both terminals use H.245.)
– Call Control Signalling conversion.

9.8 T.120 terminals on the packet based network

An H.323 terminal that has T.120 capability should be capable of being configured as a T.120-only terminal which listens and transmits on the standard T.120 well-known TSAP Identifier. This will allow the T.120 capable H.323 terminal to participate in T.120-only conferences.

A T.120-only terminal on the network shall be able to participate in the T.120 portion of multipoint H.323 conferences. See 6.2.7.1.

9.9 Gateway for H.323 media transport over ATM

It is possible to transport H.323 media streams originating from non-ATM IP networks over an ATM network using H.323-to-H.323 Gateways. This mechanism is described in AF-SAA-0124.000 [33].

10 Optional enhancements

10.1 Encryption

Authentication and security for H.323 systems is optional; however, if it is provided, it shall be provided in accordance with ITU-T Rec. H.235.

10.2 Multipoint operation

10.2.1 H.243 control and indication

H.245 contains multipoint control and indication messages carried forward from H.243. These messages may be used to provide certain multipoint capabilities (such as chair control) by following the procedures defined in ITU-T Rec. H.243.

NOTE – Clause 15/H.243 contains guidance for the implementation of these capabilities using the T.120 series of Recommendations.

10.3 Call Linkage in H.323

10.3.1 Description

Call Linkage in H.323 is an optional feature. A term "shall" within this clause shall be interpreted as a mandatory requirement provided the Call Linkage feature is supported.

10.3.1.1 General description

The Thread Identification feature allows different calls or call independent signalling connections – those that logically belong together from a service's or application's point of view in terms of their progression – to be linked together.

The Global Call Identification feature allows a call or a call independent signalling connection to be identified by one unique identifier that is applicable to the call or call independent signalling connection end-to-end without regards to its route or its history.

NOTE – The Call Identifier is defined in 7.5 as a globally unique identifier for a call. A new basic call from the same endpoint/entity or a new call as part of a service scenario would use a new Call Identifier value.

10.3.1.2 Service definitions

10.3.1.2.1 Thread identification, thread ID, TID

A value assigned to calls that are logically linked together for the purpose of correlating them. If two or more calls are logically linked together (e.g., due to service interactions), the current Thread ID of one of these calls is assigned to all of the other linked calls.

10.3.1.2.2 Global call identification, global call ID, GID

A value assigned to an end-to-end call to uniquely identify that call from end-to-end. If different calls are being transformed into a new call (i.e., due to service interactions), the GIDs of the old calls are updated (if already assigned previously) or assigned by a new GID value for the new end-to-end call.

NOTE – A call that is being transformed out of different call legs due to certain services may end up having call legs with different Call Identifiers. The Call Identifier is therefore not suitable to uniquely identify a call end-to-end.

10.3.2 Invocation and operation

A Call ID shall be assigned to each new call that is set up (see 7.5). Due to service interactions, different Call IDs may be assigned to different parts (call legs) of a call.

A Global Call ID may be assigned either at call establishment time, while in the active state or while call establishment/call clearing is in progress when two or more calls are being transformed into a new call due to certain services being invoked or due to an application request.

A Global Call ID may be changed during the lifetime of the call due to the call being transformed.

A Thread ID may be assigned either at call establishment time, while in the active state or while call establishment/call clearing is in progress when two or more calls are logically linked together due to certain services being invoked or due to an application request.

The Thread ID may be changed during the lifetime of a call (e.g., due to service interactions).

10.3.3 Interaction with H.450 supplementary services

Interactions with H.450 supplementary services for which standards were available at the time of publication of this Recommendation are specified below.

For the Call ID, no interactions with other supplementary services apply, as it shall be unique for each new call. All interactions described in this clause apply only to the Global Call ID and/or the Thread ID.

A Global Call ID and a Thread ID may be assigned, regardless of a supplementary service invocation, as part of the basic call establishment. Specific feature interactions are described below for specific supplementary service invocations.

10.3.3.1 Call transfer

This clause describes the usage of the Call Linkage fields when using H.450.2.

10.3.3.1.1 Transfer without consultation

The Thread ID of the transferred call shall be inherited from the Thread ID of the primary call. The Thread ID of the primary call shall therefore be provided by the transferring endpoint to the transferred endpoint along with the call transfer request. If the primary call does not have an assigned Thread ID, the transferring endpoint shall generate one. If the transferred entity does not receive a Thread ID along with the call transfer request, it shall inherit the Thread ID that was assigned to the primary call at call establishment time. If no Thread ID is available to inherit from at all, the transferred endpoint shall generate a Thread ID and assign it to both the transferred call (in call establishment message) and the primary call (in call clearing message).

A new Global Call ID shall be assigned to a transferred call. If a Gatekeeper establishes the transferred call on behalf of a transferred endpoint, the Gatekeeper shall assign the same Global Call ID to the remaining call leg of the primary call. This ensures that the resulting call after successful transfer has one unique GID end-to-end.

10.3.3.1.2 Transfer with consultation

At the time of transfer, the transferred call shall be assigned the same Thread ID as the former primary call if:

a) the primary call is an incoming call and the secondary call is an outgoing call; or

b) both calls are incoming calls and the primary call has been established before the secondary call; or

c) both calls are outgoing calls and the primary call has been established before the secondary call.

At the time of transfer, the transferred call shall be assigned the same Thread ID as the former secondary call if:

a) the secondary call is an incoming call and the primary call is an outgoing call; or

b) both calls are incoming calls and the secondary call has been established before the primary call; or

c) both calls are outgoing calls and the secondary call has been established before the primary call.

The Thread ID appropriate for the transferred call (either based on primary or secondary call depending on the situation) shall be provided by the transferring endpoint to the transferred endpoint along with the call transfer request. If the call from which the Thread ID shall be inherited (either primary or secondary call) does not have assigned a Thread ID, the transferring endpoint shall generate one. If the transferred endpoint does not receive a Thread ID along with the call transfer request (e.g., transferring endpoint does not support call linkage), it shall generate a Thread D that shall be inherited from the primary call if possible.

At the time of transfer, the transferred entity shall assign a new GID value to the transferred call. If a Gatekeeper established the transferred call on behalf of a transferred endpoint, the Gatekeeper shall assign the same GID to the remaining call leg of the primary call. A Gatekeeper acting on behalf of the transferred-to endpoint shall assign the same GID to the remaining part of the secondary call. This ensures that the resulting call after successful transfer has one unique GID end-to-end.

A transferring entity may, as an option, choose to "join" the primary call and the secondary call together. The call linkage rules for the resulting call ("joined" call) shall be the same as specified for a transferred call above.

10.3.3.2 Call diversion

This clause describes the usage of the Call Linkage fields when using ITU-T Rec. H.450.3 [40].

The originating call, the forwarding and the forwarded call shall use the same Thread ID.

The Thread ID of the forwarded call and the originating call shall be inherited from the Thread ID of the forwarding call. The served endpoint shall therefore assign a Thread ID to the forwarding call (if not already assigned as part of the basic call) and shall provide this Thread ID to the re-routing entity along with the call forwarding request. The re-routing entity shall use this Thread ID as the Thread ID for the establishment of the forwarded call. In addition, the originating call leg (if any) shall be assigned/updated with this Thread ID as well.

If the re-routing entity does not receive a Thread ID along with the call forwarding request, it shall inherit the Thread ID that was assigned to the forwarding call at call establishment time. If no Thread ID is available to inherit from at all, the re-routing endpoint shall generate a Thread ID and assign it to the forwarding call, the forwarded call, and to the originating call.

A new GID shall be assigned to the end-to-end call from the calling user (i.e., diverted user) to the diverted-to user by assigning a new GID in the forwarded call Setup and assigning (or updating) the same GID to the originating call leg (if any).

10.3.3.3 Call hold and consultation

This clause describes the usage of the Call Linkage fields when using ITU-T Rec. H.450.4.

A consultation call shall use the same Thread ID as the first call.

NOTE – Whether a call is considered being a consultation call rather than a further basic call is the decision of the endpoint.

A consultation call shall use a new Global Call ID.

10.3.3.4 Call park/call pickup

This clause describes the usage of the Call Linkage fields when using ITU-T Rec. H.450.5 [41].

The parked call shall have the same Thread ID as the primary call; however, it shall use a different GID.

If available, the Thread ID shall be used for associating call independent signalling connections (indicating group notifications and pickup requests), the call from a calling/parked user to the picking-up user, and a previously alerting/parked call.

NOTE – Call Park/Pickup contains a specific call pickup id that is used by the picking-up user.

The call independent signalling connections used as part of Call Park/Call Pickup shall use new GIDs. The call from the calling user/parked user to the picking-up user shall have a new end-to-end global GID.

10.3.3.5 Call waiting

There is no interaction with Call Linkage and ITU-T Rec. H.450.6 [42].

10.3.3.6 Message waiting indication

There is no interaction with Call Linkage and ITU-T Rec. H.450.7 [43].

10.3.3.7 Name identification service

There is no interaction with Call Linkage and ITU-T Rec. H.450.8 [44].

10.4 Tunnelling of non-H.323 signalling messages

In order to support existing non-H.323 signalling information in an H.323 system, it is necessary to allow for transport of non-H.323 signalling information in H.323. This clause provides a generic means of tunnelling signalling messages in any H.225.0 call control message.

The procedures of this clause apply to any type of endpoint. Signalling tunnels are terminated in a logical entity called a "tunnel termination". Typically, these tunnel terminations are located in gateways that interconnect parts of a non-H.323 network over a H.323 network as shown in Figure 53. If a Gatekeeper is present in the H.323 network, it may participate in the tunnelling of non-H.323 signalling.

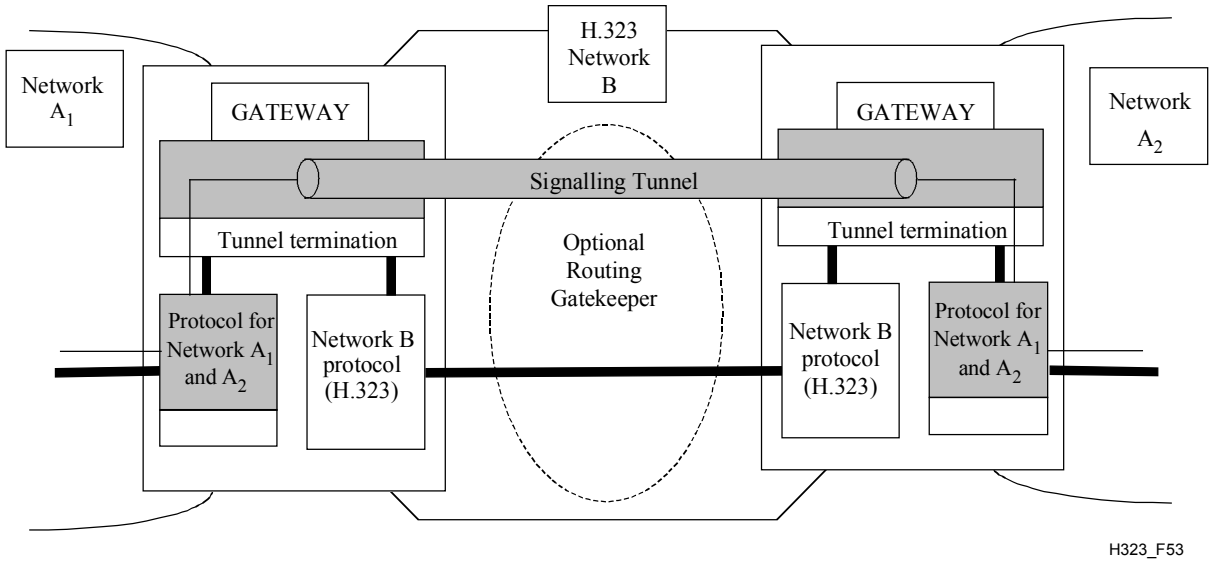


Figure 53/H.323 –Signalling tunnelling between gateways

In some cases, the tunnel termination may be located in a Gatekeeper, as illustrated in Figure 54. Clause 10.4.2 describes Gatekeeper intervention in a tunnel.

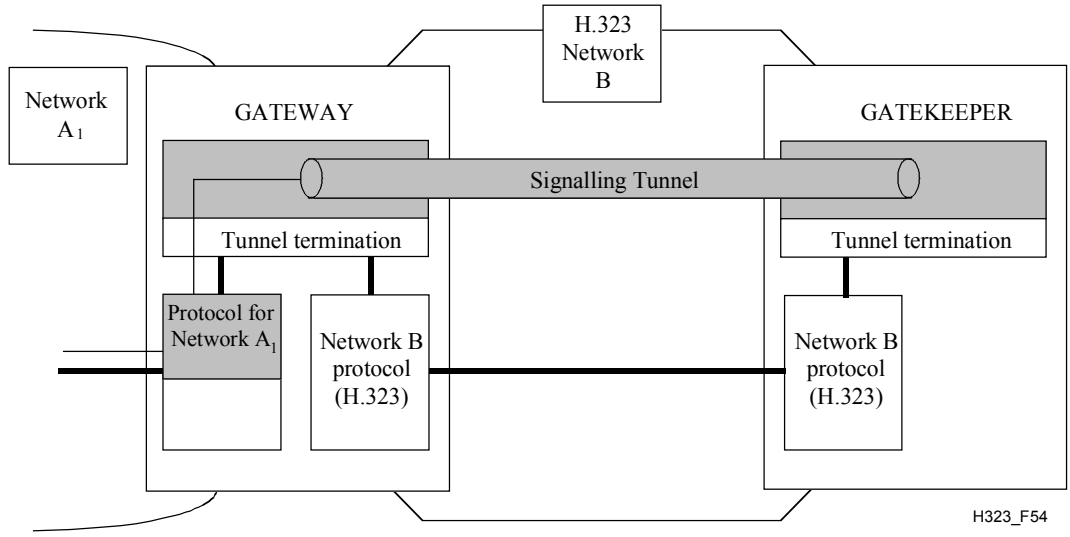


Figure 54/H.323 – Signalling tunnelling between a gateway and an tunnel termination in a gatekeeper

The call control states and procedures of the tunnelled protocol are distinct from the call control states and procedures of the H.225.0 protocol: an endpoint supporting tunnelled signalling should view the two separately.

Any signalling protocol may be tunnelled and is identified by the **TunnelledProtocol**. Examples of signalling protocols that may be tunnelled include:

- QSIG.
- ISUP.
- ISDN DSS1.
- DPNSS.
- Proprietary PBX networking protocol.

10.4.1 Indicating support of tunnelled protocols

Tunnelling support for a prioritized list of protocols is indicated with the **supportedTunnelledProtocols** field of the **EndpointType**. This list consists of a prioritized list of protocols that can be tunnelled.

When registering with its Gatekeeper, an endpoint may indicate the tunnelling protocols supported in the GRQ and RRQ as part of the **EndpointType**. The **EndpointType** contains a prioritized list of supported tunnelled protocols, with the first one being the preferred one. In the ACF or LCF that a Gatekeeper returns in response from an ARQ or LRQ, the **destinationType** indicates the destination's supported tunnelled signalling protocols also in a prioritized list. Since Annex G/H.225.0 imports the **EndpointType** sequence, this capability may also be conveyed through Annex G/H.225.0.

An originating endpoint wishing to indicate the signalling protocols it can tunnel shall include the prioritized list in the **sourceInfo.supportedTunnelledProtocols** in the Setup message. A terminating endpoint wishing to indicate the signalling protocols it can tunnel shall include the prioritized list in the **destinationInfo.supportedTunnelledProtocols** in all the messages including the **destinationInfo** field it sends in response to the Setup message. If an originating endpoint does not receive this indication, it shall assume that the terminating endpoint does not support any tunnelled protocols.

10.4.2 Requesting a specific protocol tunnel to a gatekeeper

An entity may request a specific protocol tunnel to a Gatekeeper by specifying the particular protocol in the **desiredTunnelledProtocol** field in an ARQ or LRQ.

10.4.3 Tunnelling a signalling protocol in H.225.0 call signalling messages

An endpoint may tunnel a signalling protocol by including the **tunnelledSignallingMessage** in any H.225.0 call signalling message. However, it is not recommended to tunnel a signalling protocol in H.225.0 call signalling messages that are not of end-to-end significance, such as Call Proceeding, since the information may not be received by the other end.

If an endpoint will only allow the call to proceed if tunnelling is supported, it shall set the **tunnellingRequired** flag in the Setup message; the **tunnellingRequired** flag shall not be included in any other message than Setup. If an endpoint receives a **tunnelledSignallingMessage** with the **tunnellingRequired** flag set in the Setup message and is not able to tunnel the protocol, it shall terminate the call by sending a Release Complete with a **reason** of **tunnelledSignallingRejected**; a **tunnellingRequired** flag in any other message than Setup shall be ignored.

The tunnelled protocol information is included in the **messageContent** field and the **tunnelledProtocolID** field identifies the protocol being tunnelled. Only a single protocol can be tunnelled in an H.323 call. Multiple tunnelled messages of the same protocol may be aggregated in one single H.225.0 call signalling message.

The tunnel shall be released using the normal H.323 release procedures.

The call signalling procedures of H.225.0 can be used to establish a call independent signalling connection between the peer endpoints. Tunnelling can be used in this context to provide bearer independent signalling for the tunnelled protocol. In this case, no H.245 Control Channel and no media channels are required. A bearer capability information element should be included in the H.225.0 Setup message and coded as described in Table 2/H.450.1. The Setup message used for call independent procedures shall include a **conferenceGoal** within Setup set to value **callIndependentSupplementaryService**. These call independent signalling connection procedures for tunnelling shall not be used in conjunction with an H.450 supplementary service in the same call independent signalling connection.

10.4.4 Gatekeeper considerations

In a direct routed call model, the Gatekeeper is not involved in the H.225.0 call control signalling and therefore does not perform signalling tunnelling in H.225.0. Such Gatekeepers do not affect tunnelling between two endpoints supporting signalling tunnelling. In a Gatekeeper routed model, the Gatekeeper participates in providing a tunnel between peer endpoints by passing on received tunnelled signalling information. The Gatekeeper may also utilize the Facility or Progress message to convey tunnelled messages, as discussed in 8.2.2.

In the Gatekeeper routed model, the Gatekeeper may intercept and act on tunnelled signalling messages. Termination of a signalling tunnel is performed by a tunnel termination function, which, as described earlier, can be located in the Gatekeeper. What the Gatekeeper does with the tunnelled protocol is outside the scope of this Recommendation. However, if the Gatekeeper is capable of providing non-H.323 signalling service, it may terminate the signalling tunnel and generate appropriate H.225.0 messages for the endpoints involved in the call. Alternatively, it may modify the tunnelled signalling information: if it does, it is taking the responsibility of terminating and initiating the tunnelled protocol. A Gatekeeper that does not understand the tunnelled protocol, or does not intend to act on the tunnelled protocol or provide any services in that plane, shall pass the tunnelled signalling message through unchanged to preserve the integrity of the tunnelled protocol.

10.5 Use of RTP payload for DTMF digits, telephony tones and telephony signals

It is possible to carry DTMF tones, fax-related tones, standard subscriber line tones, country-specific tones and trunk events using a distinct dynamic RTP payload type in the same RTP stream as the media. Many applications, such as IVR systems and voice systems rely on synchronization of DTMF input.

RFC 2833 [58] describes means for transporting these tones and events over RTP. An endpoint may indicate support for receiving these RFC 2833 tones and events by including the **receiveRTPAudioTelephonyEventCapability** or the **receiveRTPAudioToneCapability** in the terminal capability set. Alternatively, an endpoint may indicate support for RFC 2833 tones and events by including the **audioTelephonyEvent** or the **audioToneAudioCapability** in the terminal capability set. When using fast connect procedures, these capabilities can be sent using parallelH245 procedures of 8.2.4.

Named telephone events are a logical description of DTMF tones, fax-related tones, standard subscriber line tone, country-specific tones and trunk events. A decimal number identifies each event. When telephone events are used, support for the following DTMF is mandatory: 0-9, #, *, A, B, C, D. All others are optional.

Telephony tones are a description of the waveform properties. This is useful in cases where it is necessary to accurately reproduce non-standard tones.

After a logical channel has been opened for the media stream, the sender may send any of the telephony events or tones advertised by the receiver in the terminal capability set on that same logical channel using the RTP payload type negotiated in the terminal capability set negotiation.

If an endpoint sends DTMF information, it may send it in a **UserInputIndication** and/or using RTP payload for DTMF digits, telephony tones, and telephony signals.

If the DTMF is sent both via RTP and in a **UserInputIndication** in alphanumeric form, it shall be encoded in the **extendedAlphanumeric** structure and the **rtpPayloadIndication** field shall be included. If the DTMF is sent both via RTP and in a **UserInputIndication** in the signal form, the **rtpPayloadIndication** field shall be included in the **signal** structure. If the DTMF is sent only in alphanumeric form, it shall be encoded in the **alphanumeric** field. If the DTMF is sent only in signal form, the **rtpPayloadIndication** field shall not be included.

RFC 2833 shall not be used to relay fax information in H.323 systems. Instead, the procedures defined in Annex D shall be followed for endpoints that wish to transmit T.38 fax information.

NOTE – H.323 entities prior to version 4 did not have the capability of sending DTMF information via RTP as described in this clause. Therefore, all entities shall support the ability to send DTMF information via the **UserInputIndication** message.

11 Maintenance

11.1 Loopbacks for maintenance purposes

Some loopback functions are defined in ITU-T Rec. H.245 to allow verification of some functional aspects of the terminal, to ensure correct operation of the system and satisfactory quality of the service to the remote party.

The **systemLoop** request and **logicalChannelLoop** request shall not be used. The **mediaLoop** request is optional. An endpoint that receives the **maintenanceLoopOffCommand** shall turn off all loopbacks currently in effect.

For the purpose of loopbacks, two modes are defined:

a) Normal operation mode: No loopback. Indicated in **a)** of Figure 55. This shall be the default mode, and the mode entered when the **maintenanceLoopOffCommand** is received.

b) Media loop mode: Loopback of media stream at the analogue I/O interface. Upon receiving the **mediaLoop** request as defined in ITU-T Rec. H.245, loopback of the content of the selected logical channel shall be activated as close as possible to the analogue interface of the video/audio codec towards the video/audio codec, so that decoded and re-coded media content is looped, as indicated in **b)** of Figure 55. This loopback is optional. It should be used only when a single logical channel containing the same media type is opened in each direction. Operation when multiple channels are opened in the return direction is undefined.

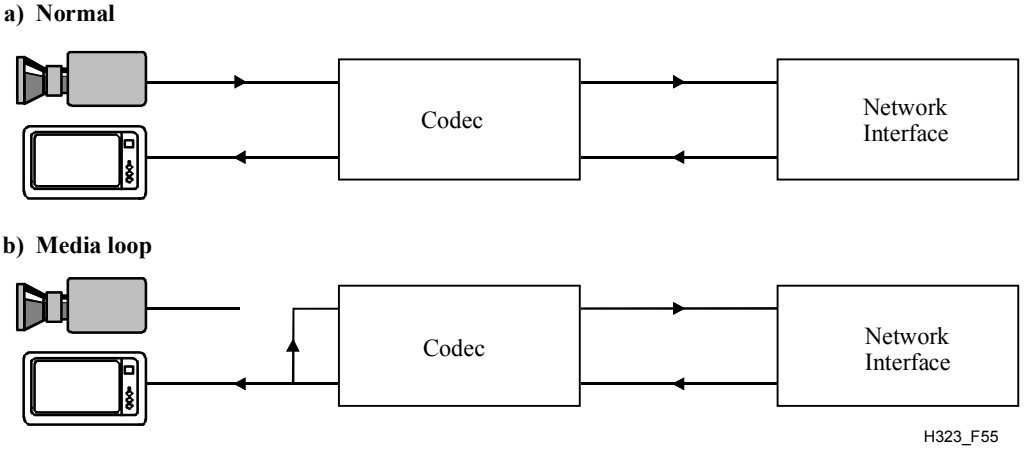


Figure 55/H.323 – Loopback

A Gateway to H.324, which receives an H.245 **systemLoop** request, H.245 **logicalChannelLoop** request, or a Gateway to H.320, H.321, or H.322, which receives an H.230 Dig-Loop command from an SCN endpoint may perform the appropriate loopback function within the Gateway. The Gateway shall not pass these requests to the network endpoint. A Gateway to H.324, receiving H.245 **mediaLoop** from an SCN endpoint shall pass the request to the network endpoint. A Gateway to H.320, H.321, or H.322, receiving H.230 Vid-loop or Aμ-loop command from an SCN endpoint shall convert it to the appropriate H.245 **mediaLoop** request and send it to the network endpoint.

A Gateway to H.320, H.321, or H.322, which receives an H.245 **mediaLoop** request from a network endpoint shall convert it to the appropriate H.230 Vid-loop or Aμ-loop command and send it to the SCN endpoint.

A Gateway to H.324 may send an H.245 **systemLoop** request or H.245 **logicalChannelLoop** request to the SCN endpoint. A Gateway to H.320, H.321, or H.322 may send an H.230 Dig-Loop command to the SCN endpoint. If a network endpoint is in a call to the SCN endpoint, the audio and video sent to the network endpoint may be the looped back audio or video, pre-recorded audio or video message indicating the loopback condition, or no audio or video.

11.2 Monitoring methods

All terminals shall support the Information Request/Information Request Response (IRQ/IRR) message of ITU-T Rec. H.225.0. The Information Request Response message contains the TSAP Identifier of all channels currently active on the call, including T.120 and H.245 control, as well as audio and video. This information can be used by third party maintenance devices to monitor H.323 conferences to verify system operation.

Annex A

H.245 messages used by H.323 endpoints

The following rules apply to the use of H.245 messages by H.323 endpoints:

- An endpoint shall not malfunction or otherwise be adversely affected by receiving H.245 messages that it does not recognize. An endpoint receiving an unrecognized request, response, or command shall return "function not supported". (This is not required for indications.)
- The following abbreviations are used in Tables A.1 to A.12:

  M Mandatory.

  O Optional.

  F Forbidden to transmit.
- A message marked as mandatory for the receiving endpoint indicates that the endpoint shall accept the message and take the appropriate action. A message marked as mandatory for the transmitting endpoint indicates that the endpoint shall generate the message under the appropriate circumstances.

Table A.1/H.323 – Master-slave determination messages

Message	Receiving endpoint status	Transmitting endpoint status
Determination	M	M
Determination Acknowledge	M	M
Determination Reject	M	M
Determination Release	M	M

Table A.2/H.323 – Terminal capability messages

Message	Receiving endpoint status	Transmitting endpoint status
Capability Set	M	M
Capability Set Acknowledge	M	M
Capability Set Reject	M	M
Capability Set Release	M	M

Table A.3/H.323 – Logical channel signalling messages

Message	Receiving endpoint status	Transmitting endpoint status
Open Logical Channel	M	M
Open Logical Channel Acknowledge	M	M
Open Logical Channel Reject	M	M
Open Logical Channel Confirm	M	M
Close Logical Channel	M	M
Close Logical Channel Acknowledge	M	M
Request Channel Close	M	O
Request Channel Close Acknowledge	O	O
Request Channel Close Reject	O	M
Request Channel Close Release	O	M

Table A.4/H.323 – Multiplex table signalling messages

Message	Status
Multiplex Entry Send	F
Multiplex Entry Send Acknowledge	F
Multiplex Entry Send Reject	F
Multiplex Entry Send Release	F

Table A.5/H.323 – Request multiplex table signalling messages

Message	Status
Request Multiplex Entry	F
Request Multiplex Entry Acknowledge	F
Request Multiplex Entry Reject	F
Request Multiplex Entry Release	F

Table A.6/H.323 – Request mode messages

Message	Receiving endpoint status	Transmitting endpoint status
Request Mode	M	O
Request Mode Acknowledge	M	O
Request Mode Reject	O	M
Request Mode Release	O	M

Table A.7/H.323 – Round trip delay messages

Message	Receiving endpoint status	Transmitting endpoint status
Round Trip Delay Request	M	O
Round Trip Delay Response	O	M

Table A.8/H.323 – Maintenance loop messages

Message	Receiving endpoint status	Transmitting endpoint status
Maintenance Loop Request		
System Loop	F	F
Media Loop	O (Note)	O (Note)
Logical Channel Loop	F	F
Maintenance Loop Acknowledge	O	O
Maintenance Loop Reject	O	M
Maintenance Loop Command Off	M	O
NOTE – Mandatory in Gateways.		

Table A.9/H.323 – Conference request and response messages

Message	Receiving endpoint status	Transmitting endpoint status
Terminal List Request	O	O
Drop Terminal	O	O
Make Me Chair	O	O
Cancel Make Me Chair	O	O
Enter H.243 Password	O	O
Enter H.243 Terminal Id	O	O
Enter H.243 Conference ID	O	O
Request Terminal ID	O	O
Terminal ID Response	O	O
MC Terminal ID Response	O	O
Enter Extension Address	O	O
Enter Address Response	O	O
Terminal List Response	O	O
Make Me Chair Response	O	O
Conference ID Response	O	O
Password Response	O	O

Table A.10/H.323 – Commands

Message	Receiving endpoint status	Transmitting endpoint status
Send Terminal Capability Set	M	M
Encryption	O	O
Flow Control	M	O
End Session	M	M
Miscellaneous Commands		
Equalize Delay	O	O
Zero Delay	O	O
Multipoint Mode Command	M	O
Cancel Multipoint Mode Command	M	O
Video Freeze Picture	M	O
Video Fast Update Picture	M	O
Video Fast Update GOB	M	O
Video Fast Update MB	M	O
Video Temporal Spatial Trade Off	O	O
Video Send Sync Every GOB	O	O
Video Send Sync Every GOB Cancel	O	O
Terminal ID Request	O	O
Video Command Reject	O	O
Make Me Chair Response	O	O
Conference Commands		
Broadcast My Logical Channel Me	O	O
Cancel Broadcast My Logical Channel Me	O	O
Make Terminal Broadcaster	O	O
Cancel Make Terminal Broadcaster	O	O
Send This Source	O	O
Cancel Send This Source	O	O
Drop Conference	O	O

Table A.11/H.323 – Conference mode commands

Message	Receiving endpoint status	Transmitting endpoint status
Communication Mode Command	M	O
Communication Mode Request	O	O
Communication Mode Response	O	O

Table A.12/H.323 – Indications

Message	Receiving endpoint status	Transmitting endpoint status
Function Not Understood	M	M
Function Not Supported	M	M
Miscellaneous Indication		
Logical Channel Active	O	O
Logical Channel Inactive	O	O
Multipoint Conference	M	O
Cancel Multipoint Conference	M	O
Multipoint Zero Comm	O	O
Cancel Multipoint Zero Comm	O	O
Multipoint Secondary Status	O	O
Cancel Multipoint Secondary Status	O	O
Video Indicate Ready to Activate	O	O
Video Temporal Spatial Trade Off	O	O
Video Not Decoded MBs	O	O
Conference Indications		
SBE Number	O	O
Terminal Number Assign	M	O
Terminal Joined Conference	O	O
Terminal Left Conference	O	O
Seen By At Least One Other	O	O
Cancel Seen By At Least One Other	O	O
Seen By All	O	O
Cancel Seen By All	O	O
Terminal You Are Seeing	O	O
Request For Floor	O	O
Vendor Indications	O	O
MC Location Indication	M	O
Jitter Indication	O	O
H.223 Skew Indication	F	F
H2250MaximumSkewIndication	O	M
New ATM Virtual Channel Indication	F	F
User Input	M (for 0-9, *and #)	M (for 0-9, *and #)

Non-standard commands, requests, etc. are allowed.

Annex B

Procedures for layered video codecs

B.1 Scope

This annex describes enhancements within the framework of the H.323 specification, to incorporate layered video codecs. The described procedure is scalable for multipoint conferences.

B.2 Introduction

Layered video coding is a technique that allows the video information to be transmitted in multiple data streams in order to achieve scalability. These may provide bandwidth scalability, temporal scalability, SNR scalability, and/or spatial scalability. Annex O/H.263 describes the use of layered coding within H.263. Conferences can take advantage of this feature to service connected endpoints that have different capabilities, using one bitstream. This will allow more efficient use of network bandwidth.

B.3 Scalability methods

Scalability of a video stream refers to the generation of a stream that may only be decoded in part due to limitations of available resources. Scalability may be desired to overcome limitations of available computing power or to accommodate bandwidth limitations.

There are three types of scaling: Temporal, Signal-to-Noise Ratio (SNR), and Spatial that are available in ITU-T Rec. H.263. Other video codecs may have similar layering capability. All of these methods can be used separately or together to create a multi-layer scalable bit stream. The resolution, frame rate, and quality of the image can only increase by adding scaling layers. The base layer can be used to guarantee a minimum level of image quality. Endpoints can then use additional layers to add image quality by increasing frame rate, display frame size, or accuracy of decoded images. Allowing multiple scaling methods in a conference can add resource efficiency, especially when endpoints participating have varying processing and bandwidth capabilities. This is especially true for multipoint and loosely-coupled conferences.

B.4 Call establishment

H.323 call establishment takes place following the same procedures described in clause 8. The layered coding capability will be signalled using the H.245 capabilities exchange methods. Codepoints within H.245 exist which clearly identify what layering methods are supported by the endpoints. The endpoints shall use these capabilities in order to signal the exact layering methods they support.

The use of simultaneous capabilities methods in H.245 shall be used to indicate which layering methods will be used together to create the video layers when they are going to be sent in two or more logical channels. It is also possible to send two or more layers in single logical channels. The exact video layers that will be used are signalled during the **openLogicalChannel** in the same manner that is currently used to indicate what video **dataType** will be used, except that the endpoint shall indicate dependencies between the base layer logical channel and the enhancement layer logical channels.

B.5 Use of RTP sessions and codec layers

It is desired to allow separate RTP sessions for the different qualities of video that are available. The base layer should be considered the primary video session, and its level considered the minimum quality of video that is available in the conference. Enhancement layers can be sent on separate RTP sessions. The **forward/reverseLogicalChannelDependency** parameter, added to

H.245 **openLogicalChannel** command, shall be used to indicate how the video layers are organized. This is outlined in the following clauses. RTP Timestamps must be the same in the base and all dependent enhancement layers corresponding to a frame to allow reassembly and proper display.

B.5.1 Associate base to audio for lip synchronization

The base video session should be associated with the audio session corresponding with the audio track of the video, for lip synch purposes. This is done in the same manner that existing non-layered video sessions are associated with their corresponding audio. This is done using the **associatedSessionID** and the **sessionID** parameters located in the **H2250LogicalChannelParameters**. The enhancement layers may also be associated with the audio or with the base layer using the **associatedSessionID**. Coding dependency shall be indicated using the **forwardLogicalChannelDependency** and the **reverseLogicalChannelDependency** parameter in the **openLogicalChannel** command as explained below.

B.5.2 Enhancement layer dependency

Enhancement layer dependency can create many complex cases using multiple layers that contain multiple enhancement frame types. Dependency between layers shall be indicated using the **forward/reverseLogicalChannelDependency** parameter, added to H.245 **openLogicalChannel** command. Dependency is used to indicate that the data sent on the logical channel cannot be used without the contents of the logical channel it is dependent on. Enhancement layers, by definition, must be differentially coded from the video layer they are enhancing and are therefore dependent on that video layer for meaningful decoding. If an enhancement layer is sent on a separate logical channel, it shall indicate the layer it was differentially coded from in the **forward/reverseLogicalChannelDependency** parameter.

Since the **forward/reverseLogicalChannelDependency** parameter allows the indication of a single logical channel, the logical channels need to be opened in order of dependence starting with the base layer. An endpoint shall have either sent or received the **openLogicalChannelAck** for any logical channel that is used in a **forward/reverseLogicalChannelDependency** parameter. An endpoint shall send an **openLogicalChannel** for a dependent logical channel, only after the logical channel on which it is dependent is opened and acknowledged. Logical channels that have common dependency may be opened in parallel. Enhancement layers must be indicated to be dependent on the highest layer that is required for proper decoding.

Assuming that separate RTP sessions are used for each layer, an example can be built as shown in Figure B.1.

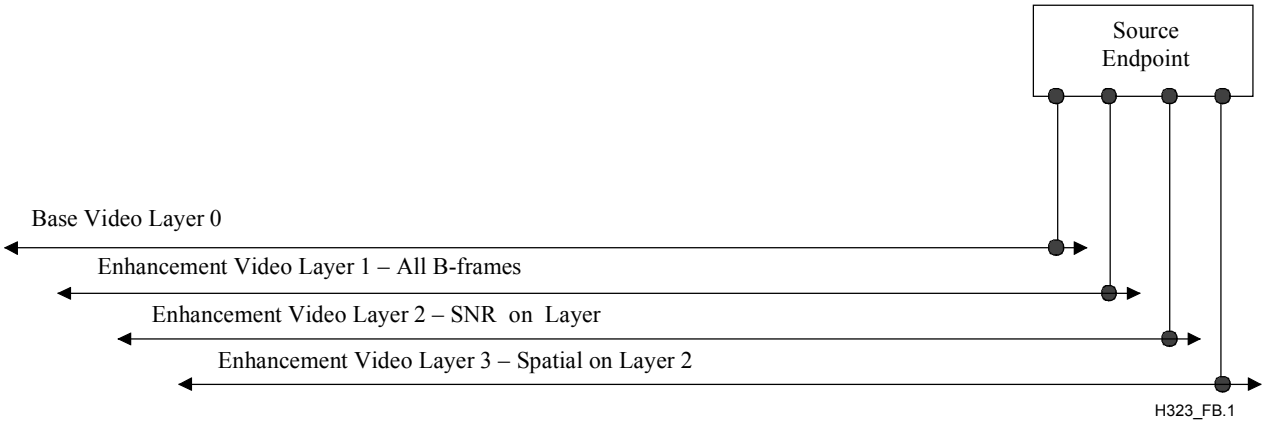


Figure B.1/H.323 – Model with layered video

In this example, layered video is created that has four layers:

1) The base video, not dependent on any other layer. This is associated with its corresponding audio.
2) Enhancement level one consisting of B-frames, dependent on the base video. This is indicated to be dependent on the base video session, Layer 0.
3) Enhancement level two that is SNR enhancement of the base video, dependent only on the base video, Layer 0. This is indicated to be dependent on the base video session.
4) Enhancement level three that consists of spatial enhancement of enhancement level two, dependent on Layer 2, which implies the base is also required. This is indicated to be dependent on the video in Layer 2.

In this example, the base video logical channel must be opened first. The **openLogicalChannel** for enhancement Layers 1 and 2 may be sent in parallel, only after receiving the **openLogicalChannelAck** for the base video logical channel. The **openLogicalChannel** for enhancement Layer 3 can only be sent after the **openLogicalChannelAck** has been received or sent for the logical channel used for enhancement Layer 2.

B.6 Possible layering models

There are many possible methods for layering of the video and organization of the corresponding RTP sessions. The reason that the layers may need to be separated is that they are used for either decoder power scaling or for bandwidth usage scaling. It may be desirable to separate all non-B-frames into separate layers that can be discarded if they cannot be used. An important feature of the layered codec is that at any time an endpoint may discard any or all enhancement layers, without affecting the quality of the base video, in order to provide decoder power scaling.

In a similar manner, the layers may need to be organized into bandwidth usage levels that correspond to the bandwidths reported by the endpoints that are connected to the conference. This would allow the conference to accommodate multipoint conferences that have endpoints using connection methods that may limit the available bandwidth and create a layer that gives them the best possible video at that bandwidth. The endpoint may add or subtract layers as its available bandwidth varies up and down.

B.6.1 Multiple logical channels and RTP sessions for a layered stream

If bandwidth scaling is the goal of using layering, each layer should flow on a separate logical channel with a separate RTP session. This means that what is a single video source will now have to be coordinated amongst multiple logical channels and RTP sessions.

If the goal of layering is processor-power scaling, the enhancement layers can be sent, with the base video on a single logical channel and RTP session.

If the goal is a mixture of bandwidth and processor-power scaling, then groups of enhancement layers, sent in logical channels on a group basis can be sent. The choice of layers and grouping is a choice based on system need. The method used to make these choices is an implementation issue and outside the scope of this Recommendation.

B.6.2 Impact of one layer per logical channel and per RTP session

The impact of using a single logical channel and RTP session for each layer is that the encoder and decoder are burdened with having to split and reassemble the video stream according to the chosen layering model. This model is signalled to the receiving side so that it can properly interpret the layer information. It is signalled using H.245 capabilities, with a capability per logical channel that, when combined with the dependencies, will sufficiently describe the layering model. Possible layering models are signalled during capabilities exchange, using the simultaneous capabilities feature of ITU-T Rec. H.245.

Strict timing consideration will need to be used to ensure that the layers are properly synchronized. For H.323, this will be handled in the RTP payload format.

B.7 Impact on multipoint conferences

The most likely envisaged usage of video layering is in multipoint conferences. In H.323, this can be performed by a centralized MCU, used for audio mixing and video switching, or using a decentralized model, with each endpoint responsible for video switching and audio mixing. In either case, the MC should perform the function of reporting what the layering model is for the conference. This is done using the **communicationModeCommand**.

In order for an endpoint to receive a video layer, a logical channel containing that layer must be opened. The decision to open a logical channel can be made by either the MC or the endpoint sending an **openLogicalChannel**. If an MC or endpoint decides not to open a logical channel, it must reject the **openLogicalChannel** when it is offered. The MC or endpoint can only offer a logical channel that corresponds to a **dataType** that is supported by the receiving endpoint.

When implementing support for layer codecs, an MC can take two approaches. If the MC does not make any decisions as to what logical channels will be opened, it can be called the "MC Impartial" model. In this model the MC offers all media to all endpoints without regard to any reported QOS. When the MC makes the decision to strictly enforce QOS, it is called the "MC Decision" model. These models are explained further below.

B.7.1 MC Impartial model

The MC Impartial model does not depend on the QOS capability set additions and as such may allow for a simpler MC implementation. In this case, the endpoint must judge whether it has sufficient bandwidth to accept logical channels offered by the MC. If it will exceed the transmission capabilities of the endpoint or the underlying network, then the endpoint may reject the logical channel. This method will require the endpoint to have knowledge of the network bandwidth available. The MC should indicate all available media in the **communicationModeCommand**.

B.7.2 MC Decision model

The MC Decision model depends upon the addition of Quality of Service (QOS) capabilities to the Terminal Capability Set. This has been previously proposed and is work in progress. The MC can then examine the QOS capabilities of the endpoints and only offer logical channels that are within the QOS of the endpoint. The endpoint will need to determine its available QOS at the start of the conference and indicate this using the QOS capabilities defined by work in progress.

In the MC Decision model, the MC may send a **communicationModeCommand** to an endpoint that only shows the sessions within the endpoint's QOS capabilities. In this way, the MC can strictly enforce bandwidth usage.

B.7.3 Multipoint conference containing endpoints on different bandwidths

In the model where the multipoint conference contains endpoints that have different bandwidth capabilities, the layering will need to be tuned to match these bandwidth levels. This can be done by using two possible models. One is illustrated in Figure B.2.

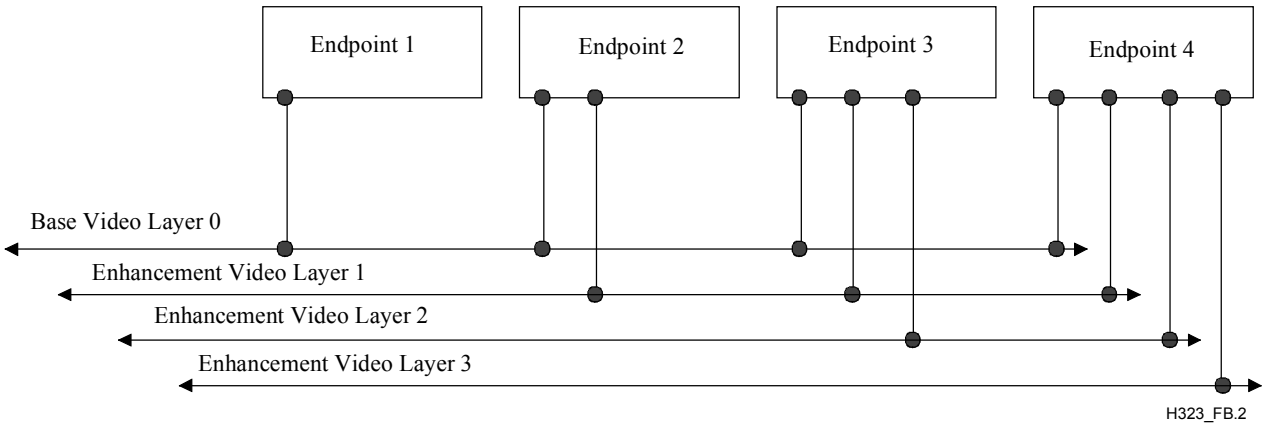


Figure B.2/H.323 – Endpoints attached to one or more layers according to bandwidth

In this case, the endpoints are attached to the base layer of video and the enhancement layers up to the total bandwidth desired. Each enhancement layer is on a separate logical channel. The endpoints are burdened with recombination of the layers to create the video stream. The sending endpoint must have capability for the combined bandwidth of all streams it sources. In this case, each endpoint may have communicated a different set of capabilities. The MC will examine the capabilities and QOS and create a layering model that is likely to provide the best use of the endpoints capabilities and bandwidth. This layering is indicated in the **communicationModeCommand** by the indication of **sessionDependency** in the **communicationModeTableEntry**. The **sessionDependency** field is set by the MC to indicate when a session is dependent on another session for meaningful decoding of its data. This information will be translated into **logicalChannelNumbers** when opening a dependent logical channel, according to the actual logical channels that are opened.

In the above case, using the MC Decision model, the MC will then offer the endpoints the logical channels that correspond to the layers that match the endpoint's capabilities. The MC will offer Endpoint 1 only the logical channel corresponding to the Base Video Layer. Endpoint 2 will be offered the logical channels corresponding to base video and enhancement video Layer 1. Endpoint three is offered three logical channels corresponding to the base video and two enhancements layers, and Endpoint 4 is offered all video logical channels.

In the MC impartial case, the MC will offer all logical channels, to all endpoints, that are within their **dataType** capabilities. The endpoints will refuse any logical channel that will cause them to exceed their bandwidth capabilities.

A second layering model is shown in Figure B.3. In this model each logical channel contains a totally independent video stream.

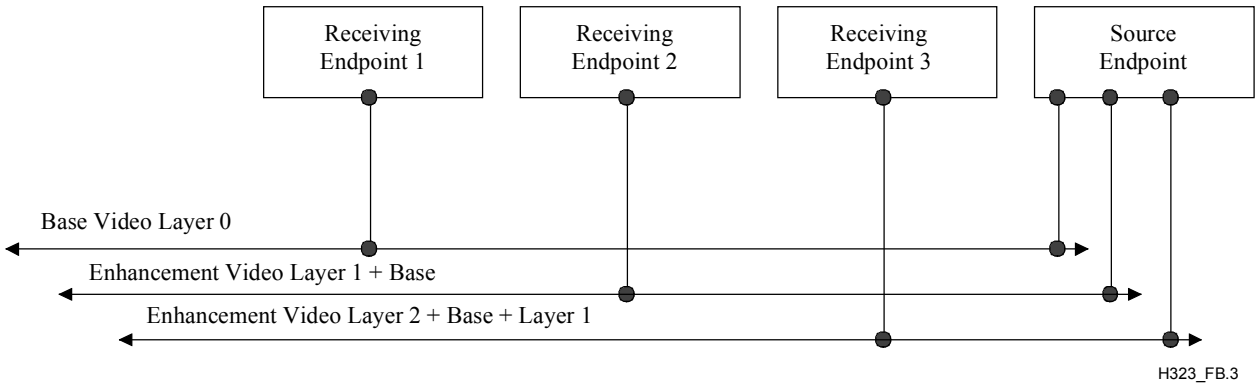


Figure B.3/H.323 – Endpoints attached to single Layer according to bandwidth

In this case, the endpoint shall connect only to the logical channel that corresponds to the bandwidth it has available. This stream has all layers that build the video stream to the bandwidth of the logical channel. This method eliminates the burden from the endpoints to recombine the video, but burdens the sender with producing several video streams. This is a less efficient use of network resources, since enhancement layers include all lower layers.

In order to perform proper lip synch, any session containing base video should be associated with the audio session corresponding to its audio track, using the **associatedSessionID** in the **H2250LogicalChannelParameters**. In the example shown in Figure B.2, the base video session should be associated with the audio session for lip synch. In the example shown in Figure B.3, all three video sessions should be associated with the audio session for lip synch, since all three contain base video.

B.8 Use of network QOS for layered video streams

Several important characteristics of the nature of layered coding usage should be considered when using network QOS for delivery of layered coded video streams. An enhancement layer cannot be decoded properly without receiving the layers on which it is dependent. Enhancement video layers may be discarded without affecting the decoding of the layer on which they are dependent.

If available, network QOS may be used to help guarantee that a video stream will be delivered by the network. Since layered video may be delivered using multiple streams, delivered on separate network connections, different QOS can be used on each video layer. QOS used on layered video streams should be specified when the logical channel is opened.

It is important that a dependent video layer has the information on which they are dependent at the time the dependent layer is to be decoded. This leads to general rules regarding use of QOS:

1) Dependent layers that are delivered using network QOS should have the layer they are dependent on, also delivered using QOS.
2) The base layer should be delivered using network QOS, if any other video layers in the conference are to be delivered using QOS.
3) The nearer the video layer is to the base layer, the stronger the delivery guarantees should be.

Annex C

H.323 on ATM

C.1 Introduction

This is an optional enhancement allowing H.323 endpoints to establish QOS-based media streams on ATM networks using AAL 5.

C.2 Scope

This annex specifies an improved method of using H.323 on AAL 5. H.323 can always be used on ATM by making use of an IP over ATM method. However, this is less efficient than using AAL 5 Virtual Channels (VCs) directly for the transport of the audio and video streams of H.323. When the media streams flow directly on AAL 5, they can benefit from a QOS-based ATM VC.

This annex retains the use of a packet network protocol for H.245 and H.225.0 communications to ensure interoperability with H.323 endpoints that are using a packet network protocol for all streams (whether over ATM or other media). Interoperability with legacy H.323 endpoints is achieved, without the use of a Gateway, by first requiring the basic mode of operation, in which an endpoint sends media streams on a datagram service using a packet network protocol, for example UDP/IP over ATM. In basic mode, unless a packet network protocol infrastructure has been upgraded, QOS may not be available from the network.

C.2.1 Point-to-point conferencing

This annex specifies a method of point-to-point communication between two H.323 endpoints using AAL 5 VCs for the media streams. The protocol necessary for entering into this mode is specified, as are information elements to be used in ATM signalling.

C.2.2 MCU-based multipoint

It follows that multipoint MCU-based communications can occur among several H.323 endpoints using AAL 5 VCs for the media streams. Currently no support is specified for the H.323 Decentralized Multipoint using ATM point-to-multipoint capability. This is left for further study.

C.2.3 H.323 interoperability with endpoints using IP

Interoperability is guaranteed with an endpoint using IP for the entire H.323 connection. This annex defines methods that allow an endpoint to detect if support is present for the option of using AAL 5 directly. An endpoint conforming to this annex must accept that the audio and video streams may occur on either AAL 5 VCs or UDP/IP ports.

C.3 Architecture

The basic protocol architecture of the system is shown in Figure C.1. It uses IP on ATM for delivery of the H.225.0 and H.245 messages and for the RTCP part of the audio and video streams. It uses AAL 5 directly for the RTP part of the audio and video streams.

NOTE – The H.323 media streams, compressed into variable length packets according to ITU-T Rec. H.225.0, are easily mapped to AAL 5. It would be difficult to map them to AAL 1, and this alternative has no clear benefit.

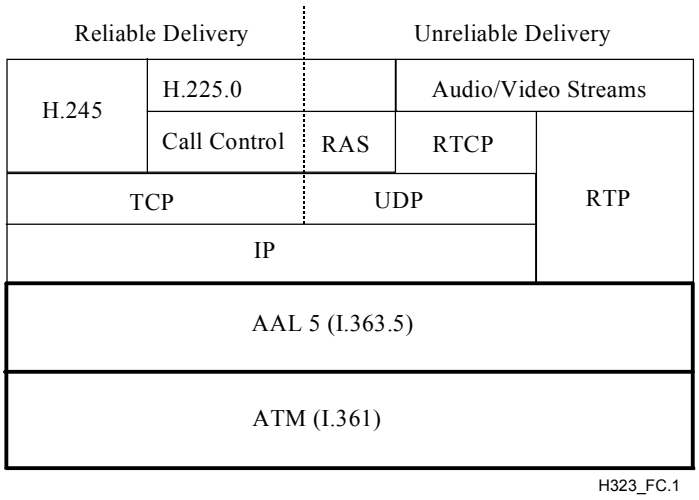


Figure C.1/H.323 – Architecture for H.323 on ATM-AAL 5

C.3.1 Overview of system

The system architecture is designed to make use of H.323 and its component protocols as they are presently specified. It is further designed to use commonly available services of AAL 5 on ATM.

C.3.2 Interoperation with other ITU-T H-series Recommendations endpoints

Interoperation with other H-series endpoints shall be done through the use of Gateway devices as described in ITU-T Rec. H.323. Gateway vendors will need to support the methods described in this annex, if they wish to support the direct use of AAL 5 VCs by H.323 endpoints.

It should be noted that interoperation with other IP-based H.323 endpoints does not require a Gateway.

C.3.3 H.225.0 on IP over ATM

H.225.0 communication requires TCP/IP and UDP/IP using one of the available methods for IP over ATM. No preference is expressed here for which method of IP over ATM to use. If two endpoints on the same network segment use different IP over ATM methods, they must rely on IP routers to forward their packets.

The endpoint shall listen on the well-known TCP ports identified in ITU-T Rec. H.225.0. If the endpoint is being used on a network with a Gatekeeper, the endpoint should use the methods described in ITU-T Rec. H.225.0 to discover and register with the Gatekeeper. This requires the support of UDP multicast. If multicast is not available on the network, the endpoint may be pre-configured with the Gatekeeper(s) address(es).

The methods outlined in ITU-T Rec. H.225.0, combined with an IP over ATM method, shall be used to establish the H.245 control channel on TCP/IP.

C.3.4 H.245 on TCP/IP over ATM

Once the reliable H.245 control channel has been established using methods described in ITU-T Rec. H.225.0, additional channels for audio, video, and data are established based on the outcome of the H.245 capability exchange using H.245 open logical channel procedures.

C.3.5 Addressing for A/V streams

H.323 has the capability for the audio and video streams to be established to a different address than the H.245 control channels. This is fortunate since a TCP/IP channel is established to an IP address, and the audio and video, optionally, are to be sent on RTP over AAL 5 directly to an ATM address.

H.323 also has the capability for the RTCP stream to be addressed separately from the RTP stream. The RTCP stream shall continue to be addressed to an IP address, even though the RTP stream is addressed to an ATM address.

C.3.6 Transport Capabilities added to TransportCapability Set

For operation of H.323 on AAL 5, an addition to the **TransportCapability** set is made in H.245. This includes transport level capabilities such as support for ATM Transfer Capability (DBR, SBR1, SBR2, SBR3, ABT/DT, ABT/IT, ABR) as defined in ITU-T Rec. I.371. Terminals that do not send this new capability parameter shall not make use of the new methods described in this annex. The **TransportCapability** information can be sent as part of the Terminal Capability set exchange in the capability exchange phase. It is also included in the **openLogicalChannel**.

C.3.7 Elements of ATM signalling

C.3.7.1 ATM address

The ATM address for an RTP stream shall be given in the **mediaChannel** subfield of **H2250LogicalChannelParameters** of the H.245 **openLogicalChannelAck** message (or the **OpenLogicalChannel** in the case of Fast Connect). The **mediaChannel** subfield **UnicastAddress** or **MulticastAddress** shall be filled with the 20-octet NSAP-style ATM End System Address.

The use of E.164 for the address is handled by embedding it as the IDP part (AFI = 0x45) of an NSAP address. In this case, an international E.164 number is required.

C.3.7.2 Port Number

The **portNumber** field of the **openLogicalChannel** message is conveyed in the GIT information element as per [34]. The format of the GIT information element is specified in C.4.1.1. This enables the receiving side to associate the ATM VC with the proper RTP logical channel.

For backward compatibility with ITU-T Rec. H.323 Version 2 endpoints, ITU-T Rec. H.323 Version 3 (and later) endpoints shall also be able to use the B-HLI, according to ITU-T Rec. H.323 Version 2 Annex C, for conveying the **portNumber** field of **openLogicalChannel**. An H.323 Version 3 (or later) endpoint shall use the B-HLI only if it has prior knowledge that the terminating endpoint is H.323 Version 2. In cases where the H.323 version of the terminating endpoint is not known, such as establishing a call using Fast Connect, the endpoints shall first attempt to establish the ATM VC using the GIT information element for carrying the **portNumber**. If the connection fails the calling endpoint shall reattempt call setup using B-HLI instead of GIT. If the VC setup with B-HLI also fails, the terminal shall assume that ATM connectivity is not available and shall fall back to using RTP/UDP/IP for media channels. The format of the B-HLI information element is specified in C.4.1.2.

C.3.8 A/V streams on RTP on AAL 5

Servicing the **openLogicalChannel** primitive in H.245 triggers the connection establishment. The audio and video streams are then set up to the destination ATM address. The size of the Maximum Transmission Unit (MTU) shall be signalled in the AAL Parameters information element. The MTU choice may effect system efficiency because of AAL 5 packetization. The packetization rules for AAL 5 are contained in ITU-T Rec. I.363.5. If the non-AAL 5 default of 1536 octets is used, the MTU is packetized in 33 ATM cells and the last AAL 5 cell contains only padding and the AAL 5 number. The address field in the **mediaChannel** should be used to determine whether an ATM VC or a UDP port should be opened.

In the event that the ATM VC setup fails, the endpoint shall retry using RTP/RTCP and the higher layer transport protocol such as UDP.

RTP header compression can optionally be used, as described in section 2 of AF-SAA-0124.000 [33], in which case it must be negotiated using the **mediaTransportType**.

C.3.8.1 Unidirectional logical channels

H.323 has no concept of the reverse direction of a unidirectional logical channel. However, an important characteristic of point-to-point ATM VCs is that they are inherently bidirectional. The use of both directions of an ATM VC is therefore desirable. Otherwise, the audio and video streams will each need to be sent on two different VC's, one for each direction.

Endpoints conforming to this annex are encouraged to open their media streams as bidirectional logical channels. This reduces the number of AAL 5 VCs to two in typical situations, one VC each for audio and for video.

C.3.8.2 Bidirectional logical channels

If the bidirectional usage is indicated, the receiving endpoint shall send an **openLogicalChannelAck** (or the **openLogicalChannel** in the case of Fast Connect) and then it must watch for an ATM VC to be opened by the other endpoint. When an ATM VC is completed, it may then use the reverse direction for the media type indicated in the **openLogicalChannel** command. The endpoint that initiates the **openLogicalChannel** command is the endpoint that shall open the ATM VC.

If QOS is to be used, it shall be limited to the **H2250Capability** declared by the other endpoint. The chosen QOS is signalled as part of the establishment of an ATM VC.

If both endpoints have uncompleted **openLogicalChannel** commands for the same media session, these are resolved using the master/slave methods described in ITU-T Rec. H.245.

C.3.8.3 Maximum transmission unit size

The maximum MTU for AAL 5 is 65 535 octets. As part of **H2250Capability**, the MTU size can be specified in the capabilities exchange during H.245 setup. The forward and backward maximum MTU size shall be equal and will be taken from the smallest of the local and remote values specified in the capabilities exchange.

The MTU size is signalled as the AAL 5 maximum CPCS-PDU size for an ATM VC.

C.3.8.4 RTCP on IP over ATM

It is mandatory to open the logical channel for RTCP traffic on a UDP/IP port, using IP over ATM. RTCP is not permitted to ride directly on an AAL 5 VC.

C.3.9 QOS considerations (Optional)

C.3.9.1 QOS classes defined in ITU-T Rec. I.356

ITU-T Rec. I.356 defines four QOS classes, Class 1 (stringent class), Class 2 (tolerant class), Class 3 (bi-level class), and U class. Table C.1 summarizes the differences among the QOS classes.

Table C.1/H.323 – Provisional QOS class definitions and network performance objectives

	CTD	2-pt CDV	CLR (0+1)	CLR (0)	CER	CMR	SECBR
Default	None	None	None	None	4×10^{-6}	1/day	10^{-4}
Class 1 (stringent)	400 ms	3 ms	3×10^{-7}	None	Default	Default	Default
Class 2 (tolerant)	U	U	10^{-3}	None	Default	Default	Default
Class 3 (bi-level)	U	U	U	10^{-5}	Default	Default	Default
U class	U	U	U	U	U	U	U
CDV: Cell Delay Variation; CER: Cell Error Ratio; CLR: Cell Loss Ratio; CMR: Cell Misinsertion Rate; CTD: Cell Transfer Delay; SECBR: Severely Errored Cell Block Ratio; U: Unspecified/Unbounded.							

C.3.9.2 ATM transfer capability defined in ITU-T Recs I.371 and I.371.1

ATM Transfer Capability (ATC), defined in ITU-T Recs I.371 and I.371.1 as a set of ATM layer parameters and procedures, is intended to support an ATM layer service model and a range of associated QOS classes. Open-loop control ATCs (DBR and SBR) and closed-loop controlled ATCs (ABT and ABR) are specified in ITU-T Recs I.371 and I.371.1. SBR is subdivided into SBR1, SBR2 and SBR3, depending on how to handle CLP = 0/1 cells. ABT is subdivided into ABT/DT and ABT/IT depending on the use of negotiation regarding the block cell rate. Table C.2 summarizes the association of ATCs with QOS classes.

Table C.2/H.323 – Association of ATCs with QOS classes (from Table 3/I.356)

ATM Transfer Capabilities (ATC)	DBR, SBR1, ABT/DT, ABT/IT	DBR, SBR1, ABT/DT, ABT/IT	SBR2, SBR3, ABR	Any ATC
Applicable QOS class	Class 1 (stringent)	Class 2 (tolerant)	Class 3 (bi-level)	U class
ABR: Available Bit Rate; ABT/DT: ATM Block Transfer/Delayed Transmission; ABT/IT: ATM Block Transfer/Immediate Transmission; DBR: Deterministic Bit Rate; SBR1: Statistical Bit Rate configuration 1; SBR2: Statistical Bit Rate configuration 2; SBR3: Statistical Bit Rate configuration 3.				

C.3.9.3 Broadband transfer capability defined in ITU-T Rec. Q.2961.2

Broadband Transfer Capability (BTC) codes (DBR, BTC5, BTC9, BTC10 and SBR1) in Broadband bearer capability information element are defined in ITU-T Rec. Q.2961.2, and valid combinations of bearer class, broadband transfer capability and ATM traffic descriptor parameters are specified in Annex A/Q.2961.2. In the Setup message, the user can specify the BTC according to the traffic he/she generates and the intended use of network services. In Table A.1/Q.2961.2, 3 valid combinations are listed for bearer class BCOB-A, 8 combinations for BCOB-C and 13 combinations for BCOB-X or FR.

C.3.9.4 Opening of Virtual Channels

The endpoint that originated the accepted **openLogicalChannel** is responsible for opening the ATM VC. Support for QOS in the ATM VC is signalled at the time it is established. If successful, the ATM network provides a guaranteed QOS for the lifetime of the opened VC. QOS is specified in terms of Q.2931 Information Elements (IEs), including ATM Traffic Descriptor and Broadband Bearer Capability.

C.3.9.5 Use of DBR

The most likely available ATM traffic type is a constant bit rate using DBR. The use of DBR is signalled as part of the ATM broadband Bearer Capability IE (Bearer class = "BCOB-A"). Use of other ATM traffic type, such as SBR with end-to-end timing required [Bearer class = "BCOB-X" and BTC field = "SBR1 (0010011)"], is also possible.

C.3.9.6 Setting the proper cell rate

It is important to set the proper cell rate parameters in the ATM Traffic Descriptor information element. The peak cell rate can be derived from the H.245 capabilities exchange parameters and the RTP payload format packet size. For video, the **maxBitRate** field can be used from the **H261VideoCapability** or the **H263VideoCapability** to determine the ATM Cell rate. For audio, the audio capability chosen implies the bit rate to be used. For example, the use of **g711Ulaw64k** suggests the use of a 64 kbit/s audio channel, while the use of **g728** indicates the use of a 16 kbit/s channel. The RTP payload format indicates the packet size. For each packet, the subsequent AAL packet overhead and any needed padding to meet the AAL packetization rules must be added. This results in an overhead bit rate that is associated with the size of the packet and the way this packet is encapsulated in the AAL and the frequency of this overhead from this encapsulation.

The bit rate of the data to be sent and the packetization of the data according to the AAL packetization rules determine the cell rate. The packetization will determine the actual number of cells that must be sent for a given data stream at a given bit rate. The choice of MTU can affect the packetization as explained in C.3.8.

C.4 Protocol section

C.4.1 ATM signalling information elements

C.4.1.1 Generic information transport

IE Parameter	Value	Notes
Identifier related standard/application (octet 5)	00001011	ITU-T Rec. H.323
Identifier Type (octet 6)	00001011	H.245 **portNumber**
Identifier length (octet 6.1)	0000 0010	2 octets
Identifier value (octets 6.2-6.3)	H.245 **portNumber**	16-bit binary coded forward H.245 **portNumber**

H.323 Version 3 (or later) endpoints shall set the IE action indicator of the GIT information element to "clear call", according to 4.5.1/Q.2931. In this case, if the terminating endpoint does not support GIT information element coding it will reject the call with the cause value 100 for *Invalid information element content* according to 5.7.2/Q.2931. If the ATM VC setup attempt is rejected because the terminating endpoint does not understand GIT it will reject the VC call setup with cause number 99 *Information element non-existent or not implemented*, according to 5.7.2/Q.2931.

It should be noted that the **portNumber** field in H.245 is only 16 bits in length.

The H.245 **portNumber** is used by the receiving endpoint to associate the ATM VC with the proper RTP logical channel. The endpoint that initiates the **openLogicalChannel** command is the endpoint that opens the ATM VC. It is possible for the initiating endpoint to select an H.245 **portNumber** that is already in use by the receiving endpoint. This would cause a failure in the OLC procedure.

Additionally the receiving RTCP port is also specified by the initiating endpoint by implication. H.323 states that the corresponding RTCP data shall flow on a UDP port number equal to the H.245 **portNumber** plus 1. It is possible that the resulting port number for RTCP, H.245 **portNumber** plus 1, will be in use on the receiving endpoint since the H.245 **portNumber** is selected by the initiating endpoint.

Due to the above problems the receiving endpoint should have the choice of selecting the H.245 **portNumber**. If the **portNumber** is not specified in the **openLogicalChannel** the receiving endpoint shall specify a **portNumber** in the **openLogicalChannelAck** message (or **openLogicalChannel** in the case of Fast Connect). It is recommended that the transmitting endpoint does not specify the **portNumber** in the **openLogicalChannel** thereby requiring the receiving endpoint to specify one in the **openLogicalChannelAck** message (or **openLogicalChannel** in the case of Fast Connect).

The **portNumber** field of the **openLogicalChannel** message is used to select the H.245 **portNumber**. The receiving endpoint uses this H.245 **portNumber** to associate the ATM VC with the proper RTP logical channel. If the receiving endpoint finds that the given H.245 **portNumber** is inappropriate it can select a new H.245 **portNumber** and use the **portNumber** field of the **openLogicalChannelAck** message (or **openLogicalChannel** in the case of Fast Connect) to indicate the new value to the initiating endpoint. The selected H.245 **portNumber** field is conveyed in the GIT information element. This enables the receiving side to associate the ATM VC with the proper RTP logical channel.

The VC association port number is represented in network byte order in octets 6.2 and 6.3 of the GIT (i.e., octet 6.2 holds the MSB and octet 6.3 holds the LSB).

C.4.1.2 Broadband High Layer Information

IE parameter	Value	Notes
Length of B-HLI contents (octets 3-4)	3	
High layer information type (octet 5)	"0000 0001"	User-specific
High layer information (octets 5-7)	H.245 **portNumber**	16-bit binary coded forward H.245 **portNumber**

The B-HLI is only used for backward compatibility with H.323 Version 2 endpoints, as described in C.3.7.2.

C.4.1.3 ATM Adaptation Layer parameters

IE parameter	Value	Notes
AAL type (octet 5)	"0000 0101"	AAL 5
Forward maximum AAL 5 CPCS-SDU size (octets 6.1-6.2)	MTU size	The smaller **mTUsize** in the local and remote **QOSCapability.atmParms**
Backward maximum AAL 5 CPCS-SDU size (octets 7.1-7.2)	MTU size	Same as forward
SSCS type (octet 8.1)	"0000 0000"	Null SSCS

C.4.1.4 ATM Broadband bearer capability Information Element

a) In the case where the ATM traffic type in ITU-T Rec. H.245 is equal to "DBR":

IE parameter	Value	Notes
Bearer class	BCOB-A	
Susceptibility to clipping	Susceptible to clipping	
User-plane connection configuration	Point-to-point	

b) In the case where ATM traffic type in ITU-T Rec. H.245 is equal to "SBR1" with end-to-end timing required:

IE parameter	Value	Notes
Bearer class	BCOB-X	
Broadband bearer capability	"0010011" (SBR1)	SBR1 with end-to-end timing required
Susceptibility to clipping	Susceptible to clipping	
User-plane connection configuration	Point-to-point	

C.4.2 H.245 usage

The establishment of a H.323 call using AAL 5 media streams is done in a manner similar to the basic mode of H.323 on IP. The difference is that the completed **openLogicalChannel** exchange in H.245 should result in an AAL 5 VC being established. This is illustrated in Figures C.2 and C.3 for the unidirectional VC usage and the bidirectional VC usage, respectively.

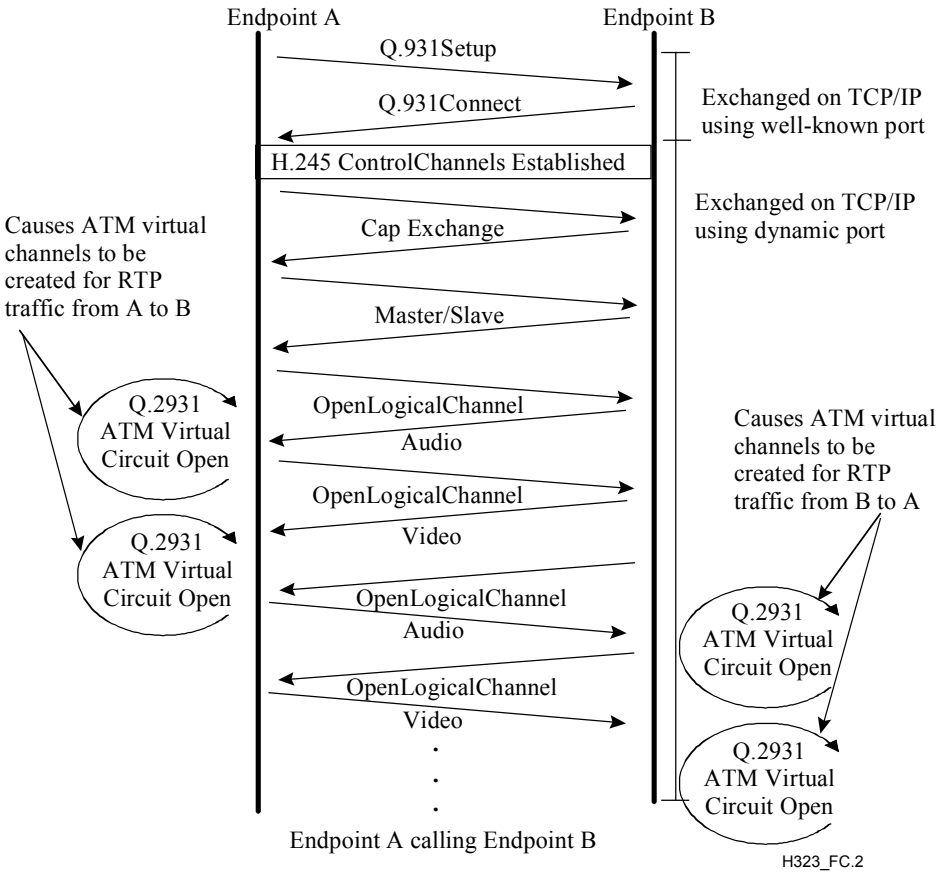


Figure C.2/H.323 – H.323 call establishment showing ATM effect – ATM VC's used unidirectionally

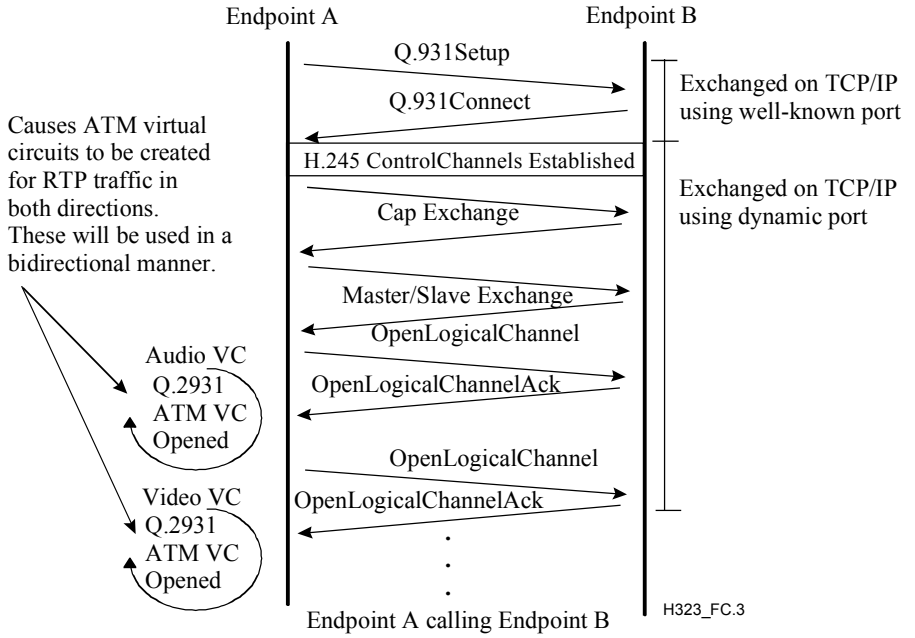


Figure C.3/H.323 – H.323 call establishment showing ATM effect – ATM VC's used bidirectionally

It should be noted that the ATM VC setups will occur in only one direction if bidirectional logical channels are used. In this case, the endpoint acknowledging the **openLogicalChannel** will merely bind the incoming ATM connection to an RTP session using the VC Association port number.

C.4.3 RTP usage

RTP and RTCP are defined in Annex A/H.225.0. RTCP is currently required for all H.323 connections and therefore is required even when using an AAL 5 VC. The RTCP is carried by UDP/IP, not directly by the AAL 5 VC.

C.4.4 Interoperation with H.323 on IP

Since the H.225.0 and H.245 communications are on IP, the endpoint will be able to receive calls from any other endpoint that is properly connected to the IP network. It is possible that H.323 endpoints will be used on ATM that do not support the methods described in this annex. They will strictly follow the basic method of using UDP/IP for the A/V streams. In this case, the endpoint will not declare the new **transportCapabilities** in H.245 and will refuse to open logical channels using ATM addressed VCs.

The protocol to **openLogicalChannel** using AAL 5 VCs for A/V streams should only be used if the received Capabilities have indicated that the method of this annex is supported. If this capability parameter is not present in the Terminal Capability Set, then the endpoint should only use **openLogicalChannel** using UDP/IP over ATM. This will ensure that the endpoint can communicate with other endpoints that support H.323, but may not support the methods in this annex.

Annex D

Real-time facsimile over H.323 systems

D.1 Introduction

Currently, facsimile and speech are typically sent using the PSTN with the same calling and addressing infrastructure. It is highly desirable to continue this approach in the context of this Recommendation. From a high level, facsimile can be viewed as another kind of real-time traffic similar to a particular speech coder. This seems appropriate, as facsimile entering the packet world via a gateway from the PSTN should logically be treated in a fashion similar to speech if the customer expects a real-time, assured end-to-end transmission service. The conversion of facsimile to email or other store-and-forward methods represents a new service that is beyond the scope of this Recommendation, which is a real-time protocol. It is recognized that manufacturers may wish to provide a gateway that falls back to a store-and-forward service when the real-time facsimile call fails. It is beyond the scope of this Recommendation when and how this decision is made, or by what means a store-and-forward facsimile service is implemented.

ITU-T Rec. T.38 [56] defines an Internet facsimile protocol consisting of messages and data exchanged between Facsimile Gateways connected via an IP network. This annex uses ITU-T Rec. T.38. Communication between the Gateways and G3/G4 Facsimile terminals is beyond the scope of ITU-T Rec. T.38. The reference model for T.38 is shown in Figure D.1 with three scenarios. In the first scenario, the two traditional Group 3 Facsimile Equipment (G3FE) terminals are virtually connected through the Gateways once the PSTN calls are established. All T.30 [55] session establishment and capabilities negotiation is carried out between the terminals. In the second scenario, the traditional Group 3 Facsimile (IAF) terminal is connected with an Internet Aware Fax terminal (IAF).

The IAF is directly connected to the IP network. In the third scenario, the two IAFs are directly connected to the IP network. In all the scenarios, T.38 packets are used on the IP network to communicate T.4/T.30 facsimile information. The transport of T.38 packets is either on TCP/IP or UDP/IP using the H.323 mechanism.

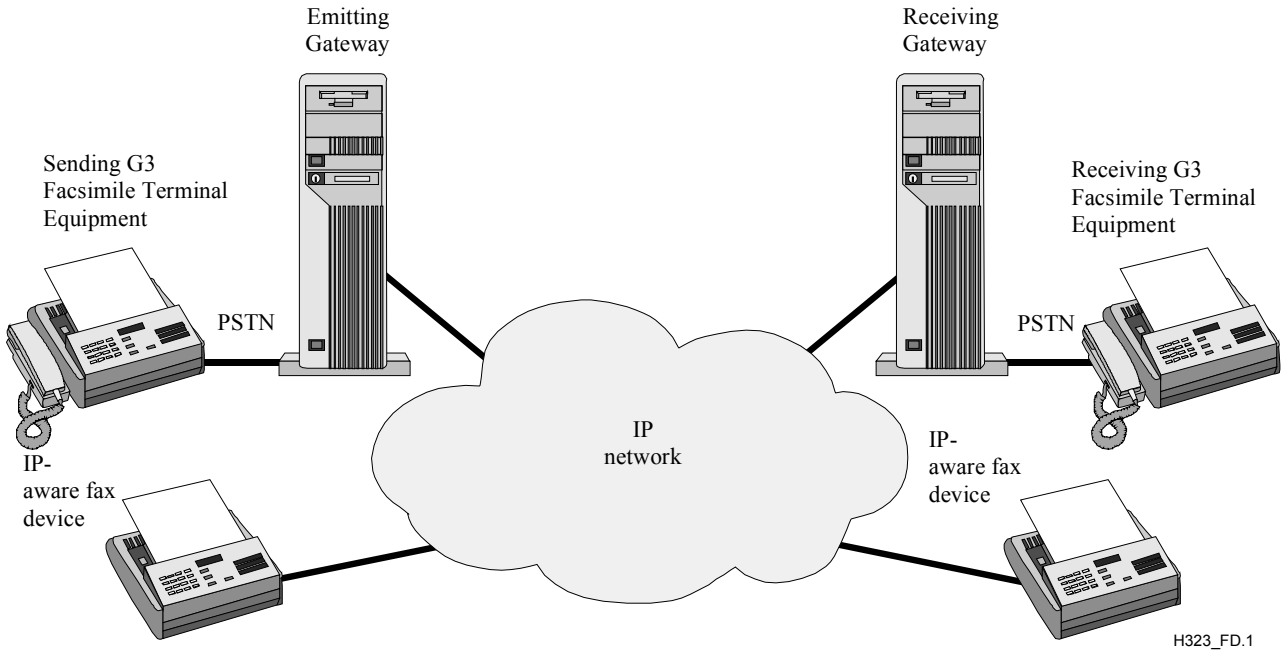


Figure D.1/H.323 – Model for facsimile transmission over IP networks

D.2 Scope

The scope of this annex is to use H.323 procedures to transfer T.38 packets in real time over the IP network. H.323 entities supporting facsimile capabilities shall use T.38 to support real-time facsimile services as described in this annex.

H.323 facsimile capable endpoints shall support the usage of TCP and UDP as described in ITU-T Rec. T.38. Annex B/T.38 describes a T.38-only capable terminal that supports a subset of H.245 messages using H.245 tunnelling. However, the T.38/Annex B terminal can interwork with an H.323/Annex D terminal using 8.1.7/H.323 "Fast Connect Procedure", and 8.2.1/H.323 "Encapsulation of H.245 Messages within H.225.0 call signalling Messages" procedures in this Recommendation. T.38/Annex B terminals interwork with H.323 terminals without being conformant to this Recommendation. An H.323 terminal that supports the procedures of this annex shall interwork with T.38/Annex B terminals.

D.3 Procedures for opening channels to send T.38 packets

Fast Connect is used to describe the H.323 procedures for opening channels for the transportation of T.38 packets. The traditional sequence can also be used, though it is not described here.

D.3.1 Opening the voice channel

Zero, one (sender to receiver channel or receiver to sender channel), or two (sender to receiver channel and receiver to sender channel) logical channels for voice may be opened depending on the capability of the sender and the receiver. If a voice channel is desired, the voice channel shall be opened as specified by the procedures in 8.1.7/H.323 "Fast Connect". Support of voice by facsimile applications is not mandatory in this annex.

D.3.2 Opening the facsimile channels

Two unidirectional reliable or unreliable logical channels (sender to receiver channel and receiver to sender channel) as shown in Figure D.2 or, optionally, one bidirectional reliable channel as shown in Figure D.3 shall be opened for the transfer of T.38 packets. T.38 packets can be transferred using either TCP or UDP. In general, the usage of TCP is more effective when the bandwidth for facsimile communication is limited. On the other hand, the usage of UDP may be more effective when the bandwidth for facsimile communication is sufficient.

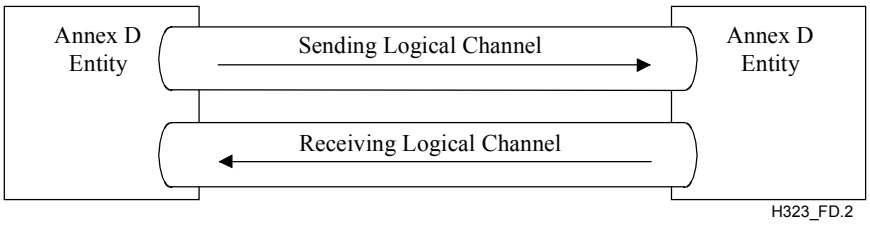


Figure D.2/H.323 – A pair of unidirectional channels

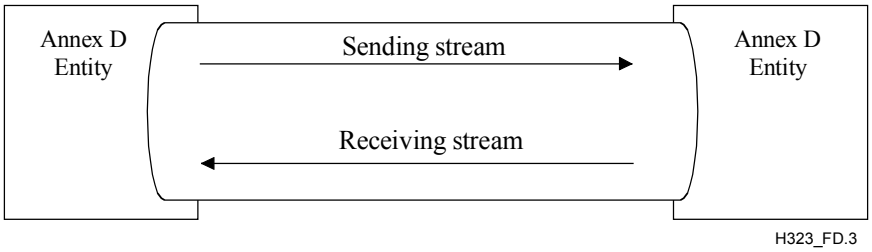


Figure D.3/H.323 – A unit of bidirectional channels

NOTE – In the first version of this annex, it was not possible to use a single bidirectional reliable channel. In order to retain backward compatibility, the endpoint may specify support for bidirectional reliable channels by including the **t38FaxTcpOptions** SEQUENCE and setting the **t38TCPBidirectionalMode** field to TRUE. If the other endpoint does not include the **t38FaxTcpOptions** SEQUENCE, the endpoint shall assume that a single bidirectional reliable channel for T.38 is not supported and shall use either two unidirectional reliable or unreliable channels.

The sender terminal specifies a TCP/UDP port in the **OpenLogicalChannel** in the **fastStart** element of *Setup*. The receiver terminal shall provide its TCP (or UDP) port in the **OpenLogicalChannel** of the **fastStart** element as specified by the procedures in 8.1.7/H.323 "Fast Connect procedure".

The receiver shall open the TCP/UDP port based on the preference of the sender. If the sender terminal has a preference for UDP or TCP, then it shall indicate its preference by ordering proposals in the **fastStart** sequence according to 8.1.7.1/H.323. The receiving terminal can select the transport, TCP or UDP, by returning the desired proposals in **OpenLogicalChannel** structures in the **fastStart** element of *Connect*.

Figures D.4 and D.5 show the signalling used to open unidirectional and bidirectional channels using Fast Connect.

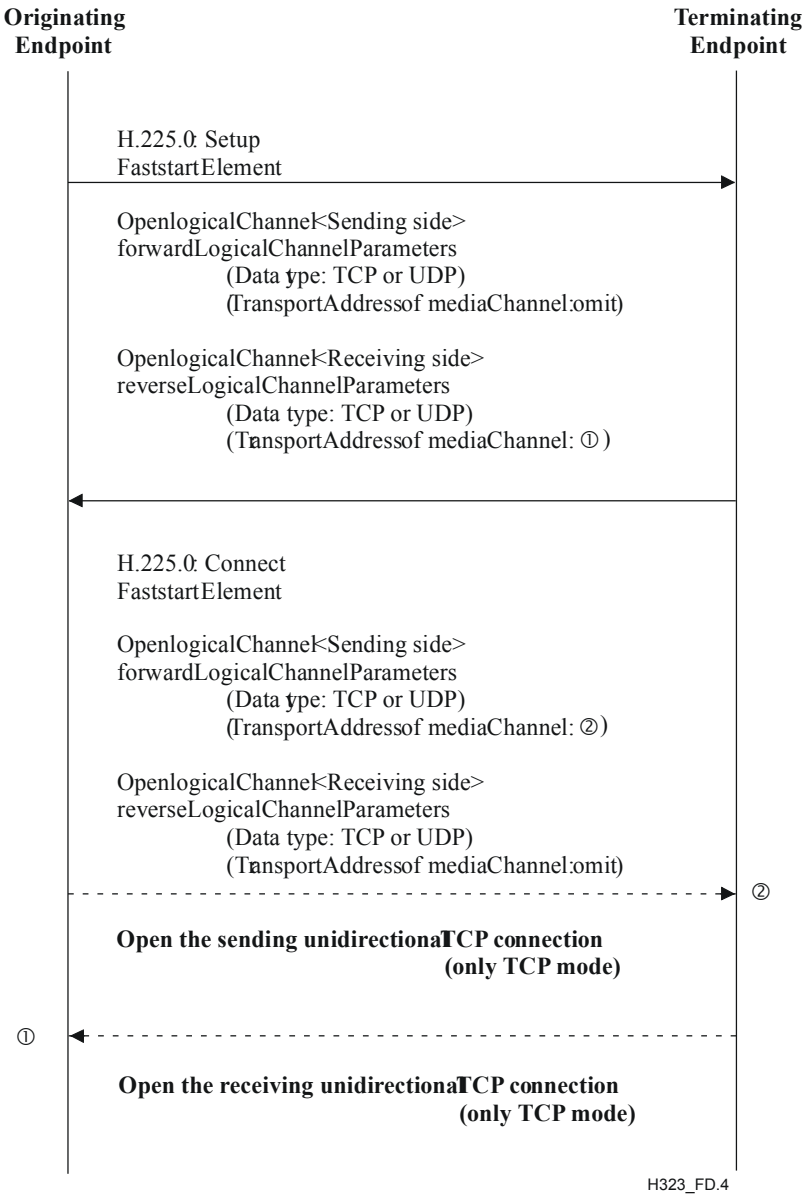


Figure D.4/H.323 – Two unidirectional channels with fast connect

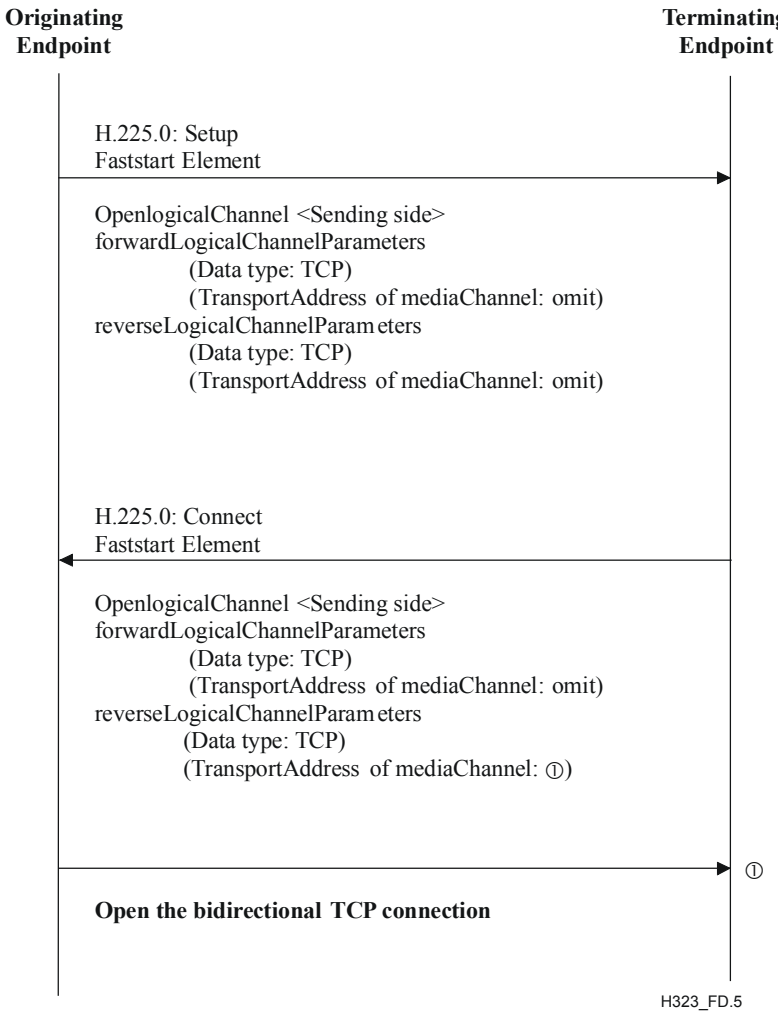


Figure D.5/H.323 – One bidirectional reliable channel with fast connect

D.3.3 DTMF transmission

DTMF tones shall be sent by H.323/Annex D terminals using **UserInputIndication** to interwork with T.38/Annex B terminals. H.323/Annex D terminals may send DTMF tones in-band with the voice when T.38/Annex B terminals are not involved in the call.

D.4 Non-Fast Connect procedures

It is noted that in Non-Fast Connect, the normal H.245-based **OpenLogicalChannel** procedures can be used to open and close both UDP and TCP fax channels (refer to 6.2.8.2/H.323). Tunnelled H.245 can also be used to open and close channels. It is also noted that non-Fast Connect and non-tunnelled H.245 procedures do not apply to interworking with ITU-T Rec. T.38.

Figures D.6 and D.7 show the signalling used to open unidirectional and bidirectional channels when not using Fast Connect.

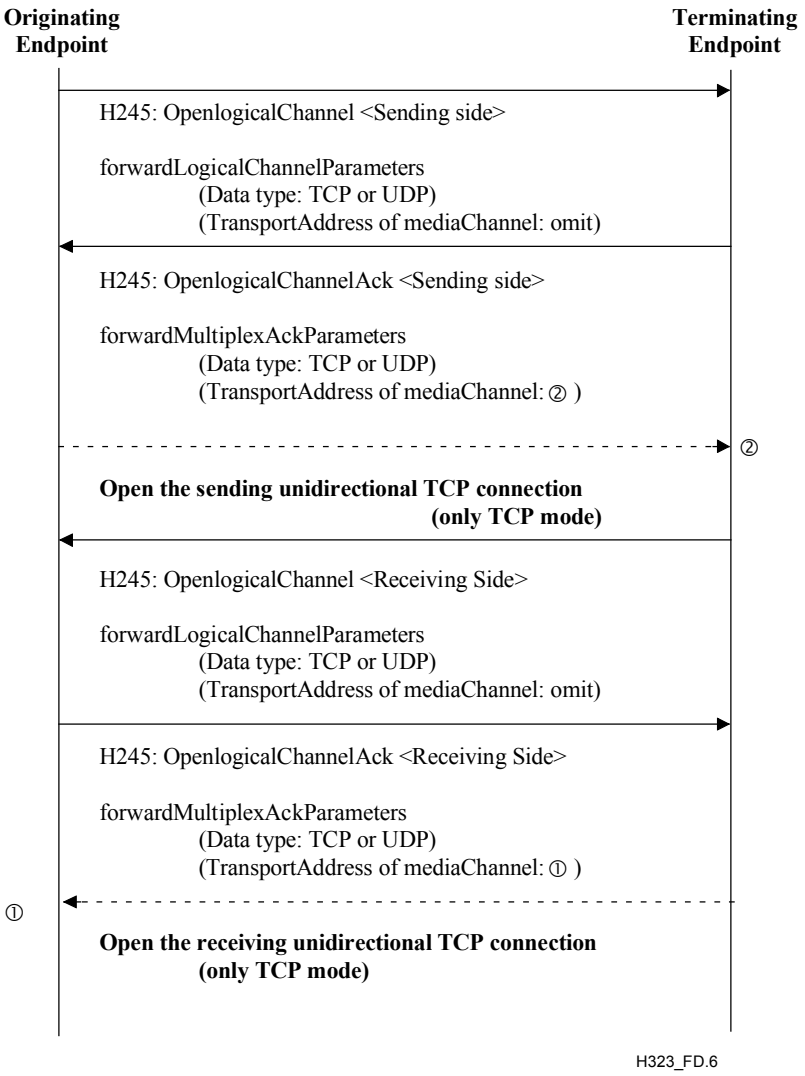


Figure D.6/H.323 – Two unidirectional channels without fast connect

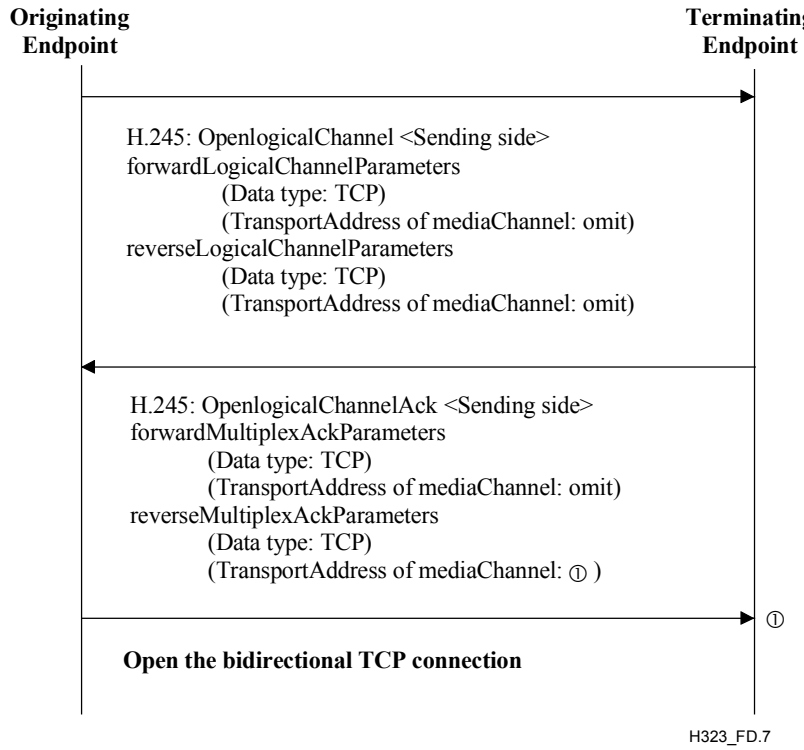


Figure D.7/H.323 – One bidirectional channel without fast connect

D.5 Replacing an existing audio stream with a T.38 fax stream

An endpoint that wishes to replace an existing audio stream with a fax stream shall use the following mechanism to achieve this goal.

Once the audio call has been established – ideally via the use of Fast Connect and prior to the receipt of the CONNECT message – the endpoint that wishes to replace the audio stream with T.38 fax shall initiate H.245 procedures via tunnelling if H.245 has not already been started.

During H.245 capability exchange, each endpoint shall express its capability of receiving and transmitting T.38 fax by including the **t38fax** field of the **DataApplicationCapability** structure. The presence of this field indicates that the remote endpoint is capable of supporting the T.38 Fax Mode.

It should be noted that the Connect message may arrive while H.245 procedures are taking place. After H.245 procedures have completed and the Connect has been received, either endpoint may detect fax tones (i.e., CNG or CED) or the presence of V.21 carrier and HDLC flags. Typical scenarios for facsimile call detection rely on the analysis of CNG calling tone and a response of the CED answer tone and/or the initiation of fax procedures using the V.21 carrier and HDLC flags. Note that in some implementations the presence of either CNG or CED are optional. Therefore, both endpoints should take an active role in order to properly detect fax.

When using two unidirectional fax channels, the endpoint that detected the tone shall initiate the standard H.245 Mode Request procedure by sending a **requestMode** message to its remote counterpart with the **t38fax** data mode as the requested mode. The endpoint that receives the **RequestMode** message shall return a **requestModeAck** message. On receiving the **requestModeAck** message, the initiating endpoint shall close its audio logical channel and open a T.38 logical channel. Similarly, the remote end shall close its audio logical channel and open a T.38 fax logical channel. After acknowledgments have been received for each of the T.38 open logical channels, fax transmission and reception takes place.

Figure D.8 illustrates a successful switchover from voice to fax when a separate H.245 channel is already open for two unidirectional media channels.

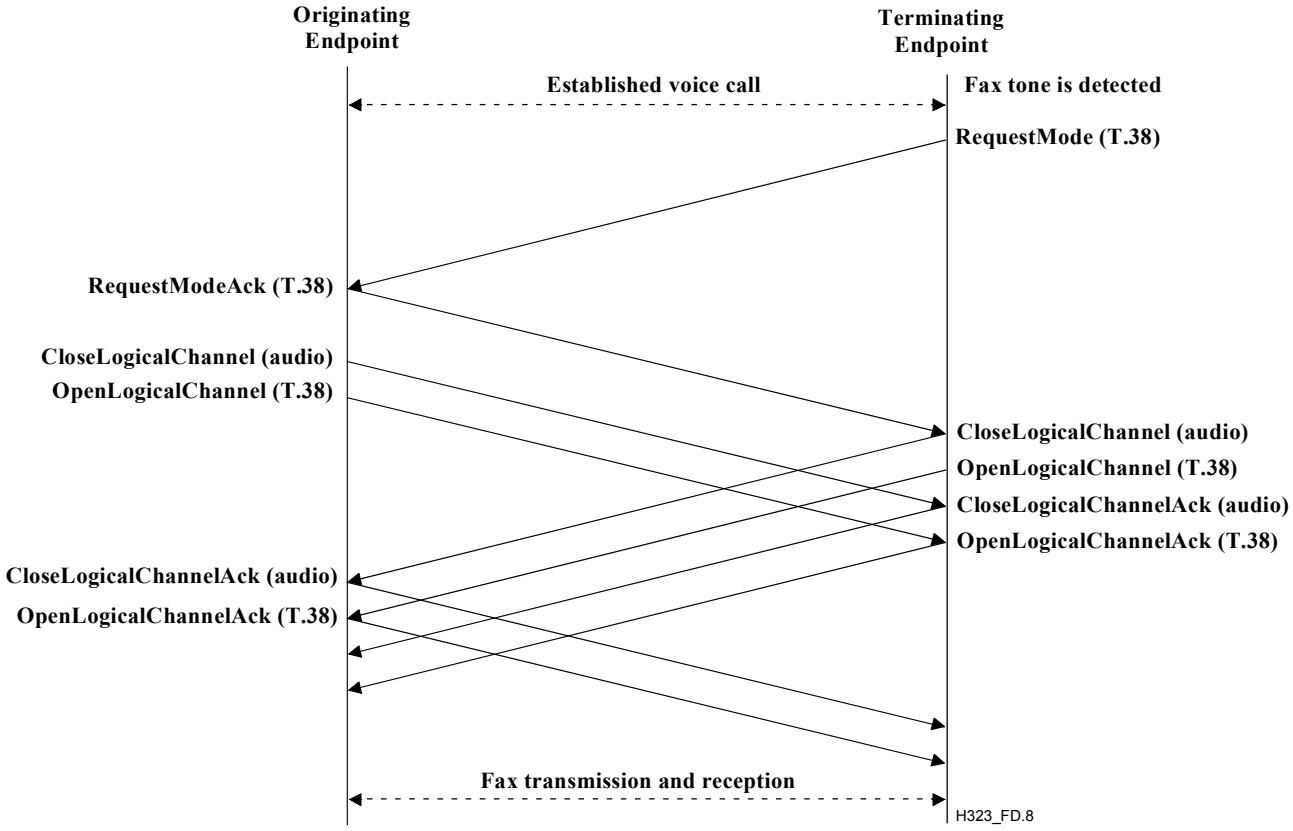


Figure D.8/H.323 – Successful switching of an existing voice call to T.38 using two unidirectional media channels without tunnelling

Figure D.9 illustrates a successful switchover from voice to fax using H.245 tunnelling for two unidirectional media channels.

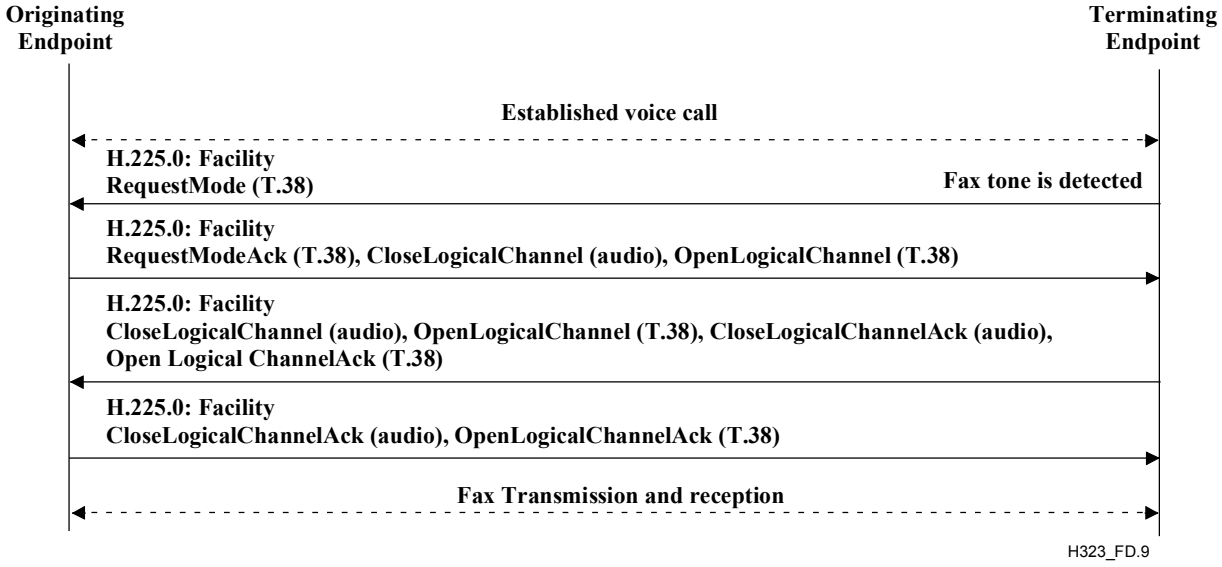


Figure D.9/H.323 – Successful switching of an existing voice call to T.38 using two unidirectional media channels with tunnelling

When using a bidirectional fax channel (for TCP only), the request mode command is not necessary: the endpoint that detected the tone shall close its open channels, request the reverse channels to be closed by the other endpoint, and open a bidirectional T.38 channel. Upon reception of the request channel close command, the remote end shall close its audio channel. After acknowledgements have been received for each of the T.38 open logical channels, fax transmission and reception takes place.

Figure D.10 illustrates a successful switchover from voice to fax when a separate H.245 channel is already open for one bidirectional media channel.

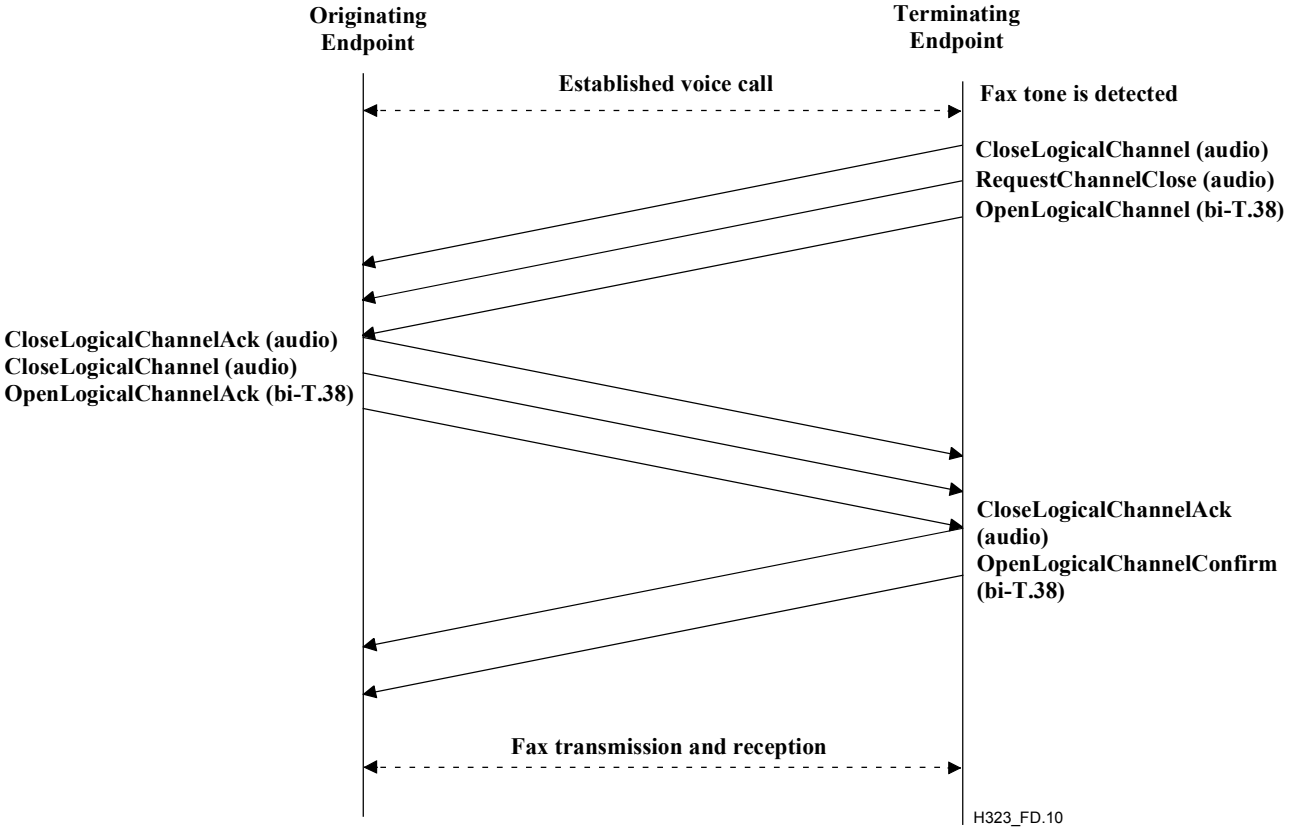


Figure D.10/H.323 – Successful switching of an existing voice call to T.38 using one bidirectional media channel (TCP) without tunnelling

Figure D.11 illustrates a successful switchover from voice to fax using H.245 tunnelling for one bidirectional media channel.

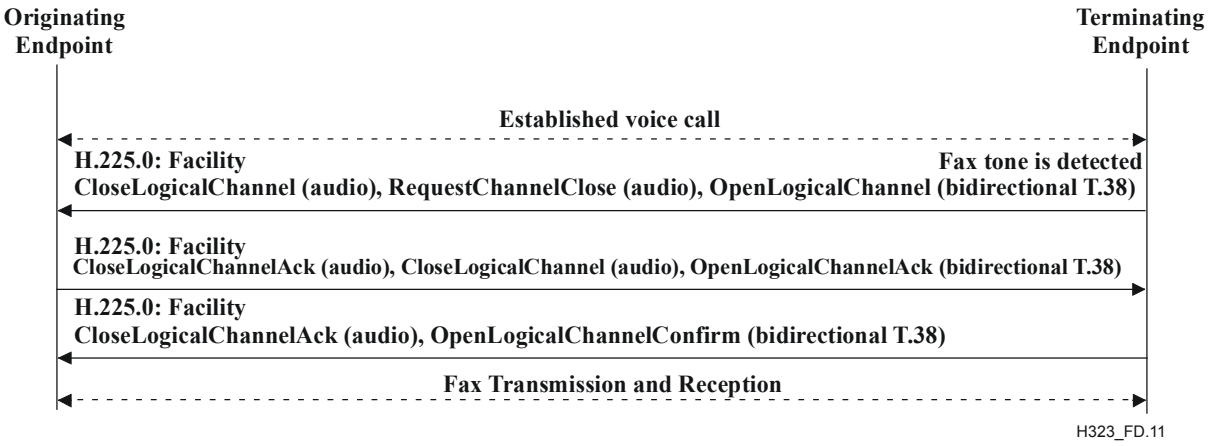


Figure D.11/H.323 – Successful switching of an existing voice call to T.38 using one bidirectional media channel (TCP) with tunnelling

Should either endpoint wish to return to an audio call after the fax transmission has ended, the Mode Request procedure shall be initiated using an audio codec as a parameter. The above procedure also applies to traditional "slow start" cases, in the event that Fast Connect cannot be established between the two endpoints.

D.6 Usage of the MaxBitRate in messages

When TCP is used for T.38 fax transmission, **maxBitRate** in the ARQ/BRQ does not include the fax data rate, and if a voice link is switched off when the fax session starts, a BRQ shall be used to indicate to the Gatekeeper that the bandwidth has changed. When UDP is used for T.38 fax transmission, **maxBitRate** in the ARQ/BRQ does include the bit rate needed for the fax session. The endpoint (terminal, gateway) shall send BRQs to the Gatekeeper as bandwidth needs change during the call. It is noted that the **maxBitRate** in the **OpenLogicalChannel** element in the Setup during Fast Connect is different from the **maxBitRate** in ARQ/BRQ, and does refer to the peak bit rate that the fax call will use.

D.7 Interactions with gateways and T.38/Annex B devices

The following case must be considered:

H.323/Annex D device (with voice) <–> T.38/Annex B device (without voice).

Note that these devices may be terminals or gateways; it does not affect the discussion. A fax call arrives from the "voiceless" side, but the voice side must generate an outgoing voice call that is not connected to anything although tones or announcements might be played. In the opposite direction, the H.323/Annex D device cannot offer a voice call to a "voiceless" device, as it cannot receive voice.

The H.323/Annex D gateway may send both a voice and fax **OpenLogicalChannel** element in the Setup message. If it encounters a T.38 device, only the fax channel will be opened if both were proposed. If the call mistakenly encounters a non-fax H.323 device, the fax port will not be opened. This is the equivalent of a fax machine calling a telephone.

An H.323/Annex D device becomes aware that it is talking to a T.38/Annex B device due to the following sequence of events:

1) The T.38/Annex B device does not supply an H.245 port in the Connect or Setup.

2) The H.323/Annex D device uses the Facility message as described in 8.2.3/H.323 and transmits a **FACILITY** message with a **FacilityReason** of **startH245** and provides its H.245 address in the **h245Address** element. The T.38/Annex B endpoint receiving a **FACILITY** message with a **FacilityReason** of **startH245** will respond with a **FACILITY** message having a **FacilityReason** of **noH245**. At this point the H.323/Annex D device should cease all attempts to open the H.245 channel.

Annex E

Framework and wire-protocol for multiplexed call signalling transport

E.1 Scope

This annex describes a packetization format and a set of procedures (some of which are optional) that can be used to implement UDP and TCP based protocols. The first part of this annex describes the signalling framework and wire-protocol, and subsequent clauses detail specific use cases. The only profile currently specified in this revision is for transporting H.225.0 call signalling messages.

This annex is designed to operate in engineered networks and use the security services provided by H.323 (e.g., H.235, IPSec). This annex should not be used over the public Internet, due to security and traffic considerations.

E.1.1 Introduction

E.1.1.1 Multiplexed transport

This annex provides a multiplexed transport layer that can be used to transmit multiple protocols (with optional reliability) in the same PDU. Often-used protocols have specific code points (also called "payload types"). Other protocols can be carried and identified using the ObjectID-typed payloads.

E.1.1.2 Multiple payloads in a single PDU

Annex E PDUs can contain multiple "payloads", each a different protocol and targeted at a different session (the definition of a "session" is protocol dependent). Note that there is no implicit relation between payloads when they arrive in the same PDU.

E.1.1.3 Flexible header options

Annex E PDU and Payload headers are configurable. Minimum header size can be as small as 8 octets, and may grow up to 20 octets when all optional fields are present.

E.1.1.4 Ack message

Messages carried using UDP can get lost. If the application needs assurance that a sent message arrived successfully, it may request an Ack message for the PDU.

A sender shall specify in the <ackRequested> field whether it wants to receive an Ack message for a PDU being sent, and the receiver shall reply with an Ack Payload if the <ackRequested> field is set.

NOTE – Ack messages shall be sent by the Annex E transport layer, not by the application using the Annex E stack. The specific Ack behaviour is mandated by the signalling model the Annex E stack is instructed to use by the application.

E.1.1.5 Nack message

A Nack message shall be used to signify some error condition. Such errors may be the inability to support a specific payload type, the arrival of a malformed PDU, and others. These messages may or may not have the effect of dropping an ongoing call.

NOTE – Nack messages are be sent by the Annex E transport layer, not by the application using the Annex E stack.

E.1.1.6 Sender sequence number policy

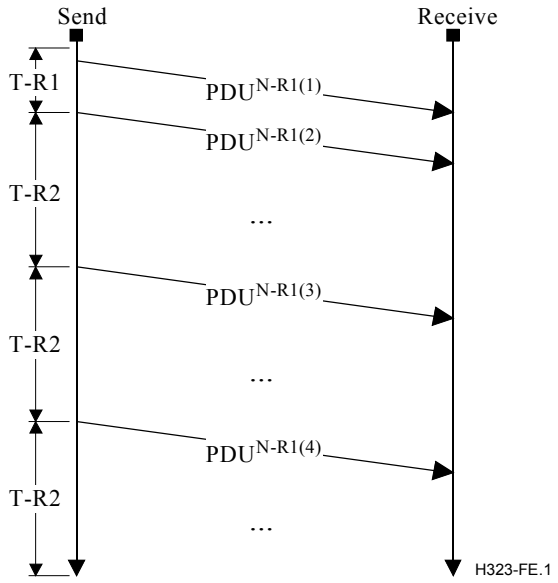
Assigned per host-address and source-port, sending Annex E layers shall start with some random value, incrementing by 1 for every PDU sent. If the sequence number reaches 2^{24} (16 777 216) it shall wrap around to 0.

E.1.1.7 Receiver sequence number policy

When receiving a UDP packet, the Annex E layer shall check the host-address, source-port and sequence number to recognize duplicate messages. The Annex E layer may reorder messages according to sequence numbers and recognize packet-loss when finding gaps in sequence numbers.

E.1.1.8 Retransmissions

When messages get lost (and an Ack was requested and not received) the sender may retransmit the message. The retransmission policy attempts to combat first-message lost by retransmitting quickly, but if that message is lost too, the sender is required to backoff the retransmission delay by a factor of more than two. See Figure E.1.



Retransmission timers and counters:

Item	Value	Comments
T-R1	500 ms	A reasonably small value is chosen here to compensate for possible 1st packet loss
T-R2	(T-R1 | T-R2) × N-R2	If the first retransmitted packet is lost, apply some back-off. If a previous T-R2 value is available, use it instead of the initial value (T-R1).
N-R1	8	Maximum number of retransmissions before abandoning the connection
N-R2	2.1	Multiplier to be used for back-off

Figure E.1/H.323 – PDU retransmission

When there is a known round-trip message interval value from a previous transmission, timer T-R1 should be set to that round-trip message interval value +10%.

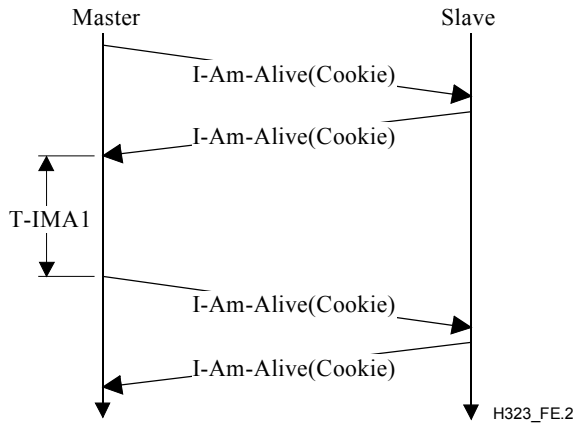
E.1.1.9 Connection keep-alive

When running over TCP, the presence of a persistent TCP connection can ensure that one side is aware of the remote side failures (by observing TCP failures). When running over UDP, there is no such "state" associated, and another procedure must be used.

The solution is for one side of the call (usually the "server" or "master" side if such classification is relevant) to send an "I-Am-Alive" message to the other side, to let the remote application know the host is still up. The remote side will answer with an I-Am-Alive message of its own as proof that it too is up. A cookie may be provided by the originator of an I-Am-Alive sequence, and if made available, shall be returned in the answer I-Am-Alive.

The retransmission timer of the I-Am-Alive messages may be reset on receiving other relevant message, as it is proof the remote end is alive. This saves bandwidth, as I-Am-Alive messages will be sent only when really needed. This capability is decided on a per-protocol basis.

Generating I-Am-Alive messages is optional, however, all entities shall support the ability to reply to I-Am-Alive messages (e.g., the ability and requirement to answer an I-Am-Alive message is not optional, and when such a message is received, it shall be answered according to the procedures defined in this annex). See Figure E.2.



I-Am-Alive timers

Item	Value	Comments
T-IMA1	6 seconds	I-Am-Alive transmission interval[a)]
N-IMA1	6	Number of consecutive I-AM-ALIVE messages not responded to after which the remote peer is declared dead
[a)] These timers should follow the recommended values in Annex R if Annex R is also being used between the two entities.		

Figure E.2/H.323 – I-Am-Alive transmission

E.1.1.10 Forward error correction

Annex E messages may be sent more than once to enable forward error correction. If the arrival of a message is crucial, the Annex E layer may choose to send the same message twice (without incrementing the sequence number). If both messages arrive, the second one will be treated as normal message duplication.

E.1.1.11 Reply hints

It is advisable for Annex E implementers to add a slight delay before an Ack message is sent back, to allow the application to attach a protocol payload to accompany the Ack payload. A Header option is available to allow senders to Hint to the remote transport layer that a reply is expected for a given message.

NOTE – For example, when a H.225.0 SETUP message is sent, the stack can delay the reply of the Ack payload slightly when the ReplyHint bit is set to ensure the application will have time to provide the return CONNECT payload (for example). The returning PDU will then contain both an Ack (for the SETUP) and the CONNECT payload.

E.1.1.12 Well-known port and port spawning

This annex supports one main well-known port UDP/TCP port 2517. Applications supporting Annex E operations when receiving a payload that the main well-known port does not support (identified either using the static payload type or the object-ID payload type) may reply with a Nack message that instructs the sender to send this specific payload type to a different port and IP address.

E.1.2 Signalling models

Signalling may follow many models. Each protocol implementation using this annex shall support one of the models (as described below) or choose a different signalling model that suits its requirements.

E.1.2.1 Real-time model

In the real-time model, if a PDU is lost, there is no use to re-send the PDU as the information may already be irrelevant. An example of such a protocol is RTP when used for real-time audio or video streaming. For such protocols the delay caused by retransmission is worse than losing the information.

When using this model, the Ack-flag shall always be cleared.

E.1.2.2 Serial model

In the serial-model, when a PDU is sent, the Annex E layer waits until a positive reply is returned for the same Session-Identifier. This behaviour is used for protocols that cannot sustain out-of-order message arrival and require real-time operations while sending small amounts of information. An example of such a protocol is Q.931.

When using this model, the Ack-flag shall always be set for static-typed messages. Unless otherwise specified, Annex E implementations shall use the default retransmission timers (**T-R1** and **T-R2**) and counter (**N-R1**).

E.1.2.3 Mixed model

The mixed model may imply that the protocol state machine and the Annex E state-machine are intertwined. Such implementations may use the Ack-bit where appropriate.

When using this model, use of the Ack-flag can be forbidden, optional, or mandatory, as prescribed by the protocol.

E.1.2.4 Annex E over TCP

This annex may be used over TCP. When used over TCP, the Ack message shall not be used. In addition, the L-bit in the PDU header shall be set, triggering the availability of the payload-count or PDU-length fields.

E.1.3 Optional payload fields

E.1.3.1 Session identifier

Annex E payloads support an optional session field that may be used to identify a session within the multiplexed transport that the payload belongs to. The session field is 16 bits long.

NOTE – This field may be used for example to carry the CRV (e.g., Call Reference Value as defined in ITU-T Rec. Q.931) in H.225.0 messages. The interpretation of the session field is protocol specific.

E.1.3.2 Source/Destination address identifier

Annex E payloads support an optional Source/Destination field that may be used to identify the source, the destination (or both) of the payload. The Source/Destination field is 32 bits long.

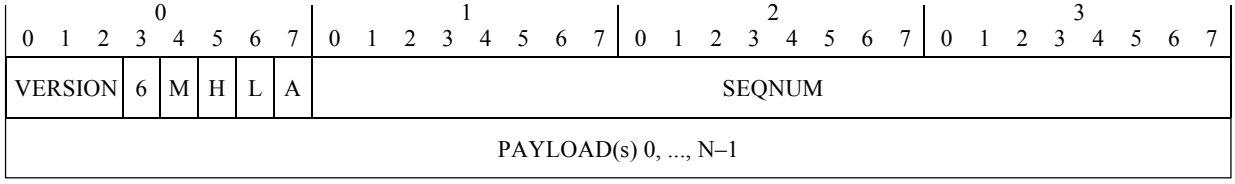
NOTE – This field may be used for example in ITU-T Rec. H.283 to express the [<M><T>] address identifying the source node of the packet, and the [<M><T>] address identifying the destination node of the packet. The interpretation of the source/destination field is protocol specific.

E.1.4 Wire-protocol

Annex E transport uses binary encoding as defined in the rest of this subclause. Structures and multi-byte fields shall use network-byte-ordering (e.g., big-endian).

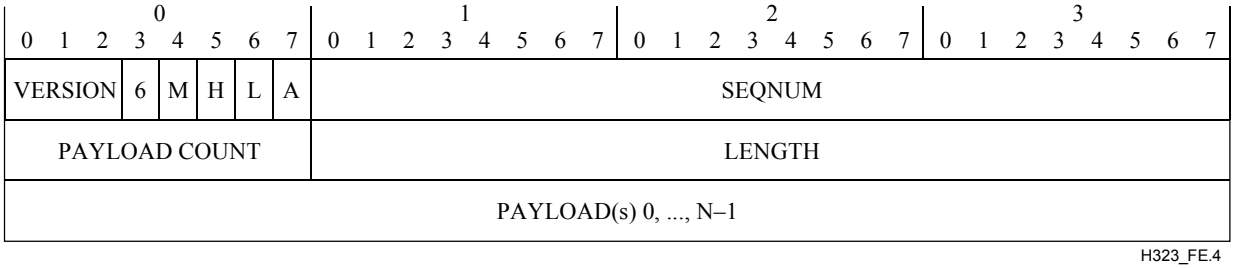
E.1.4.1 Header structure

The following structure shall be used to encode the Annex E Header. If the L-bit is cleared (hence there is no payload-count or PDU length indication), the length of the payloads within the message, and their number can be inferred from the message size as reported by the transport layer. See Figures E.3 and E.4.



Field	Content of fields	Bits
VERSION	Unsigned Integer; senders shall set this field to zero. Version number 7 is reserved for experimental use and shall be ignored by commercial implementations	3
6	When cleared, it means all IP addresses are IPv4 compliant (using 32 bits). When set, means all IP addresses are IPv6 compliant (using 128 bits)	1
M	Multicast bit. If set, the PDU was sent using Multicast, if cleared, the PDU was unicast. Senders shall set this bit if the PDU was multicast, otherwise they shall clear the bit	1
H	Reply-Hint bit – when set, this message will result in a reply, e.g., when set, the Ack message should be delayed to give the application a chance to provide an answer payload with the Ack payload	1
L	Length indicator. If present, an additional 4 OCTETs are present that contain the number of Payloads in the PDU (8 bits) and the total length (in OCTETs) of the PDU (24 bits)	1
A	Boolean: TRUE indicates that an Ack is requested for this PDU	1
SEQNUM	Unsigned Integer between 0 and 16 777 215: the sequence number of this PDU	24
PAYLOAD(s)	Sequence of payload structures	$8 \times n$

Figure E.3/H.323 – Header structure when the L-bit is cleared



Field	Content of L-bit supplementary fields	Bits
PAYLOAD COUNT	Total number of payloads in PDU –1 (e.g., 0 means there is one payload, 1 means there are two, etc.)	8
LENGTH	Total length in OCTETs of all payloads (excluding header)	24

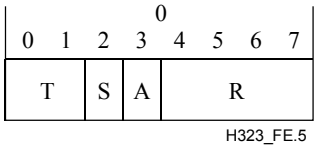
Figure E.4/H.323 – Header structure when the L-bit is set

E.1.4.2 Payload structure

The following structures shall be used to encode Annex E payloads.

E.1.4.2.1 Payload header flags

Every payload begins with a flags OCTET, that describes what optional fields are in the payload header. See Figure E.5.



Field	Content of fields	Bits
T	Two bits defining the payload identification type: **00**: Annex E Transport Messages **10**: Static-Payload typed messages **01**: OBJECT IDENTIFIER typed messages **11**: Reserved for Future Use	2
S	Signifies the presence of a Session field	1
A	Signifies the presence of a Source/Destination Address field	1
R	Reserved for future use, shall be cleared by senders	4

Figure E.5/H.323 – Payload flags

E.1.4.2.2 Annex E transport messages

Both T bits in the Payload header flags OCTET shall be set to 0 (zero) for all Annex E Transport Messages. The next octet shall signify what Annex E transport message is following. Both S and A bits shall be cleared. See Figure E.6.

Value	Interpretation
0	I-Am-Alive message
1	Ack message
2	Nack message
3	Restart Message
4..255	Reserved for future use

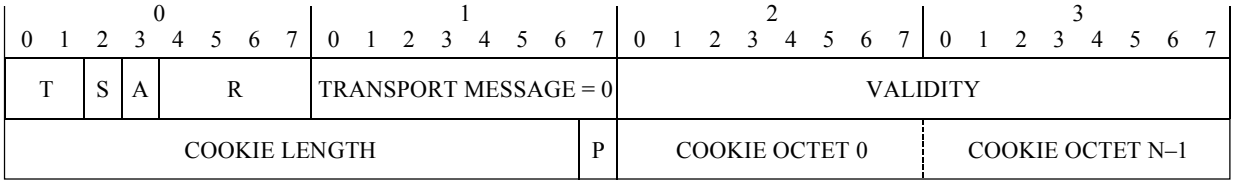
Figure E.6/H.323 – Annex E transport messages

E.1.4.2.2.1 I-Am-Alive message

The following structure shall be used to encode Annex E I-Am-Alive payloads. The transport-message octet shall be set to 0 (zero). The validity period is expressed in 100s of milliseconds.

- If the replyRequested bit (**P**) is set, the receiver shall reply with an I-Am-Alive message with the cookie (if provided).
- ReplyRequested is not the same as ackRequested in the PDU header, which results in an Ack message. replyRequested results in an I-Am-Alive message.
- If a validity period is set to ZERO (0), timer **T-IMA1** shall be used.
- PDUs that contain only an I-Am-Alive Payload shall clear the Ack-bit in the PDU header.

See Figure E.7.

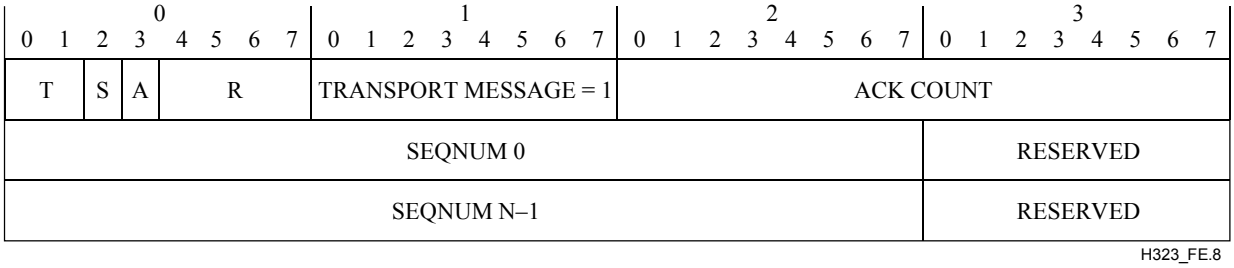


Field	Content of fields	Bits
VALIDITY	Unsigned Integer: The time in 100s of milliseconds that this I-Am-Alive is valid for	16
COOKIE LENGTH	The length (in BYTEs or OCTETs) of the COOKIE field	15
P	Reply Requested	1
COOKIE	BYTEs or OCTETs of the cookie	8 × n

Figure E.7/H.323 – I-Am-Alive message

E.1.4.2.2.2 Ack message

The following structure shall be used to encode Ack messages. The transport-message octet shall be set to 1 (one). PDUs that contain only an Ack Payload shall clear the Ack-bit in the PDU header. See Figure E.8.

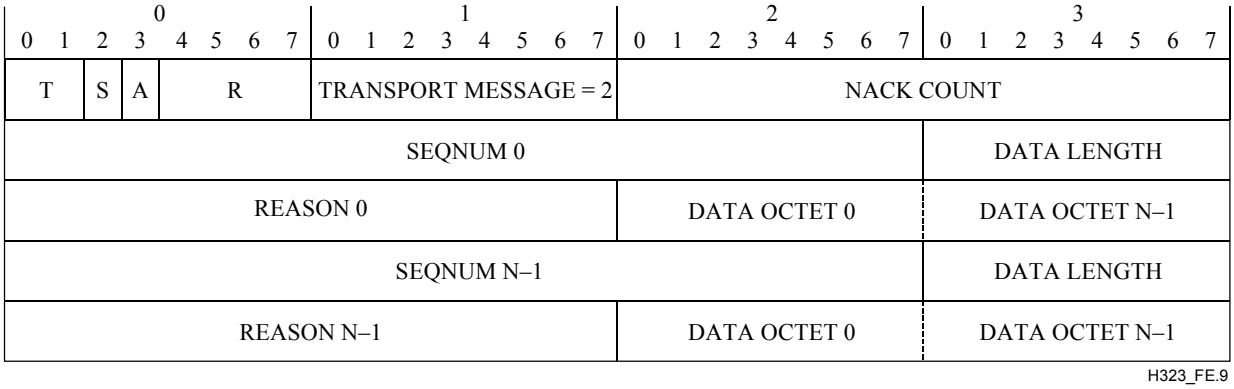


Field	Content of fields	Bits
ACK COUNT	The number of SEQNUM fields that follow	16
SEQNUM 0, ..., N–1	The Sequence Number(s) of the PDUs that are being ACKed for	24 × n
RESERVED	Reserved for future use	8 × n

Figure E.8/H.323 – Ack payload

E.1.4.2.2.3 Nack message

The following structure shall be used to encode Nack messages. The transport-message octet shall be set to 2 (two). The Nack message shall be used to signal transient errors, or more serious errors, such as the arrival of a malformed message. Unexpected Nack messages (such as ones bearing illegal sequence numbers) shall be ignored. See Figure E.9.



Field	Content of fields	Bits
NACK COUNT	The number of SEQNUM fields that follow	16
SEQNUM 0, ..., N–1	The Sequence Numbers of the PDUs that is being NACKed for	24 × n
LENGTH 0, ..., N–1	Length of Nack-specific data	8 × n
REASON 0, ..., N–1	The reason for the NACK	16 × n
OCTETs	Nack-specific data octets	8 × n

Reason value	Nack reason meaning	Length of Nack Data in octets	Data
0	Non-standard reason	1 + n	LENGTH OCTET followed by OBJECT IDENTIFIER OCTET(s)
1	Request the sender to use an alternate port for the specified static payload type	8	As defined in Figure E.10
2	Request the sender to use an alternate port for the specified ObjectID payload type	1 + n + 6	As defined in Figure E.11
3	Transport-payload not supported	1	Unsigned integer
4	Static-payload type not supported	1	Unsigned Integer; Payload as defined in the static-typed protocol that is not supported
5	Object-ID payload not supported	1 + n	LENGTH OCTET followed by OBJECT IDENTIFIER OCTET(s)
6	Payload Corrupted	1	The Payload number in the message that was corrupted
7.. 65535	Reserved for future use		

Figure E.9/H.323 – Nack message

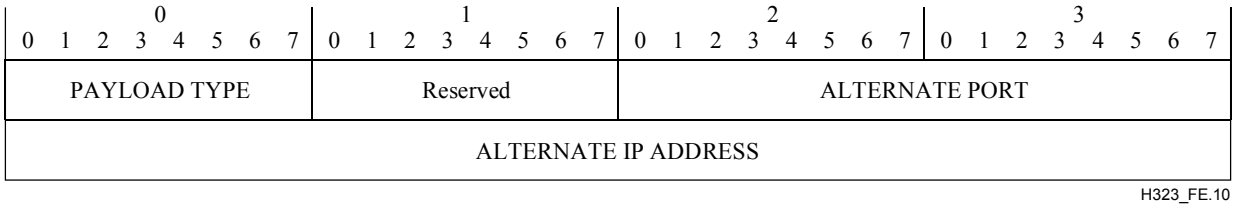


Figure E.10/H.323 – Nack reason 1 structure

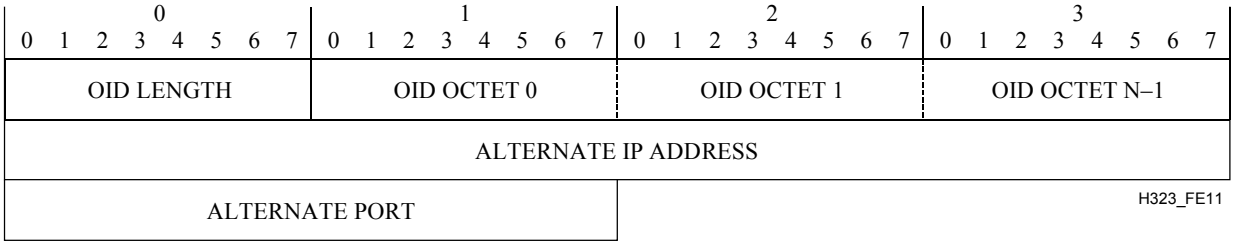


Figure E.11/H.323 – Nack reason 2 structure

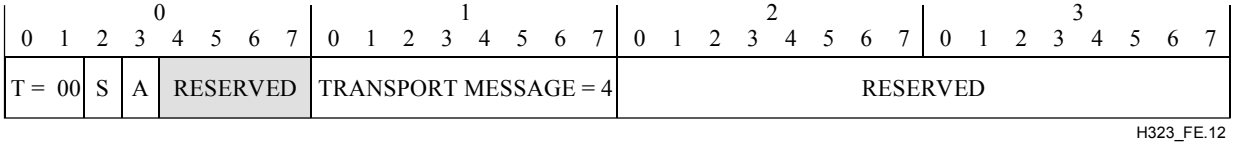
If the IP address is set to zero, the IP address of the sender shall be used (as identified by the TCP/IP layer). If the UDP port is set to zero, the port transmitted from shall be used (as identified by the TCP/IP layer).

E.1.4.2.2.4 Restart message

The following structure shall be used to encode Annex E Restart payloads. The transport-message octet shall be set to 3. Restart payloads are used to signal to the remote peer that the sender has restarted. Restart payload should be sent as a part of the first message to the remote entity. The receiver shall reset its receiver sequence number range on receiving the Restart payload. It shall consider any message arriving from the previous sequence number range as stale and shall ignore it.

The receiver shall tear down existing calls or start recovery procedures depending on the "action" field in the Restart payload.

If a restart does not affect ongoing calls, then it is invisible to the Annex E layer, and therefore shall not be signalled. See Figure E.12.



Field	Content of fields	Bits
action	The action desired by the receiver of the Restart payload	8

Action value	Meaning
0	Unspecified
1	Tear down calls
2	Start Recovery procedures
3..	Reserved for future use

Figure E.12/H.323 – Restart message structure

E.1.4.3 Static-typed messages

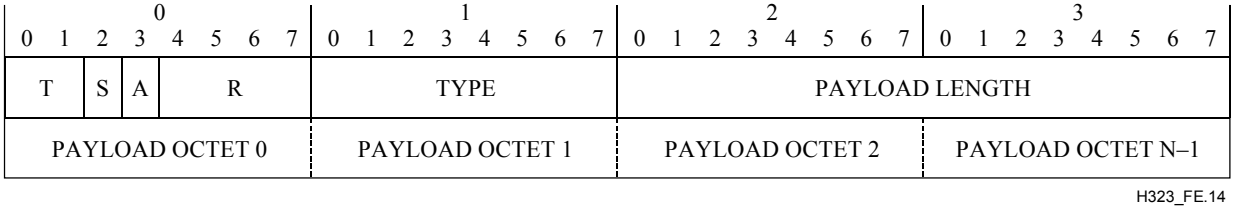
The first T bit in the Payload header flags OCTET shall be set to 1 (one) for all static-typed messages. The second T bit in the Payload header flags OCTET shall be set to 0 (zero) for all static-typed messages. The next octet shall signify what static-payload is present (see Figure E.13):

Value	Interpretation
0	Octet-stream contains a call signalling message as defined in ITU-T Rec. H.225.0
1..255	Reserved for future use

Figure E.13/H.323 – Static-typed payloads

E.1.4.3.1 Basic static-typed message (S-bit and A-bit cleared)

When both the S and A bits are cleared, the following payload format shall be used (see Figure E.14):

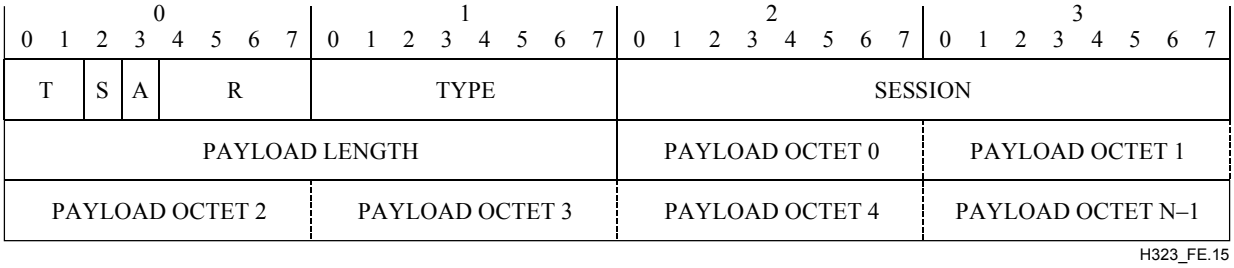


Field	Content of fields	Bits
TYPE	Unsigned Integer: the type of the payload, as defined in Figure E.13	8
LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
DATA	The actual payload data OCTETs	8 × n

Figure E.14/H.323 – Basic static-typed payload

E.1.4.3.2 Extended-1 static-typed message (S-bit set and A-bit cleared)

When the S-bit is set and the A-bit is cleared, the following payload format shall be used. The S-bit signifies the presence of a SESSION field. See Figure E.15.

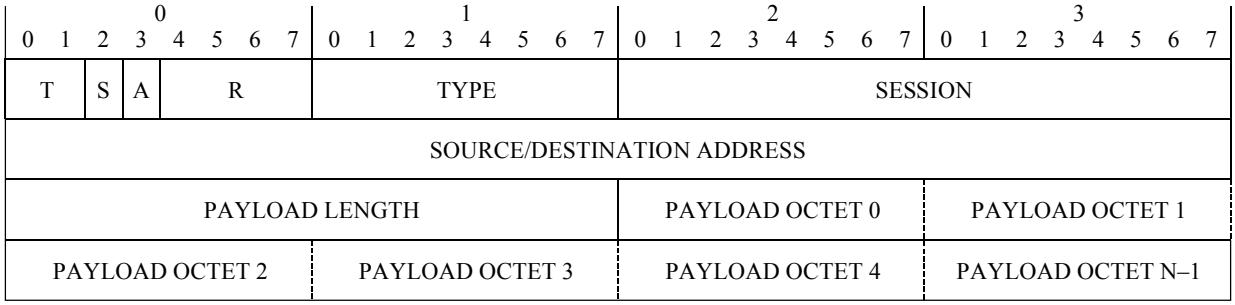


Field	Content of fields	Bits
TYPE	Unsigned Integer: The type of the payload, as defined in Figure E.13	8
SESSION	Unsigned Integer: The meaning of the session field is protocol dependent	16
PAYLOAD LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
DATA	The actual payload data OCTET(s)	8 × n

Figure E.15/H.323 – Extended-1 payload format

E.1.4.3.3 Extended-2 static-typed message (S-bit and A-bit set)

When both the S-bit and the A-bit is set, the following payload format shall be used. The A-bit signifies the presence of a Source/Destination Address field. See Figure E.16.



Field	Content of fields	Bits
TYPE	Unsigned Integer: The type of the payload, as defined in Figure E.13	8
SESSION	Unsigned Integer: The meaning of the session field is protocol dependent	16
SOURCE/DESTINATION ADDRESS	Unsigned Integer: The meaning of the source/destination address field is protocol dependent	32
PAYLOAD LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
DATA	The actual payload data OCTET(s)	$8 \times n$

Figure E.16/H.323 – Extended-2 payload format

E.1.4.3.4 Extended-3 static-typed message (S-bit cleared, A-bit set)

When the S-bit is cleared and the A-bit is set, the following payload format shall be used. The A-bit signifies the presence of a Source/Destination Address field. See Figure E.17.

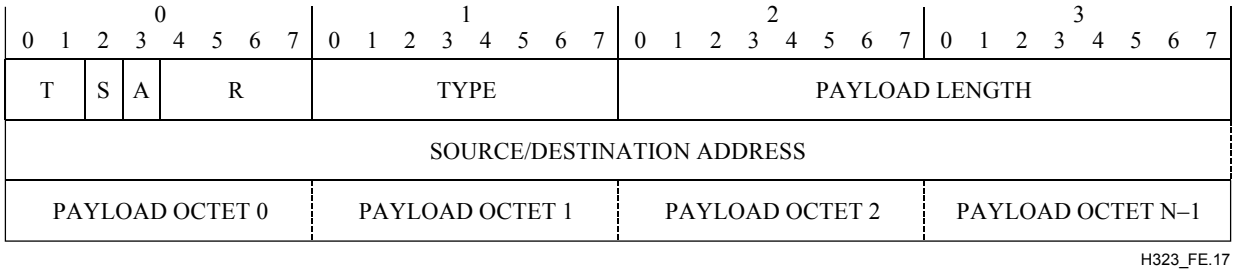


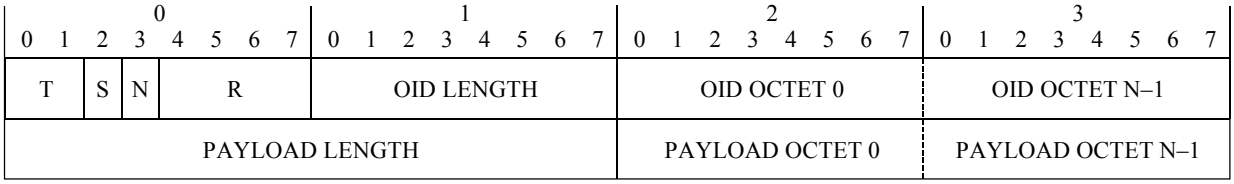
Figure E.17/H.323 – Extended-3 payload format

E.1.4.4 ObjectID-typed messages

The first T bit in the Payload header flags OCTET shall be set to 0 (zero) for all ObjectID-typed messages. The second T bit in the Payload header flags OCTET shall be set to 1 (one) for all ObjectID-typed messages. The next two octets shall signify the length of the Object-ID that follows.

E.1.4.4.1 Basic ObjectID-typed message (S-bit and A-bit cleared)

When both the S and A bits are cleared, the following payload format shall be used (see Figure E.18):

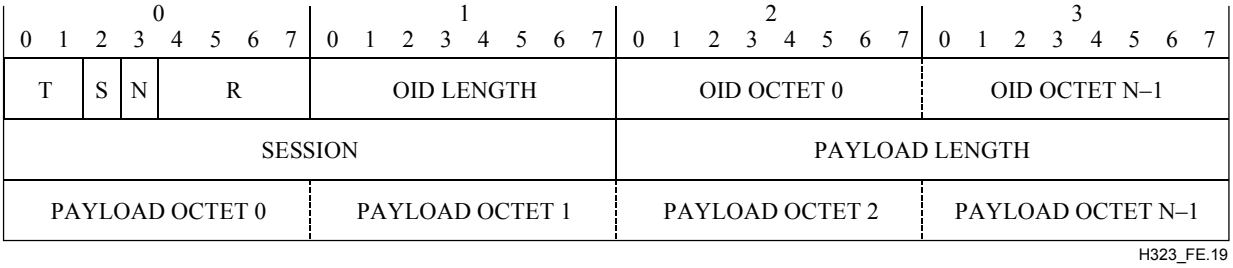


Field	Content of fields	Bits
OID LENGTH	Unsigned Integer: The length in OCTETs of the Object Identifier following	8
ObjectID	Object Identifier OCTETs	8 × n
LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
DATA	The actual payload data OCTETs	8 × n

Figure E.18/H.323 – Basic ObjectID-typed payload

E.1.4.4.2 Extended-1 ObjectID-typed message (S-bit set and A-bit cleared)

When the S-bit is set and the A-bit is cleared, the following payload format shall be used. The S-bit signifies the presence of a SESSION field, which is used by the application to associate payloads with a specific session. The definition of a session is protocol specific. See Figure E.19.

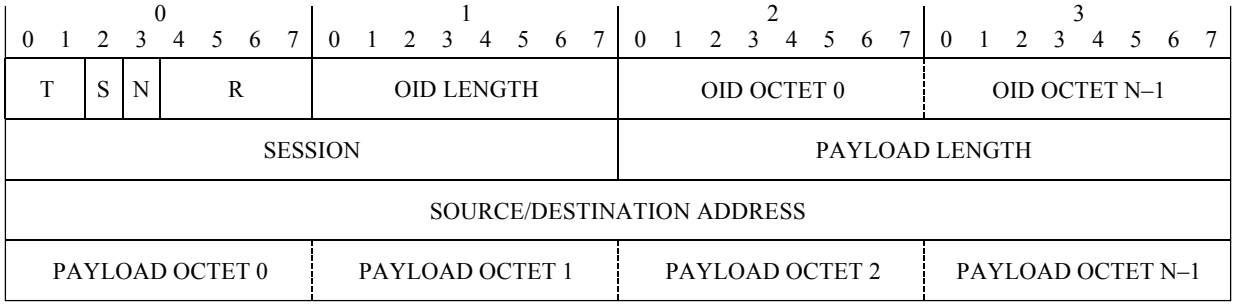


Field	Content of fields	Bits
OID LENGTH	Unsigned Integer: The length in OCTETs of the Object Identifier following	8
ObjectID	Object Identifier OCTETs	8 × n
SESSION	Unsigned Integer: The meaning of the session field is protocol dependent	16
LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
DATA	The actual payload data OCTETs	8 × n

Figure E.19/H.323 – Extended-1 ObjectID-typed payload format

E.1.4.4.3 Extended-2 ObjectID-typed message (S-bit and A-bit set)

When both the S-bit and the A-bit are set, the following payload format shall be used. The A-bit signifies the presence of a Source/Destination Address field. See Figure E.20.



Field	Content of fields	Bits
OID LENGTH	Unsigned Integer: The length in OCTETs of the Object Identifier following	8
ObjectID	Object Identifier OCTETs	8 × n
SESSION	Unsigned Integer: The meaning of the session field is protocol dependent	16
LENGTH	Unsigned Integer: The length (in OCTETS or BYTES) of the payload data	16
SOURCE/DESTINATION ADDRESS	Unsigned Integer: The meaning of the source/destination address field is protocol dependent	32
DATA	The actual payload data OCTETs	8 × n

Figure E.20/H.323 – Extended-2 ObjectID-typed payload format

E.1.4.4.4 Extended-3 ObjectID-typed message (S-bit cleared, A-bit set)

When the S-bit is cleared and the A-bit is set, the following payload format shall be used. The A-bit signifies the presence of a Source/Destination Address field. See Figure E.21.

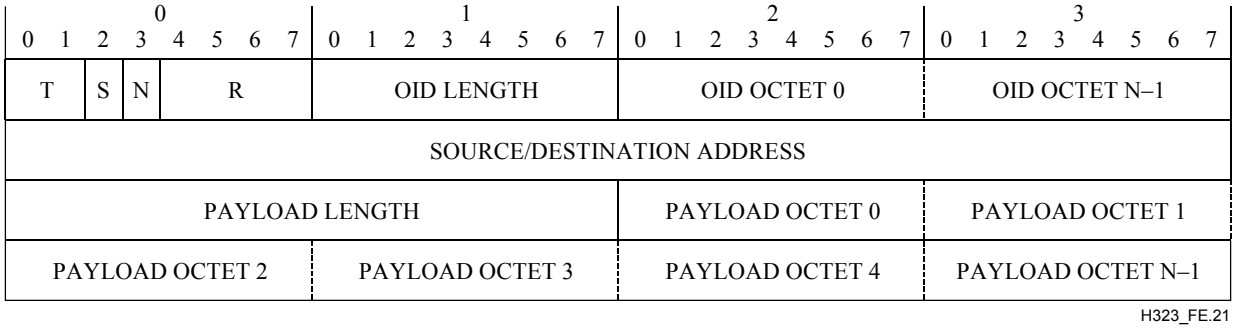


Figure E.21/H.323 – Extended-3 ObjectID-typed payload format

E.2 H.225.0 call signalling over Annex E

This clause describes how to carry H.225.0 Call Signalling messages using the Annex E transport, over UDP. Annex E is used to provide a "reliable-UDP" transport, to allow H.225.0 implementations to work over Annex E largely unchanged.

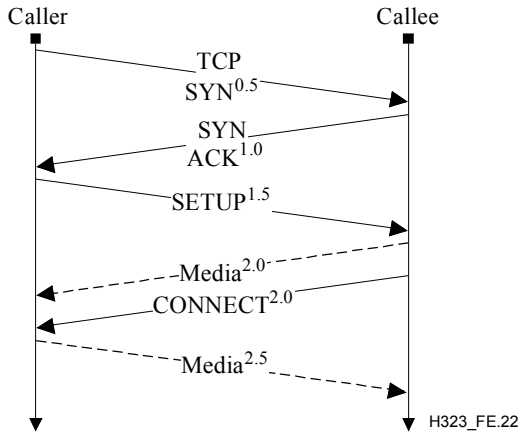
E.2.1 Rationale

ITU-T Rec. H.323 version 2 (1998) introduces the concept of "Fast Connect", which allows media cut-through in as little as in 2 round trips from callee to caller (including TCP messages), and in 2.5 round-trips from caller to callee.

This can be reduced to 1rt and 1.5rt respectively by using UDP as the transport for H.323 messages, instead of TCP. This is especially important when using the Gatekeeper-Routed-Model.

E.2.2 H.323 Call-Setup using this annex

ITU-T Rec. H.323 version 2 (1998) uses the TCP transport to carry H.225.0 messages, which means the least number of round trips possible to get media cut-through is 2 from Callee to Caller, and 2.5 from Caller to Called party. See Figure E.22.



NOTE – Some messages in the TCP handshake procedure have been omitted for clarity.

Figure E.22/H.323 – Information flow for H.323 version 2 (1998) FastConnect

E.2.2.1 UDP-based procedure

To get faster media cut-through, it is possible to use UDP for call signalling transport, which effectively enables media cut-through in a single round trip (see Figure E.23):

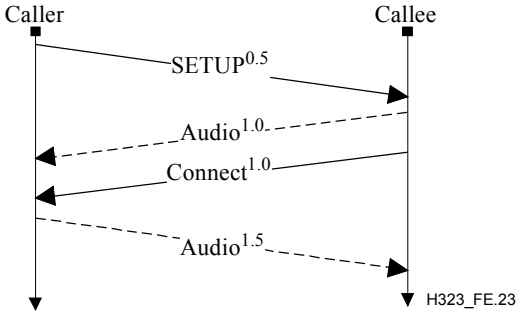


Figure E.23/H.323 – Information flow for UDP-based Call Setup

The Annex E layers should retransmit a lost packet if it does not get a reply after some time. The precise retransmission procedure is detailed in E.1.1.8.

E.2.2.2 Mixed TCP and UDP procedure

The procedures for TCP-based and UDP-based call setup are not mutually exclusive. If UDP-based and TCP-based call setup are carried out in parallel then the procedure in this clause shall be used. In the mixed procedure the originator transmits the SETUP message over UDP, and simultaneously establishes a TCP connection. If the originator has not received a response to the UDP SETUP when the TCP connection is established, then it also transmits the SETUP messages over the

TCP connection. If a callee receives the same SETUP message over UDP and over TCP, then it shall respond using either transport protocol (usually the one which arrived first) but not both.

If the originator receives a response over UDP then the TCP connection shall be released and communication continues over UDP. If the originator receives a response over TCP (for example because the remote peer does not support the Annex E procedures), then communication continues over TCP, UDP-based communication shall no longer be used for this call.

A callee that supports this annex shall select the transport protocol according to which arrives first: TCP Setup message, or UDP Setup message. Note these messages may be reordered in delivery. The caller is notified of the selection according to the transport protocol over which the subsequent message (e.g., Connect) has arrived. See Figure E.24.

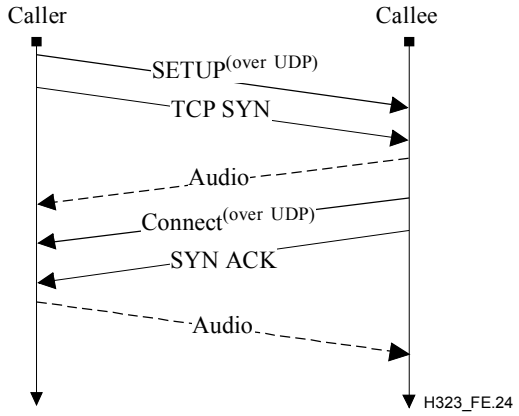


Figure E.24/H.323 – Information flow for mixed TCP and UDP procedure

This ensures that if the UDP procedure fails, usual TCP-based procedures can take over immediately (see Figure E.25):

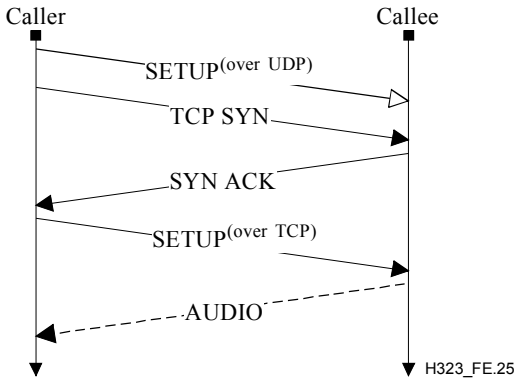


Figure E.25/H.323 – Information flow when UDP is not supported

This means that backwards compatibility when calling ITU-T Rec. H.323 version 1 (1996) or 2 (1998) entities is transparent, as the v1/v2 H.323 application will not be aware of the UDP packet.

NOTE – It is recommended for entities that initiate a call and do not know if the remote side supports Annex E operations to use the procedure detailed above. If the calling entity knows by some means that the remote callee supports UDP-based operations, it may use a UDP only call setup.

E.2.3 Specifics

E.2.3.1 Message identification

H.225.0 over Annex E payloads shall use static payload type **0** (zero).

E.2.3.2 Well-known port

UDP port **2517** shall be used for the well-known port. Entities may transmit from any random port. A single H.323 entity on a physical device shall use a single, distinct UDP port as the advertised port for receiving messages. However, it may utilize a distinct port on each interface if the physical device has multiple network interfaces.

The calling entity shall send all Annex E messages for a call to the called entity's advertised destination port. The called entity shall send all Annex E messages related to said call to the IP address and port from which the initial Annex E message for the call was received. The called entity shall send all Annex E messages using the same port on which it received the initial H.225.0 PDU from the caller.

The calling entity may transmit messages from any random port, but shall use the same port throughout the duration of the call.

E.2.3.3 Signalling model

H.225.0 over Annex E shall use the **serial-model** as described in E.1.2.2.

E.2.3.4 Timers

H.225.0 over Annex E shall use default timers and values. The **T-IMA1** timer shall be reset upon reception of any Call Signalling message (e.g., but not when receiving RTP packets).

E.2.3.5 Session field

The session field shall be present in all payloads. The Session value shall contain the CRV from the H.225.0 call signalling messages. Specifically, the call reference flag shall be included as the most significant bit of the CallReferenceValue. This restricts the actual CRV to the range of 0 through 32 767, inclusive.

E.2.3.6 Source/destination address field

Use of the Source/Destination field is optional, but shall be present in all messages originating, or destined to an MCU or when a Gatekeeper acts as an MC.

E.2.3.7 MTU

Call-Signalling messages that require sending large amounts of data (such as certificate-based authentication and authorisation) should use TCP for call-setup, as using them over this annex may cause fragmentation due to messages being larger than path MTU.

E.2.3.8 H.245

H.245 shall be transmitted using the ITU-T Rec. H.323 version 2 (1998) H.245 Tunnelling procedures.

E.2.3.9 Receiver sequence number policy for H.225.0 over Annex E

When receiving a H.225.0 message over Annex E, an entity shall check the host-address, source-port and sequence-number to recognize duplicate messages. The transmitting entity follows serial model for the same Session-Identifier and assigns sequence numbers per host-address and source-port. Since, for a single H.323 call, it is not possible for messages to get out of order, the Annex E layer shall not attempt to reorder messages according to sequence numbers. Gaps in the sequence numbers are possible and an entity shall not recognize it as a packet loss.

Annex F

Simple endpoint types

F.1 Introduction

Simple Endpoint Types, i.e., devices manufactured for a single purpose, may comprise a significant fraction of the overall set of H.323 capable end systems. In contrast to full-featured H.323 devices (many implementations of which are PC-based), the so-called Simple Endpoint Types (SETs) may be implemented in inexpensive stand-alone boxes, the most prominent example being the simple telephone.

NOTE – Sample application scenarios for such systems were found to include:

1) palmtop computer with audio communications capabilities (voice, file transfer, fax, etc.);
2) telephone with an RJ-45 connector;
3) text telephones (using ITU-T Rec. T.140);
4) cellular IP phone;
5) mobile system with integrated voice and data communications (UMTS, IMT-2000).

All these systems have in common that they support a relatively fixed set of functionality: voice and/or rudimentary (i.e., not T.120) data communication facilities. It is important to note that this functionality does not need to be extended for the respective system's purpose: a telephone set without (an elaborate) display does not need to support video functionality, neither does it require data conferencing capabilities.

All of these systems have a limited amount of resources available (e.g., processing power, communication bandwidth, memory).

This annex outlines the scope of SET devices in general and defines the procedural and protocol details of a Simple Audio Endpoint Type (Audio SET device). In particular, this annex defines the functional baseline for all types of Simple Endpoint Types; hence, further SETs are to be defined by referencing this annex and then only specifying additions to the procedures and conventions set forth in this annex.

This annex defines a subset of H.323 functionality and any deviations from Recommendation H.323 are explicitly identified. Any procedures not explicitly described in this annex are covered by the main body of this Recommendation.

The development of SET devices has potential implications on other H.323 devices: in particular, MC(U)s and gateways should be aware of their potentially minimal support for H.323 (1998) functionality in order to provide SET devices with seamless access to enhanced H.323 services such as multipoint conferences and supplementary services. Alternatively, external proxy devices may be provided to bridge the different functional ranges between SET devices and full-featured H.323 (1998) endpoints. Interoperability issues are addressed in more detail in F.9.

F.2 Specification conventions

This annex specifies only those services, procedures, protocol messages, etc. that are mandatory for the implementation of a SET device, which is a subset of the mandatory functionality of an H.323 (1998) system. This implies that a SET device shall not assume any functionality beyond what is specified mandatory in this annex from another SET device.

In addition to the mandatory components, several clauses of this annex specify conditionally mandatory services, procedures, protocol messages, etc. based on the concept of functional blocks that are optional as a whole. However, a SET device that decides to implement a particular

functional block, must support all the components defined as mandatory for this functional block; optional components may be supported.

All other features defined in ITU-T Rec. H.323 are, by definition, optional, and their implementation in a SET device is entirely at the discretion of the manufacturer.

F.3 Scope

This annex specifies rules on the use of ITU-T Rec. H.323 that enable Simple Endpoint Types to be implemented in a simple fashion. The following (non-exhaustive) list of Simple Endpoint Types is envisioned for standardization by the ITU-T:

1) **simple telephone (Simple Audio Endpoint Type)** – defined in this annex;

2) **simple telephone with security capabilities** – for further study;

3) **text conversation terminal** – for further study;

4) **fax device** – for further study.

The simple telephone is defined in this annex. Secure simple telephone, text terminal, and simple fax device are Simple Endpoint Types for further study. The profiles for Simple Endpoint Types can be categorized as follows:

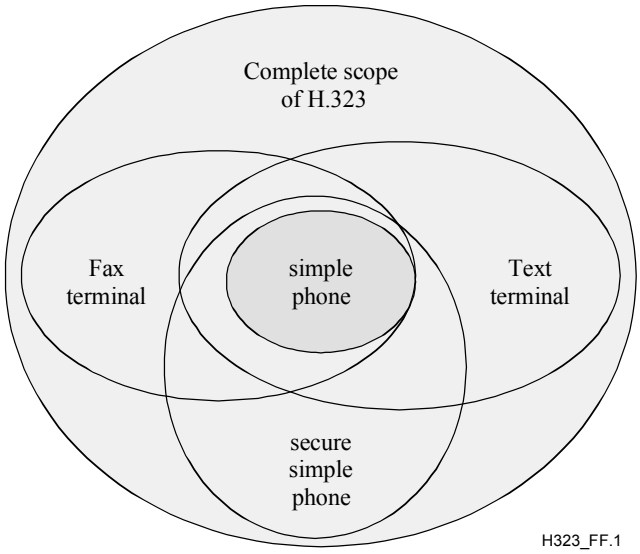


Figure F.1/H.323 – Venn diagram showing functional ranges of the various SET devices

Figure F.1 is a schematic picture of the different Simple Endpoint Types that are being defined in the context of H.323 'profiles', in a so-called Venn diagram. In this diagram, the relation between the SETs is illustrated. The wider ellipse show the context of a full H.323 compliant system. As an example, the simple telephone is put in the figure. As it is clearly a subset of the full compliant H.323 system, it lays completely within its scope. A secure simple telephone, containing additionally the security capabilities, comprises the capabilities of the simple telephone (e.g., same audio codecs, same call setup, etc.). The interoperability between a simple telephone SET implementation and a secure simple telephone will therefore be ensured.

The SET devices are defined in a way that enables them to interoperate seamlessly with one another and with H.323 (1998) devices supporting the FastConnect procedure as well as with all SET-aware H.323 endpoints.

NOTE – Although SET devices are defined with simple devices in mind, it is equally possible to build gateways based upon the respective SET specification. No additional definitions are required for other types of devices.

F.4 Normative references

All the normative references in the body of this Recommendation and in ITU-T Rec. H.225.0 (2003) apply.

F.5 Abbreviations

This annex uses the following abbreviations:

Audio SET	Simple Audio Endpoint Type
Fax SET	Simple Facsimile Endpoint Type
Secure Audio SET	Secure Simple Audio Endpoint Type
SET	Simple Endpoint Type
Text SET	Simple Text Telephony Endpoint Type

F.6 Simple (Audio) Endpoint Type – System functionality overview

The following characteristics apply to Simple Audio Endpoint Types (Audio SET devices):

Media capabilities

- Voice-capability
  - mandatory: G.711 (A-law and μ-law)
  - suggested options: G.723.1, G.729, GSM.
  - suggested options: audio redundancy encoding with any combinations of the above codecs.
- Audio SET devices shall only support symmetric audio operation.
- No data-capability.
- DTMF capability mandatory; transmission as H.225.0 Information messages mandatory; transmission as RTP payload for further study.
- No video capability.
- No T.120 capability.
- Media distribution: support for unicast mandatory.

The mandatory and optional media capabilities shall be defined separately for other Simple Endpoint Types.

Control capabilities

The following minimum control capabilities shall equally apply to all Simple Endpoint Types.

- FastConnect sequence of ITU-T Rec. H.323 (1998) mandatory.

NOTE – Audio SET devices are by default capable of participating in multipoint conferences, where they are obviously limited to audio communications.

Most other control capabilities are optional, in particular:

- UDP-based Faster-Connect Annex E/H.323 optional.
- Supplementary services (solely based upon H.450.x) optional.
- Support for H.245 messages and procedures optional.
- Support of more than a single call/conference at a time is optional.

Some control capabilities are disallowed for Audio SET devices.

- MC functionality prohibited.

F.7 Procedures for Simple Endpoint Types

This clause specifies for all the protocols required by this Recommendation, the detailed level of support by SET devices in general, and the specific requirements for Audio SET devices:

- Registration, Admission and Status (RAS) signalling (H.225.0), see F.7.1;
- Call signalling (H.225.0), see F.7.2;
- Multimedia system control signalling (H.245), see F.7.3;
- Media packetization and transport (H.225.0, RTP), see F.7.4;
- Supplementary Services (H.450.x), see F.7.5 and F.7.6;
- Multipoint conference operation, see F.7.7;
- Loosely-coupled conferences (H.332), see F.7.8;
- Management Information Bases, see F.7.9.

Security services as specified in ITU-T Rec. H.235 to create Secure SET devices are considered in F.8.

F.7.1 RAS Signalling (H.225.0 RAS)

SET devices shall comply with the RAS procedures as defined in ITU-T Recs H.323 (1998) and H.225.0 (1998) with the following modifications applying.

A SET device shall use the pre-granted ARQ procedures as specified in ITU-T Rec. H.225.0 (1998) and shall be able to determine whether an incoming call request is received from its Gatekeeper. A SET-aware Gatekeeper shall support pre-granted ARQ and shall pre-grant for placing and receiving calls with call routing through the Gatekeeper for SET devices (to be indicated in the preGrantedARQ component). If a contacted Gatekeeper does not support pre-granted ARQ or does not provide the aforementioned pre-granting configuration, a SET device shall register with another Gatekeeper.

SET devices shall at a minimum support the following RAS messages: transmission of GRQ, RRQ, URQ, UCF, and XRS and reception of GCF, GRJ, RCF, RRJ, URQ, UCF, URJ, and XRS. SET devices may support additional RAS messages.

A SET device shall include the "set" component of the H.225.0 EndpointType when communicating with a Gatekeeper and set the bits as follows.

Bit 0: = 1 if the device has Audio SET functionality.

Bit 1: = 0 if the device is not conference-aware.

Bit 1: = 1 if the device is conference-aware.

Use of the other bits will be defined by additional SET specifications.

F.7.2 Call signalling (H.225.0 Call Control)

SET devices shall comply with the call control procedures defined in ITU-T Recs H.323 (1998) and H.225.0 (1998). SET devices shall not close the call signalling channel after call establishment.

SET devices shall implement the FastConnect procedures as specified in ITU-T Rec. H.323 (1998). When originating a call, a SET device shall place a call using FastConnect.

SET devices shall support H.225.0 Information messages in the call signalling channel. Such messages should be used for, but are not restricted to, conveying user input in the Keypad Information Element.

SET devices should use the Status Enquiry and Status messages of ITU-T Rec. H.225.0 to estimate round-trip times to its peer.

SET devices may implement UDP-based call setup as outlined in Annex E/H.323. If UDP-based call setup is implemented, a SET device should attempt to call another endpoint via UDP-based call setup first.

Implementation of supplementary services based upon H.450.x is optional for SET devices. SET devices shall be able to safely ignore H.225.0 Facility messages that they do not understand.

A SET device shall include the "set" component of the H.225.0 EndpointType when exchanging call signalling PDUs with its peer. The bits of the "set" components shall be set as defined in F.7.1.

F.7.3 Multimedia system control signalling (H.245)

F.7.3.1 H.245 control channel

The FastConnect procedure shall be used for connection establishment. Repeated transmission of the fastStart element in H.225.0 call signalling messages shall be used to reconfigure or re-route media streams.

SET devices shall not open a separate H.245 connection:

a) They shall restrict H.245 signalling to the **OpenLogicalChannel** structure in the FastConnect sequence along with implicit Master Slave Determination.

b) If further H.245 signalling is required, they shall perform tunnelling as defined in ITU-T Rec. H.225.0 (1998).

SET devices shall use the syntax of ITU-T Rec. H.245 (1998) or later versions.

No specific procedures are defined for H.245 messages. If SET devices implement H.245 functionality, they shall adhere to the procedures defined in ITU-T Recs H.323, H.225.0 and H.245.

F.7.3.2 Master-Slave Determination

SET devices shall implicitly assume the slave role in any communication relationship without an H.245 control channel.

In case an H.245 tunnel is established, following the rules of 6.2.8.4/H.323 (1998), the SET device shall indicate a value of 40 for the **terminalType**. This ensures that in case a SET device connects to a full H.323 (1998) device, the latter will win the Master-Slave-Determination.

F.7.3.3 Terminal capability exchange

Although SET devices are by definition restricted in their supported functional range, a capability exchange procedure cannot be circumvented to allow for a minimum of diversity in the devices. However, the range of possible capabilities that may be signalled by a SET endpoint is restricted to what is defined in the following, and the capability exchange procedures shall adhere to the rules set forth in this subclause.

The Capability Exchange procedure for media types and transmission modes shall be carried out following the rules of the FastConnect procedure using multiple Open Logical Channel structures as a selection of possibilities offered by the caller out of which the callee chooses a subset to send and receive.

The following subclause list which capabilities need to be understood on the receiving (called) side and which may be transmitted on the sending (calling) side for Audio SET devices.

F.7.3.3.1 Audio capability

- G.711 (μ-Law, A-Law, 56 kbit/s, 64 kbit/s)
  The following alternatives shall be supported:

```
AudioCapability.g711Alaw64k         ≥ 20     number of frames
AudioCapability.g711Alaw56k         ≥ 20     number of frames
AudioCapability.g711Ulaw64k         ≥ 20     number of frames
AudioCapability.g711Ulaw56k         ≥ 20     number of frames
```

- G.723.1 (silence suppression or not, low and high rate)

  A SET supporting G.723.1 must at a minimum support:

```
AudioCapability.g7231
   maxAl-sduAudioFrames             ≥ 1      number of frames
   silenceSuppression                        True/False as appropriate
```

- G.729 (plain or Annex A)

  A SET supporting G.729 must at a minimum support:

```
AudioCapability.g729                ≥ 1      number of frames
AudioCapability.g729AnnexA          ≥ 1      number of frames
```

- GSM (full rate, enhanced full rate, half rate).

  A SET supporting GSM must at a minimum support:

```
AudioCapability.gsmFullRate         GSMAudioCapability,
AudioCapability.gsmHalfRate         GSMAudioCapability,
AudioCapability.gsmEnhancedFullRate GSMAudioCapability
```

  with GSMAudioCapability defined as appropriate for each of these rates:

```
GSMAudioCapability.audioUnitSize    ≥ 1      number of frames
GSMAudioCapability.comfortNoise              True/False as appropriate
GSMAudioCapability.scrambled                 True/False as appropriate
```

F.7.3.3.2 Video Capability

Audio SET devices do not support video.

F.7.3.3.3 Data Capability

Audio SET devices do not support data.

F.7.3.3.4 Conference Capability

SET devices are assumed to be proxied into centralized conferences with centralized data distribution (see F.7.7).

F.7.3.3.5 User Input Capability

SET devices shall support transmission of DTMF as Keypad Information Elements in the H.225.0 call signalling connection (e.g., using Information messages).

F.7.3.3.6 Security Capability

Security for SET devices, i.e., the definition of Secure SET devices, is for further study. Refer also to F.8.

F.7.3.3.7 maxPendingReplacementFor

Shall be supported by Audio SET devices. A value equal to '1' shall be implicitly assumed:

```
maxPendingReplacementFor = 1
```

Hence, the **maxPendingReplacementFor** parameter shall not be signalled explicitly.

F.7.3.3.8 nonStandardCapability

Use of non-standard capabilities, on the top level of the capability structure as well as within the aforementioned capability categories, should be avoided as far as possible.

F.7.3.3.9 Additional rules for the use of capabilities

For Audio SET devices, audio capabilities shall only be signalled via the FastConnect procedure and repeated exchange of **OpenLogicalChannel** structures using the FastConnect.

Video capabilities, data capabilities, conference capabilities, security capabilities, and h233encryption capabilities shall not be used.

The values of the MultiplexCapability table entry of an Audio SET device shall be assumed as follows:

maximumAudioDelayJitter	≥ 250 ms
receiveMultipointCapability, transmitMultipointCapability, and receiveAndTransmitMultipointCapability	
multicastCapability	TRUE/FALSE as appropriate, default FALSE[1]
multiUnicastConference	TRUE/FALSE as appropriate, default FALSE[1]
mediaDistributionCapability	
centralizedControl	TRUE
distributedControl	FALSE
centralizedAudio	TRUE
distributedAudio	TRUE/FALSE as appropriate, default FALSE[1]
centralizedVideo	FALSE
distributedVideo	FALSE
centralizedData	ABSENT
distributedData	ABSENT
mcCapability	
centralizedConferenceMC	FALSE
decentralizedConferenceMC	FALSE
rtcpVideoControlCapability	ABSENT
mediaPacketizationCapability	ABSENT
...,	
transportCapability	ABSENT
redundancyEncodingCapability	Audio redundancy encoding only (if any)
logicalChannelSwitchingCapability	FALSE
t120DynamicPortCapability	FALSE

Capabilities signalled from the remote side that are not understood shall be ignored.

---

[1] Multicast, multi-unicast, and distributed audio may be supported by Conference-aware Audio SET devices.

F.7.3.4 Logical Channel Signalling Messages

The opening of logical channels shall adhere to the FastConnect specifications of ITU-T Rec. H.323 (1998).

In addition, SET devices shall support reconfiguration of media streams at any time during a call. Open Logical Channel structures shall be tunnelled in H.225.0 call signalling messages following the procedures defined in ITU-T Recs H.225.0 (1998) and H.323 (1998) reusing the fastStart element of the H.225.0 call signalling message. Open Logical Channel structures outside the FastConnect procedure shall be used to alter media stream parameters – to provide a basis for supplementary services. Such Open Logical Channel structures shall be interpreted upon reception as follows.

- If the logical channel number matches a currently open logical channel, the respective channel shall be reconfigured following the principles of the FastConnect procedure if the **dataType** component is not "null". If the **dataType** component is "null" – indicating a "NullChannel" – the respective logical channel shall be considered closed and media transmission on this logical channel shall cease.
- If the logical channel number does not match a currently open channel, a new logical channel shall be opened following the princicples of the FastConnect procedure.

In the following, the restrictions on Open Logical Channel request are outlined:

```
OpenLogicalChannel
    forwardLogicalChannelNumber         LogicalChannelNumber
    forwardLogicalChannelParameters
        portNumber                      ABSENT
        dataType                        a valid audio data type
                                        (see F.7.3.3.1)
        multiplexParameters             CHOICE:h2250LogicalChannelParameters
        forwardLogicalChannelDependency ABSENT,
        replacementFor                  used if another Logical Channel is
                                        to be replaced

    reverseLogicalChannelParameters
        dataType                        a valid audio data type
                                        (see F.7.3.3.1)
        multiplexParameters             CHOICE:h2250LogicalChannelParameters
        reverseLogicalChannelDependency LogicalChannelNumber OPTIONAL,
        replacementFor                  used if another Logical Channel is
                                        to be replaced
    separateStack                       ABSENT
    encryptionSync                      ABSENT for Audio SET devices; FFS.
```

To the **H2250LogicalChannelParameters** structure, the following restrictions apply:

```
H2250LogicalChannelParameters
    nonStandard                     should be ABSENT
    sessionID                       INTEGER(0..255)
    associatedSessionID             ABSENT
    mediaChannel                    TransportAddress - should be a unicast
                                    address
    mediaGuaranteedDelivery         ABSENT
    mediaControlChannel             PRESENT - reverse RTCP channel
    mediaControlGuaranteedDelivery  FALSE
    silenceSuppression              as appropriate
    destination                     typically ABSENT
    dynamicRTPPayloadType           as appropriate,
    mediaPacketization              as appropriate; may only specify the
                                    payload format used
```

```
    rtpPayloadType
    payloadDescriptor           should refer to an rfc-number
    payloadType                 (dynamic) payload type value to be used

transportCapability
    nonStandard                 should be ABSENT
    qOSCapabilities             should be ABSENT (may only contain RSVP
                                parameters)
    mediaChannelCapabilities    should be ABSENT (may indicate "ip-udp")

redundancyEncoding              optional; only audio redundancy is
                                allowed
source                          typically ABSENT
```

F.7.4 Media exchange

For media exchange, SET devices shall follow the H.323 and H.225.0 procedures using RTP/UDP/IP to convey the media streams. The appropriate media packetization formats shall be used.

F.7.5 Supplementary services (H.450.x)

Support of any of supplementary services according to the H.450.x series of Recommendations is optional.

NOTE – If H.450.x functionality is not provided by a SET device, the SET device should implement the message rejection functionality (Interpretation APDU) of H.450.1 to enable its peer to quickly determine non-availability of supplementary services on side of the SET device. If H.450.1 message rejection is not implemented, the peer has to rely on a timeout.

A baseline for supplementary services to be supported by SET devices is for further study.

F.7.6 Third-party initiated pause and re-routing

Support for third-party initiated pause and re-routing is similar to the procedures outlined in 8.4.6/H.323 (1998), with the following modifications applying.

F.7.6.1 Initiating side

To re-route a call connecting to a SET device its peer (typically a Gatekeeper) shall transmit a NullChannel specification in the fastStart element in a message of the call signalling channel.

Subsequently, the initiating entity shall again transmit the (for the new peer) appropriate **OpenLogicalChannel** structures, similar to the capability negotiation and media stream establishment in the FastConnect procedure, and include the new transport addresses to redirect the media stream sourced by the SET device. The **OpenLogicalChannel** structures are carried in an H.225.0 call signalling message.

The **OpenLogicalChannel** structure should offer the same audio encodings that were offered in the initial call.

F.7.6.2 Receiving side (SET device)

Upon reception of a NullChannel specification in a fastStart element, a SET device shall stop transmitting the media stream(s) immediately and shall be prepared to handle interruptions in the received media stream(s). The SET device shall expect a repeated exchange of capability and transport addresses following the principles of the FastConnect procedure.

Upon reception of an **OpenLogicalChannel** structure carried in an H.225.0 call signalling message, the SET device shall select an acceptable media encoding from the selection offered by the initiating entity, following the rules of the FastConnect procedure. The SET device shall then start transmitting its media stream(s) to the transport address(es) newly indicated in the **OpenLogicalChannel** structures.

F.7.7 Conference-mode operation

SET devices may participate in multipoint conferences in either of two ways:

- by being proxied into a conference through a dedicated external device, such as a SET-aware MC combined with a suitable MP or a SET-specific proxy as outlined in F.7.7.1 as the default mode of operation for SET devices; or
- by implementing the necessary procedures of the H.225.0 and H.245 protocols as outlined in this clause. This mode of operation is defined in F.7.7.2.

F.7.7.1 Conference-unaware SET devices

The default mode of operation for SET devices does not require any awareness of conferencing functionality in a SET device itself. Instead, an external entity is assumed that bridges between a full-featured H.323 device and the SET device. This logical entity may be a stand-alone proxy device or may be part of an MC(U), a Gateway, or a Gatekeeper.

NOTE – The functionality of a logical bridging entity may include the following:

- concealing the existence of conference-related H.245 commands and responding appropriately in the direction of the full-featured H.323 device;
- adapting H.245 capability and logical channel signalling including multipoint mode commands;
- mixing several incoming audio streams and providing a single stream to the SET device;
- translating transport addresses for the audio stream;
- transcoding audio streams; and
- offering access to conference control functions via simple input means (such as DTMF signalling) to the SET device.

F.7.7.2 Conference-aware SET Devices

The specification of conference-aware SET devices is for further study.

Nevertheless, SET devices may follow the full procedures for conference-mode operation defined in the H.323 series of Recommendations.

F.7.8 Support for loosely-coupled conferences (ITU-T Rec. H.332)

Support for loosely-coupled conferences according to ITU-T Rec. H.332 is optional:

- Participation as a member of the panel is optional; it is provided either if conference-mode operation and media distribution via multicast are supported, or if an appropriate MC/MP combination hides all the conference commands from the SET device and only presents a single audio-stream.
- Participation as a member of the audience is optional; it is possible if the SET device supports multicast reception of information and is capable of receiving and interpreting H.332 session announcements.

F.7.9 Management Information Bases (MIBs)

Implementation of Management Information Bases is optional for SET devices. If MIBs are included in the implementation, the following H.323-related MIBs should be implemented:

- Call signalling;
- Terminal entity;
- RAS;
- Real time Protocol (RTP).

Details are for further study.

F.8 Security extensions

Plain SET devices are not capable of supporting H.235 security services. Secure SET devices, however, define a simple extension to SET devices covering security functionality using a subset of the mechanisms specified in ITU-T Rec. H.235.

The details of Secure SET devices are covered in Annex J.

F.9 Interoperability considerations

This annex specifies a SET device as a well-defined subset of the total H.323 functionality.

SET devices should always be used in conjunction with SET-aware Gatekeepers. The SET-aware Gatekeeper shall perform pre-granted ARQ and shall employ the Gatekeeper-routed call model to ensure full interoperability with other H.323 (1996) and H.323 (1998) devices.

In addition, SET-awareness may be built into MC(U)s or gateways to achieve seamless interoperability.

Table F.1 presents an overview of interoperability achieved between Audio SET devices and other H.323 endpoints.

Table F.1/H.323 – Interoperability of SET devices with other H.323 devices

	H.323 (1996)	**H.323 (1998)**	**H.323 (1998) with Fast Connect**	**SET device**
H.323 (1996)	√	√	√	√ (GK)
H.323 (1998)	√	√	√	√ (GK)
H.323 (1998) with Fast Connect	√	√	√	√ a)
SET device	√ (GK)	√ (GK)	√ a)	√

(GK) Indicates that a SET-aware Gatekeeper is needed for interoperation.

a) Optional redirection of media channels requires repeated execution of FastConnect in both endpoints.

F.10 Implementation notes (Informative)

This clause provides informative text on simple encoding of most of the necessary H.245 messages without requiring specific ASN.1 encoders/decoders.

NOTE – All these messages are transmitted as tunnelled H.245 messages; i.e., the resulting bit patterns are encoded as a single OCTET STRING of the SEQUENCE in the fastStart component of a H323-UU-PDU. In the tables shown below, the leftmost octet (octet #0) of the first row (word #0) is placed in the first octet of the octet string, followed by octet #1 of the first row, and so on. Octet #3 of word #n is followed by octet #0 of word #(n+1).

If numbers are to be encoded, 2-complement encoding is used for numbers that may be negative. Otherwise, simple binary encoding is used. Encoding of numbers spanning multiple octets is done in a way that the most significant bit of the encoded value is located in the first octet of the value (network byte order).

F.10.1 Open Logical Channel

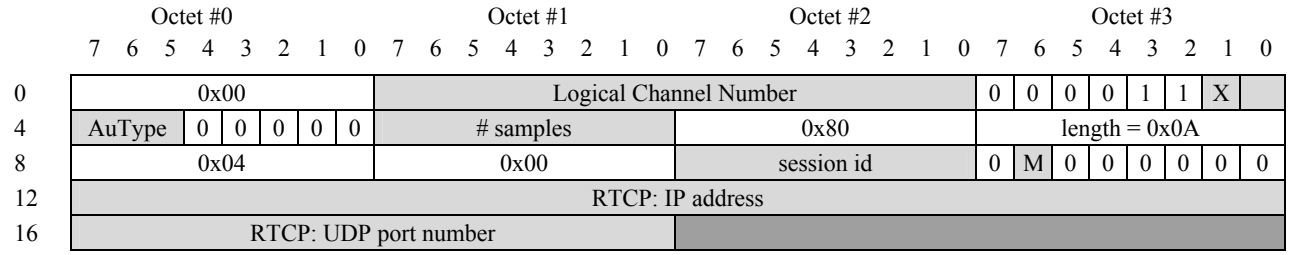
The **OpenLogicalChannel** structures are used by SET devices during the FastConnect procedure to indicate their capabilities and simultaneously open media channels in both directions and to reconfigure media streams during a conference. By definition, the **OpenLogicalChannel** structures contain only either forward logical channel parameters or backward logical channel parameters.

F.10.1.1 Forward Logical Channel parameters

An Open Logical Channel structure containing only **ForwardLogicalChannel** parameters may be coded in three different ways, depending on the audio type (AuType) and the X bit.

F.10.1.1.1 ITU-T Recs G.711 and G.729

The most common structure is the following (ITU-T Recs G.711, G.729 and Annex A/G.729):



Logical Channel Number: This field contains the number of the H.245 logical channel – 1.

X bit: Used to distinguish between basic and extended audio types. If X = 0, AuType (see next field) applies; otherwise (X = 1), the extended audio types described below apply (primarily GSM) along with a different packet structure.

AuType: Identifies the audio codec to be used. The following values are acceptable for AuType. The leftmost bit is placed in bit 1 of octet #3 above, the rightmost in bit 5 of octet #4.

No.	Codec description	AuType value
1	G.711 A-law 64 kbit/s	0001
2	G.711 A-law 56 kbit/s	0010
3	G.711 μ-law 64 kbit/s	0011
4	G.711 μ-law 56 kbit/s	0100
5	G.723.1	1000
6	G.729	1010
7	Annex A/G.729	1011
8	GSM and others (see below)	X = 1

samples: For codecs 1, 2, 3, 4, 6, and 7 this component contains the number of samples – 1 per audio packet as defined in ITU-T Rec. H.245.

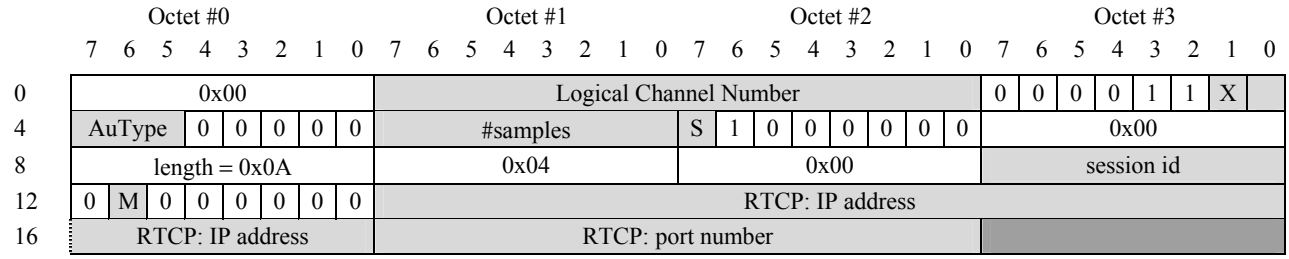
session id: Contains the session id parameter to be used in conjunction with RTP/RTCP.

M bit: Multicast address bit: indicates that the following address is a multicast address. While many address types are defined besides IPv4 (including IPv6 and IPX), the structures shown here are only valid for IPv4 addresses.

RTCP IP address/port: Contains the transport address for the RTCP receiver reports to be sent to.

F.10.1.1.2 G.723.1 codec

For ITU-T Rec. G.723.1, the structure is slightly modified as follows:

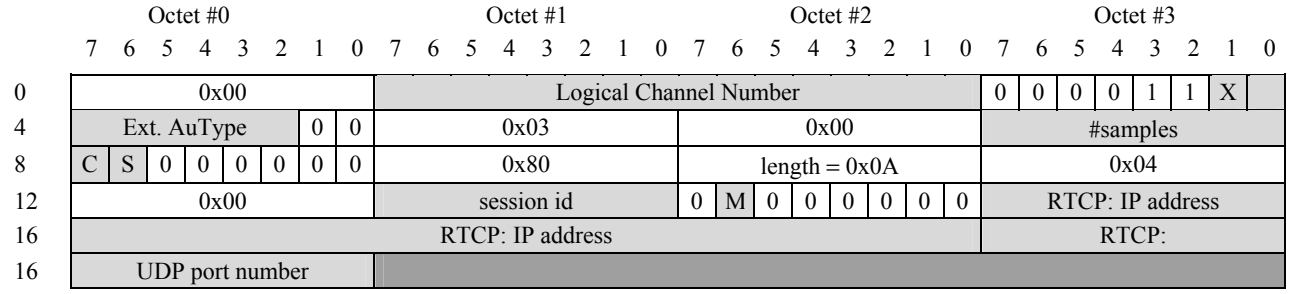


The meaning of the fields is identical to the meaning defined for the above format. In addition, the following fields are relevant:

S bit: Indicates support for silence suppression if S = 1.

F.10.1.1.3 GSM

For GSM, identified by bit #1 of octet #3 set to X = 1, the structure looks as follows:



The fields have the same meaning as in the above packet formats. In addition, the following fields are defined for GSM:

Ext. Audio Type: Identifies the extended audio codec:

GSM Full Rate = 000 0011

GSM Half Rate = 000 0100

GSM Enhance Full Rate = 000 0101

C bit: C = 1 indicates support/use of comfort noise

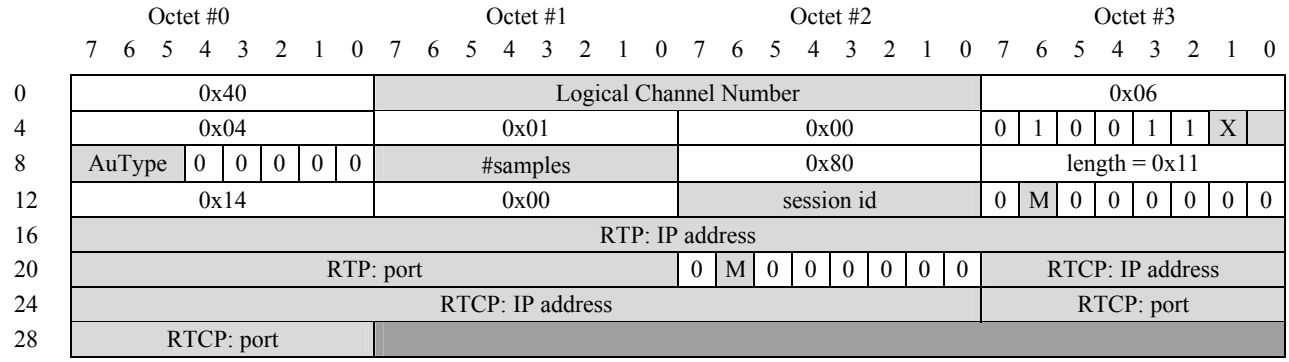
S bit: S = 1 indicates support/use of scrambling

F.10.1.2 Reverse Logical Channel Parameters

Open Logical Channel Message containing **ReverseLogicalChannel** parameters are encoded as described in this subclause.

F.10.1.2.1 ITU-T Recs G.711 and G.729

The most common structure is the following (ITU-T Recs G.711, G.729 and Annex A/G.729):



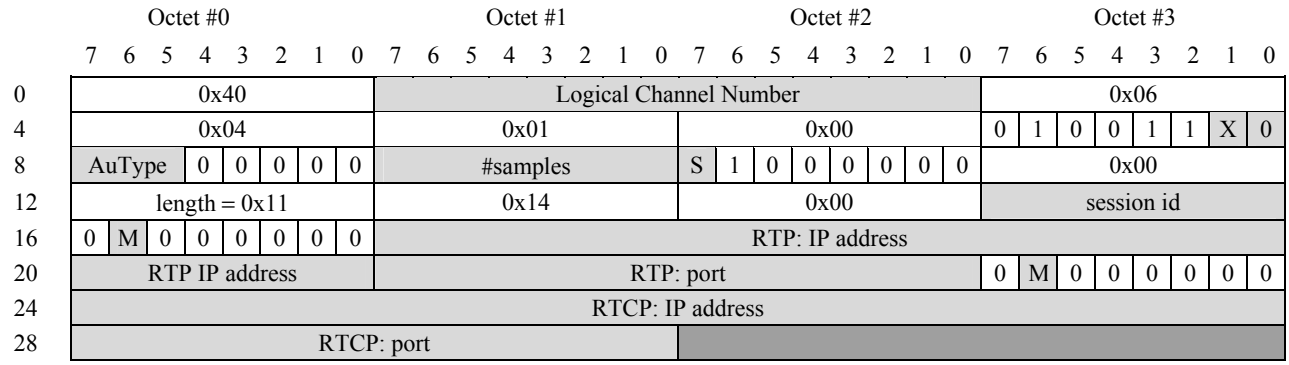
The fields have the same meaning as above. In addition, the following fields are defined:

RTP IP address/port: Target transport address for the RTP audio stream to be sent to.

RTCP IP address/port: Target transport address for RTCP sender reports to be sent to.

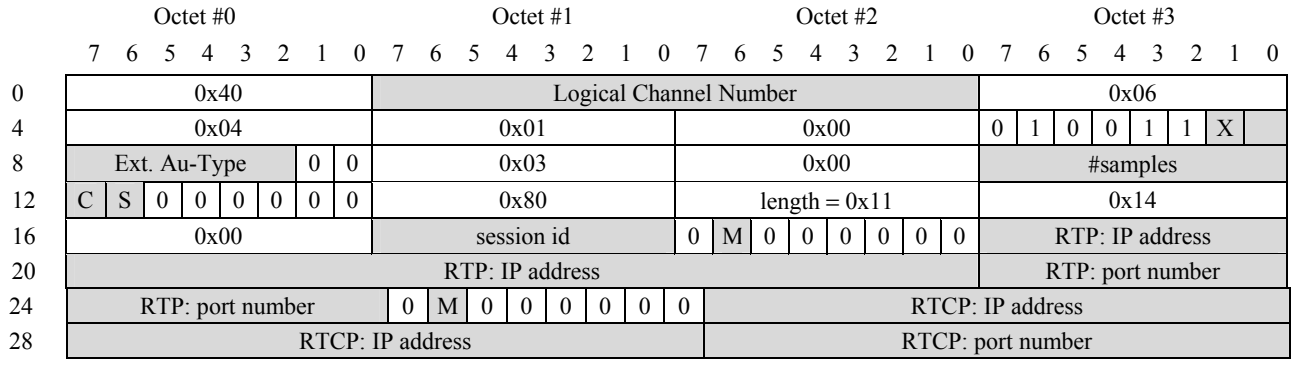
F.10.1.2.2 ITU-T Rec. G.723.1

For ITU-T Rec. G.723.1, the structure differs slightly from the above as follows:



F.10.1.2.3 GSM

For GSM, identified by bit #1 of octet #7 set to X = 1, the structure looks as follows:



Ext. Au Type: Identifies the extended (GSM) audio codec to be used as follows:

GSM Full Rate = 000 0011

GSM Half Rate = 000 0100

GSM Enhance Full Rate = 000 0101

Annex G

Text conversation and Text SET

G.1 Introduction

Standardized, character-oriented text conversation facilities are needed in all networks. When building text conversation facilities on multimedia protocols, an opportunity is created to use any combination of text, video and voice in a conversation. The initiative to standardize this combination comes from the needs of people with communication-related disabilities. The availability of the three media in a conversation offers communication opportunities over any one of the media alone. Anyone may find a commonly available, standardized text conversation addition to multimedia conversation services valuable, enhancing videotelephony to "Total Conversation".

Since H.323 is a framework, where components can be included when required, single function text terminals as well as text and voice terminals can be useful subsets of the full Total Conversation terminal. These subsets correspond to text telephones available for the PSTN.

ITU-T Rec. T.140 [G1] specifies a text conversation protocol. It is a common presentation level suitable for straightforward real-time text conversation in multimedia services and in text telephony. It is based on the ISO/IEC 10646-1 character code so as to be suitable to any language. It is introduced throughout the H-series multimedia protocols.

This specification describes how text conversation facilities are added to the H.323 multimedia environment in packet networks.

The text conversation facility is established in a data channel identified by the H.245 **OpenLogicalChannel** message. The same identification is used for opening text conversation channels in H.324. Only the protocol and procedures of the data channel to carry T.140 data differ.

Thereby, Total Conversation gets a uniform implementation across different networks. The complexity of gateways and other network components can be kept low.

G.2 Scope

The scope of this annex is to specify H.323 procedures to establish and carry text conversation sessions in real time over packet networks in the H.323 multimedia environment. It also specifies rules on the use of H.323 that enable Text Conversation Simple Endpoint Type Devices (Text SET) to be created as supersets of the Audio Simple Endpoint type devices specified in Annex F/H.323. The Text SET specification describes a device that can be used for real-time conversations in voice and text simultaneously over packet networks.

G.3 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[G1] ITU-T Recommendation T.140 (1998), *Protocol for multimedia application text conversation,* plus amendment.

[G2] HELLSTRÖM (G.): RTP Payload for Text Conversation, *RFC 2793*, *Internet Engineering Task Force*, 2000.

G.4 Definitions

This annex defines the following terms:

G.4.1 total conversation: Conversational services offering real-time communication in video, text and voice.

G.4.2 T140PDU: Protocol Data Unit from T.140 = a collection of data submitted in T.140 format for transmission.

G.5 Procedures for opening channels for T.140 text conversation

The session requirements of T.140 are reflected in the following specification for the channel setup using the H.245 Open Logical Channel Message structure in the H.323 environment.

A reliable or unreliable channel may be selected to carry the T.140 session. The unreliable channel shall always be supported. The unreliable channel may be selected for cases when the terminal is expected to participate in sessions where a reliable channel is unfavourable or impossible to use. The reliable channel is a preferred option.

- In the capabilities exchange, when using a reliable channel, specify:

  ```
  DataApplicationCapability.application = t140
  DataProtocolCapability = tcp
  ```

- In the capabilities exchange, when using an unreliable channel, specify:

  ```
  DataApplicationCapability.application = t140
  DataProtocolCapability = udp
  ```

- In the Open Logical Channel procedure, specify:

  ```
  OpenLogicalChannel.forwardLogicalChannelParameters = dataType
  DataType = data
  ```

  And select a reliable or unreliable channel for the transfer of T.140 data by specifying the `DataApplicationCapability` and the `DataProtocolCapability` as above.

The fast-start or the normal procedures may be used.

The destination node and originating node concepts of ITU-T Rec. T.140 are mapped to the two H.323 endpoints.

The T.140 user identity is an alias for the far H.323 endpoint.

G.6 Framing and buffering of T.140 data

Transmission of T.140 data shall be done according to the following specifications, different for the reliable and the unreliable channel.

G.6.1 Common considerations

T.140 data may be collected in a buffer before transmission in the channel. On low bit-rate channels, such buffering is recommended in order to reduce packet overhead. Buffering of data in 0.3-second intervals is recommended as default.

On reception, the data contents of the data channel is retrieved and used as T.140 data.

G.6.2 Usage of reliable channels

When a reliable channel is selected for T.140 transmission, TCP is used, and T.140 data is transmitted in the channel without further framing.

G.6.3 Usage of unreliable channels

When an unreliable channel is specified for the T.140 transmission, RTP is used. The details of the RTP payload format "T140" is found in [G2]. The recommended procedures described in [G2] should be used. The payload type allocation is dynamic. For the plain "T140" payload format, Payload Type 96 is used. For the payload type "RED" with redundancy, Payload Type 98 is used.

The procedures offer the possibility to include a number of already transmitted T140PDUs in the packet. This is done in order to include redundant data to reduce the risks of data loss.

The transmitting station may select a number of T.140 PDU generations to retransmit in each packet. A higher number introduces better protection against loss of text. If network conditions are not known, it is recommended to use two generations. It is recommended to use not more than six generations.

RTCP should be used to monitor packet loss, so that a decision can be made on the number of generations of redundant data to transmit.

G.7 Interaction with text conversation facilities in other devices

The information in this clause is not normative and is provided for information only, beyond the scope of this annex.

ITU-T Rec. T.140 is established as the text conversation protocol throughout a series of H-series multimedia protocols, T.120 data conferencing and for ITU-T Rec. V.18 text telephones. The data channels are specific to each environment.

When gateways to these different environments are established, the T.140 channel in the H.323 environment is mapped into the T.140 channel in the other environment. The T.140 channel data can be transparently transferred through the gateway.

When gateways to other text conversation protocols are established, the data and protocol mechanisms of that protocol shall be mapped into a T.140 text conversation channel in the gateway. Such mapping functions can be called T.140 equalizers. Gateway functions to the different text telephone systems involve T.140 equalizers.

Figure G.1 gives an overview of text conversation protocols and gateway services.

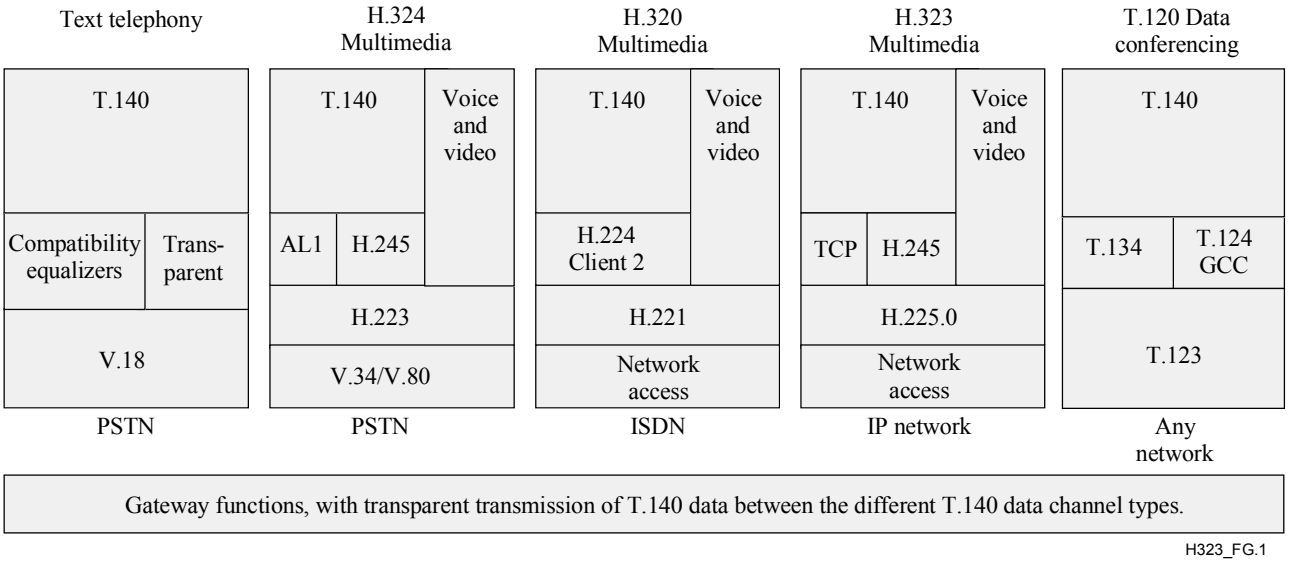


Figure G.1/H.323 – Multimedia real-time text conversation Recommendations and interworking needs

G.8 Multipoint considerations

Without further specification, three alternative options exist for H.323 endpoints with T.140 text conversation to participate in multipoint text conversations.

Alternatives:

- One separate T.140 channel is set up for each remote H.323 endpoint. The text streams can be coordinated for display through a multipoint-aware user interface, that also transmits T.140 data to all connected endpoints.
- An MCU coordinates the T.140 data stream to the H.323 endpoint to contain data from a number of remote endpoints.
- Instead of the procedures described in this annex, the T.134 application member of T.120 data conferencing is used as the channel for T.140 data. Multipoint sessions are coordinated through the T.120 concepts.

G.8.1 Situations for multipoint text conversation

In order to clarify the use of text conversation, and especially the different multipoint cases, the following examples of possible setups and applications are given without being normative.

G.8.1.1 One-to-one

The one-to-one case represents a direct conversation in text between two parties, where the text entered at one endpoint is displayed character by character or in small groups of characters as they are entered at the other end. Typical examples are situations like the traditional text telephony in PSTN and multimedia conversation applications with video, text and data used for person-to-person calls. See Figure G.2.

Anne	Eve
Hi, this is Anne. Have you heard that I will come to Paris in November?	Oh, hello Anne, I am glad you are calling! No, that was new to me. What brings you here?

Figure G.2/H.323 – Possible display of a one-to-one text call

G.8.1.2 Many-to-many

All users have write permission, forming an unmanaged conference.

The display can be arranged as specified in ITU-T Rec. T.140 with one window for each participant. See Figure G.3.

Anne	Eve
Hi, this is Anne. Have you heard that I will come to Paris in November?	Oh, hello guys! How are you Steve?
Steve	**Bill**
Hi there! This is Steve, I'm fine.	Hello Anne! I am happy that you are on the big Internet!

Figure G.3/H.323 – Possible display of an unmanaged four-to-four text session

The display of a many-to-many conference can also be ordered in one window with labels for each participant's entries (IRC style) (see Figure G.4):

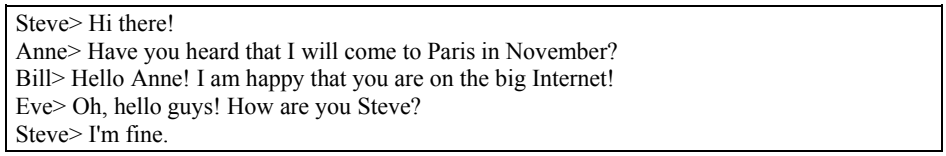


Figure G.4/H.323 – Possible display of an unmanaged four-to-four text session IRC style

G.8.1.3 One-to-many with managed right to type

One writer at a time is given the right to transmit text to many readers. The right to type may be passed to other writers, in a managed meeting.

Typical application is in distance education when the teacher normally has the right to type, but can hand it over to a participant.

G.8.1.4 One-to-many with fixed right to type

One writer types text in the session from one fixed endpoint, the other endpoints display the text in a receiving window. The right to write cannot be transferred.

Typical application is found in subtitled speeches.

The user terminals may be H.332 loosely coupled endpoints.

See Figure G.5.

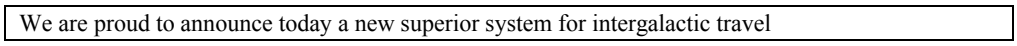


Figure G.5/H.323 – Example of one-to-many text session

G.9 Text SET: Text Conversation Simple Endpoint Type

This part of the annex specifies Text Conversation Simple Endpoint Type Devices that operate using a well-defined subset of H.323 protocols. They are well suited for IP Text Telephony applications while retaining the interoperability with regular H.323 Version 2 (1998) devices. The specification adds real-time text conversation facilities as specified in ITU-T Rec. T.140 to the simple IP-voice telephone as specified in Annex F/H.323, to form the IP-text telephone with simultaneous voice and text functionality.

G.9.1 Introduction to Text SET

The procedural and protocol details of a Simple Endpoint Type Text Telephone Device for IP networks is defined in terms of modifications and additions to the Audio SET specification found in Annex F/H.323. The device here is called Text SET.

The general SET concepts are described in Annex F/H.323. This is a set of modifications to the Audio SET specification that comprises what is needed to add text conversation functionality to the Audio SET. This annex indicates the clause numbers of the original.

G.9.2 Text SET System Functionality Overview (F.6/H.323)

In **Media capabilities**; modify:

- Data-capability mandatory; T.140.

G.9.3 Procedures for Text SET devices (F.7/H.323)

Modify the Media packetization and transport to:

- Media packetization and transport (H.225.0, RTP, TCP, T.140) – See F.7.4/H.323.

G.9.4 RAS Signalling (H.225.0 RAS – F.7.1/H.323)

As for Audio SET, but a SET H.225.0 endpoint type code booked for Text SET is used.

Bit 2 = 1 Indicates that the device has Text SET capabilities.

Bit 2 = 0 Indicates that the device has no Text SET capabilities.

NOTE – The Gatekeeper protocols must be designed so that they will allow voice-only sessions with a Text SET device.

G.9.5 Call Signalling (H.225.0 Call Control – F.7.2/H.323)

SET H.225.0 endpoint type code bit 2 is used to indicate a Text SET function.

G.9.6 Data Capability (F.7.3.3.3/H.323)

Data capability T.140 shall be specified.

DataApplicationCapability.application = t140.

G.9.7 Additional rules for usage of capabilities (F.7.3.3.9/H.323)

Audio and data capabilities shall only be signalled via the FastConnect procedure and repeated exchange of **OpenLogicalChannel** structures using the FastConnect.

Video capabilities, conference capabilities, security capabilities, and h233 encryption capabilities shall not be used.

The values of the **MultiplexCapability** table entry shall be assumed as for Audio SET with the following exceptions:

```
mediaDistributionCapability
centralizedDataTRUE
distributedDataTRUE/FALSE as appropriate, default FALSE
```

G.9.8 Logical channel signalling messages (F.7.3.4/H.323)

Add in the **OpenLogicalChannel** request.

```
OpenLogicalChannel.forwardLogicalChannelParameters.DataType.data = t140
MultiplexParameters     as appropriate for the selected reliable or
                        unreliable channel type.
```

G.9.9 Media exchange (F.7.4/H.323)

For text exchange, SET terminals shall follow the procedures specified in this annex.

G.9.10 Initiating side (F.7.6.1/H.323)

Add:

The **OpenLogicalChannel** structure should offer the same data encoding for text that were offered in the initial call.

G.9.11 Conference-unaware Text SET terminals (F.7.7.1/H.323)

Add the following functionality points:

- Merging several incoming text sessions to the Text SET device.
- Translating the transport addresses for the text stream.
- Transferring and possibly transcoding text data streams.

G.9.12 Support for loosely-coupled conferences (ITU-T Rec. H.332) (F.7.8/H.323)

A Text SET device can participate in a Loosely-coupled Conference using the H.332 procedures provided that the conference is expanded to include text, and that the channel for text transmission is selected to use an unreliable channel.

Annex J

Security for H.323 Annex F

J.1 Introduction

This annex describes security for H.323 Annex F simple endpoint types. The specified security profile is based upon H.235v2 and uses the featured baseline security profile of H.235 Annex D. The shown security profile in H.323 Annex J adopts ITU-T Rec. H.235 for the purpose of simple endpoint types and their specific security requirements. The security profile selects appropriate security features from H.235 with its rich set of options.

The described text provides an overview on the security profile; H.235v2 Annex D provides all the technical and implementations details.

Basically, a **security simple endpoint type** (**security SET**) is a SET as defined by H.323 Annex F that implements additionally certain security features of this annex.

Currently, this annex focuses only on a "secure audio SET (SASET)" and leaves any other security simple endpoint types (e.g., secure FAX SET, secure text terminal, secure Video SET, etc.) for further study.

J.2 Specification conventions

Some explanation is useful for understanding the terms used in this annex:

The annex applies the **baseline security profile** for a SASET (**secure audio simple endpoint type**). The baseline security profile provides basic security by simple means using secure password-based cryptographic techniques; the functionality provided should be implemented by each SASET. The baseline security profile may use the **voice encryption security profile** for achieving voice confidentiality if necessary. It is for further study, whether there will be other, more sophisticated security profiles for SASETs.

In order to avoid references to a trademark (RC2®), this annex actually references an "RC2-compatible" encryption algorithm.

This annex uses well-known security terms as key, key management and SET, which have different meanings in other contexts (e.g., touch key pad, Q.931/Q.932 feature key management, and Secure Electronic Transaction protocol).

J.3 Scope

This annex describes security for simple endpoint types. As shown in F.3, this currently includes:

- **Secure simple telephone terminal** (Secure Audio Simple Endpoint Type) – Defined in this annex (see J.6).

Any other security SETs are for further study.

J.4 Abbreviations

This annex uses the following abbreviations:

DES Data Encryption Standard

GK Gatekeeper

HMAC Hashed Message Authentication Code

ITU International Telecommunication Union

MAC Message Authentication Code

RAS Registration, Admission & Status

RTP Real Time Protocol

SASET Secure Audio Simple Endpoint Type

SET Simple Endpoint Type

SHA Secure Hash Algorithm

J.5 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

– ITU-T Recommendation H.235 (2000), *Security and encryption for H-series (H.323 and other H.245-based) multimedia terminals*.

– ITU-T Recommendation H.245 (2003), *Control protocol for multimedia communication*.

– IETF RFC 2268 (1998), *A Description of the RC2® Encryption Algorithm*.

J.6 Secure Audio Simple Endpoint Type (SASET)

This annex describes a baseline for **secure audio simple endpoint types (SASETs)**. An example of a SASET is a secure simple phone.

J.6.1 Assumptions

The baseline security profile mandates the GK-routed model for secure H.323 Annex F SETs. SASETs and other H.323 entities that implement this security profile (e.g., GKs) are assumed to implement the fast connect procedure.

In accordance to Annex F the baseline security profile mandates the fast connect procedure with integrated key management elements but does not support H.245 tunnelling. Thus, the baseline profile does not provide means for key update and synchronization using (tunnelled) H.245 messages. SASETs implementing only the baseline security profile but still need some key-update mechanism should hangup the call and reconnect and thereby obtain a new session key.

J.6.2 Overview

The baseline security is applicable in administered environments with symmetric keys/passwords assigned among the entities (SASETs-gatekeeper, gatekeeper-gatekeeper).

Table J.1 summarizes all the procedures defined in H.235v2 Annex D.

Table J.1/H.323 – Summary of Secure Audio Simple Endpoint Types (see H.235v2 Annex D)

Security Services	Call functions					
	RAS	H.225.0		H.245 (Note)	RTP	
Authentication	*Password HMAC-SHA1-96	*Password HMAC-SHA1-96		*Password HMAC-SHA1-96		
Non-Repudiation						
Integrity	*Password HMAC-SHA1-96	*Password HMAC-SHA1-96		*Password HMAC-SHA1-96		
Confidentiality					♦56-bit DES	♦56-bit RC2-compa-tible / ♦168-bit Triple-DES
Access Control						
Key Management	*Subscription-based password assignment	*Subscrip-tion-based password assignment	♦authen-ticated Diffie-Hellman key-ex-change	♦Integrated H.235 session key management (key distribution, key update using 56-bit DES/56-bit RC2-compatible/ 168-bit Triple-DES)		

* Blue area: Password-based scheme

♦ Green area: Voice encryption security profile

NOTE – Embedded H.245 inside H.225.0 fast connect.

For authentication and integrity, the user shall use a password-based scheme (blue area in Table J.1). The password-based scheme is highly recommended for authentication due to its simplicity and ease of implementation. Hashing the fields in the H.225.0 messages is the recommended approach for integrity of the messages (also using the password scheme). SASETs realize authentication in conjunction with integrity using the same common security mechanism.

SASETs when deploying the voice encryption security profile (green area in Table J.1) shall implement 56-bit DES as the default encryption algorithm; SASETs may implement 168-bit Triple-DES while SASETs implementing exportable encryption may implement 56-bit RC2-compatible.

For voice confidentiality, the suggested scheme is encryption using RC2-compatible, DES or Triple-DES based on the business model and exportability requirement. Some environments that are offering already a certain degree of confidentiality may not require voice encryption. In this case, Diffie-Hellman key agreement and other key management procedures are not necessary as well.

Access control means are not explicitly described; they can be implemented locally upon the received information conveyed within H.235 signalling fields (ClearToken, CryptoToken).

This Recommendation does not describe procedures for subscription-based password/secret key assignment with management and administration. Such procedures may happen by means that are not part of this annex.

SASETs may use back-end services according to the procedure described in H.235v2 Appendix I.4.6.

Annex K

HTTP-based service control transport channel

K.1 Introduction

This annex describes an optional way of controlling supplementary services in an H.323 environment. By opening a separate connection conveying a service independent control protocol, new services may be developed and deployed without updates to the H.323 endpoints.

This service control channel is intended to be used for a wide range of services, some which require the use of H.450 or proxy signalling (e.g., as in Appendix III) for invocation/execution. As this channel is service independent, no specific services are defined or advocated. The data exchanged on this channel are meant to be informative (user interface) and should be followed by appropriate actions (e.g., H.450 invocations) in the call signalling plane when needed. Although some serverside applications need to support H.450 services for interworking, this annex is totally independent of the H.450.x Recommendations.

The service control channel may be utilized for both call-related and non-call-related services. It may be opened between the terminal and the network, or between two endpoints (in a call or with a call independent connection).

While several protocols might be used, this annex describes the use of the hypertext transfer protocol (HTTP) for this purpose. HTTP is open, flexible, firewall friendly and well known. Any device claiming to support Annex K shall support HTTP as a transport for service control, optionally also S-HTTP for applications requiring security. The actual service application protocol is dynamic, and is indicated using MIME types in the HTTP signalling. Example applications may include XML pages possibly including Java™ and scripts, download of tones and announcements to be played to the client, upload of Call Processing scripts from client to a gatekeeper, etc. While this annex focuses on user directed supplementary services, this service control channel could also be used for other means. It could, for example, be used for software upgrades or for pushing commercials to the clients.

Clause K.2 describes the use of H.323 for providing the HTTP connection's URL between the service provider and the client, clause K.3 shows the use of HTTP, and clause K.4 shows some examples of possible services and the corresponding signalling.

The interface between the service control plane and the call control plane on the client or the service provider is not within the scope of this annex, but could include HTML or XML tags such as mailto or H.323 URLs. See Figure K.1.

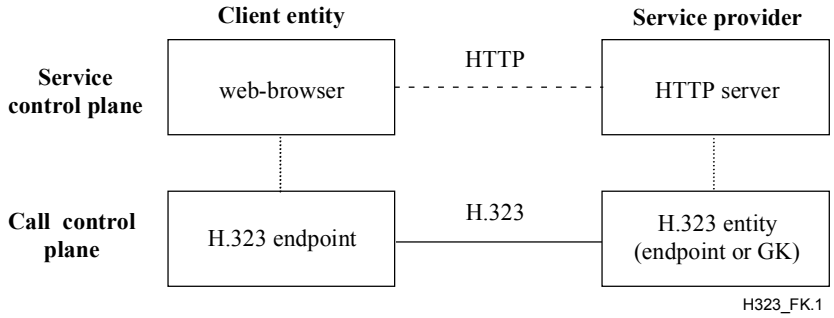


Figure K.1/H.323 – System overview for HTTP-based service control

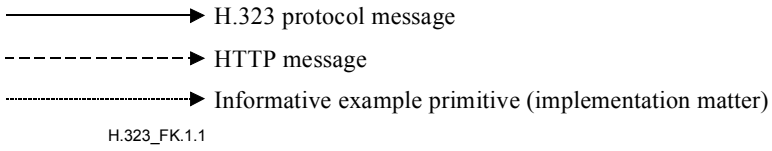
It is generally up to the provider of the URL to define and implement the control functions and services that are being presented by the given URL (standard or non-standard services may be supported). If the service control interacts with H.323 call processing, the provider of this URL should make the binding between the HTTP service and the H.323/H.450 services that are being supported by the gatekeeper or endpoint.

As the HTTP service control channel is stateless and unaware of the services in scope, it cannot take service interaction problems into account. An application that utilizes this service control channel should, however, consider this carefully.

Any sequence charts or references to H.323 signalling in this annex are informative examples for describing possible interactions with service control and call control. They do not redefine H.323 signalling rules, as most are greatly simplified for briefness.

K.1.1 Notation

The following notation is used:



HTTP and RAS messages are capitalized (HTTP:GET, RAS:ARQ), while H.225.0 call signalling messages are written with the first letters capitalized (Setup). ASN.1 codepoints in H.225.0 are written in bold (**ServiceControlAddress**).

K.2 Service control in H.323

This clause describes how H.323 messages are used for maintaining the service control sessions.

K.2.1 Service control session

A service control session is a one way relation between the client entity and the service provider, in this case an HTTP session. It is initiated from the client after the receipt of a **ServiceControlAddress** URL in H.225.0 messages. The URL may be received through two different H.323 signalling channels:

- A **ServiceControlSession** structure containing a URL is received in a message over the RAS channel. If there is no appropriate message to send, the **ServiceControlIndication** (SCI) message may be sent to the endpoint at any time.
- A **ServiceControlSession** structure containing a URL is received in a message on the H.225.0 call signalling channel.

The service control session is identified with a **sessionId**, a unique number for the signalling channel. The **sessionIds** received through RAS and call signalling may overlap, as the senders of these may not be aware of each other.

A service provider wishing to initiate a new service control session does so by sending a **ServiceControlSession** structure to the client. It contains a new **sessionId**, the URL for the service, and the reason field set to "**open**". The client may open a connection to this address and request the resource from the URL, but no acknowledgement is given from the client in the call signalling plane. If the user wishes to end the session at any time, e.g., by closing a pop-up window for the session, this is done without any notification to the provider.

If a service provider needs to notify an endpoint about new services or events relating to a previously opened session, it may do so by issuing a new **ServiceControlSession** structure on the RAS or call signalling channel (as was used in the "open" sequence). The structure shall contain the same **sessionId** as before (to reuse the same resource, e.g., screen window), a new URL to be loaded, and the reason set to "**refresh**".

If the service provider needs to terminate the session, it may send a **ServiceControlSession** structure with the same **sessionId** and reason set to "**close**". The client should, if it still has the session open, close any resources such as windows dedicated to the session.

The reason for the support of multiple sessions is that unrelated service provider nodes may use the same notification mechanisms, e.g., the call signalling channel. Service applications utilizing this annex should take care not to overuse the number of sessions, as many notifications quickly will confuse an end user. Clients supporting this annex are not required to support more than two sessions, one call related and one non-call related.

K.2.2 Non-call-related service control

To provide services relating to the registration session, and not a given call, the gatekeeper may return a **ServiceControlSession** structure containing a URL in the RCF message. The returned URL should be complete in terms of defining protocol, server and resource, i.e., <protocol>://<server-address>/<resource>. The endpoint may load this URL and display the services and service control functions as provided by the data given by this URL (e.g., a web-page with menus and links).

If the network needs to notify the endpoint about service related events, during a call or as part of the registration, it can issue a Service Control Indication (SCI) with a URL to this endpoint. To indicate that this URL relates to an already active non-call-related service control session, the **sessionId** shall be the same as previously and the **callSpecific** field shall not be present. The endpoint may then load this URL and be provided with updated services and service control functions. An endpoint that receives such a SCI shall respond with a Service Control Response (SCR) message to avoid retransmissions of the SCI from the provider. The SCR message is only an acknowledgement of the receipt of the SCI message, and not necessarily an application level response.

The Service Control Indication message may also be used to open a new session or to close the session.

If an entity other than the local gatekeeper wishes to open a call unrelated service control session towards an endpoint, this can be done by opening a call independent signalling connection towards the endpoint, and sending a Setup message with a **ServiceControlSession** structure including an URL. The **conferenceGoal** parameter shall be set to **callIndependentSupplementaryService** and the bearer capability information element of the Setup shall be set as defined for call independent connection in 7.2.2.1.2/H.225.0. Otherwise the same procedures as in K.2.2 with the **ServiceControlSession** transported in call signalling messages applies, with the absence of media on the connection.

K.2.3 Call-related service control

Two methods are provided to open service control session related to a specific call:

1) A service control session is opened between an endpoint and its gatekeeper with a URL carried in a call-related RAS message, especially for gatekeepers using direct endpoint call signalling. If the SCI message is used the **callSpecific** field of the SCI shall contain the **callIdentifier**, the **conferenceId** and the **answerCall** field as used in previous signalling for this call. A new **sessionId** shall be used. This session should not affect the call unrelated service control session as in K.2.2.

2) Service control sessions are opened between an endpoint and a gatekeeper or between two endpoints with a **ServiceControlSession** field containing a URL in the call signalling messages.

If a service provider needs to notify an endpoint about new services or events in an existing session, it may do so by means of refreshing data on an URL that has been previously loaded (e.g., applet/servlet dialogues), or it can issue a H.225.0 message (Facility or SCI) with a new URL, the reason set to "**refresh**" and the same **sessionId** as earlier for the session. An endpoint that receives such a Facility message should load this URL and render the data presented by it to the same resource (e.g., screen window) as was first used for this session.

If a service providing entity wishes to initiate a new session after the call is connected, it may also use the Facility/SCI message with a **ServiceControlSession** containing a new **sessionId**, the URL in scope and the reason set to "**open**". H.225.0 messages without the **ServiceControlSession** present does not influence the HTTP session, except Release Complete, which without a URL indicates that all sessions for this call are ended. This signalling should be seen separate for all sessions in use (non-call related, call related with SCI and in-call signalling messages).

Gatekeepers that use the HTTP service control should be careful not to interact with end-to-end service control. This is in particular the case for non-gatekeeper routed calls where the gatekeepers are unaware of the call control messages and states. To alleviate this problem, it is recommended that endpoints use separate browser-windows for the different service control sessions. Intermediate devices such as gatekeepers or MCUs utilizing this annex must be aware of the possibility for conflict with other service providing entities along the call signalling path. Messages (call signalling or other, e.g., an LCF with service control data that can be sent to the client in an ACF) may arrive towards the client with a **ServiceControlSession** using the same **sessionId** as already used between the intermediate provider and the service client. If the intermediate device decides to pass on the **ServiceControlSession**, it must be able to map the **sessionId** to a unique number for the client. Another possibility is to multiplex these two sessions into the same presentation level protocol.

To provide call-related services between different zones or domains, a terminating entity may return a **ServiceControlSession** structure containing a URL in other messages than on the call signalling channel (e.g., LCF/LRJ). It is up to the local gatekeeper to forward the **ServiceControlSession** received in corresponding messages (e.g., ACF/ARJ) toward the client. Applications needing detailed call state information, the possibility to perform actions in the call control plane or the possibility to update the session later should not use this mechanism, but rather use the call signalling channel to convey the **ServiceControlSession** structure.

K.3 Usage of HTTP

K.3.1 Non-call-related services control channel

The HTTP protocol is defined in RFC 2068. This clause provides an informative indication of how the HTTP protocol could be employed for the purpose of providing the described service control protocol.

For non-call-related services, the endpoint is provided with an URL that it could retrieve by means of the standard GET method. The data is collected and rendered according to normal procedures for an HTTP user-agent[2]. The following example (Figure K.2) illustrates the flow:

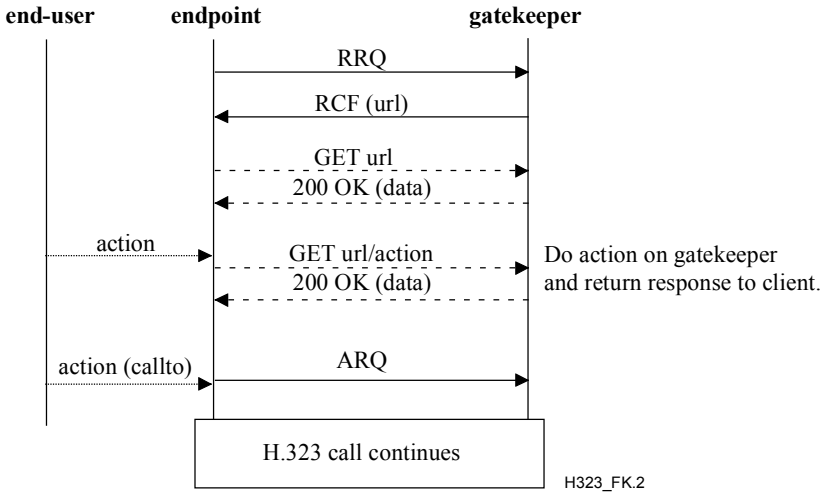


Figure K.2/H.323 – Example of non-call-related service control

K.3.2 Call-related services control channel

In order to support call-related service control a URL is conveyed in different H.225.0 messages as in K.2.3. An endpoint that supports this annex should when it receives such an URL request a standard HTTP user-agent to open and render that URL.

The HTTP user-agent should render the given URL and support style-sheets, scripts, links and images according to that defined for HTTP in RFC 2068. Actions defined and executed by the contents of this URL could be executed locally (e.g., mailto links) or remote on any linked HTTP server, e.g., being implemented or related to an endpoint or a gatekeeper. An example with the endpoint as service provider is given below (see Figure K.3), and gatekeeper service provider is in K.4 – Example 2.

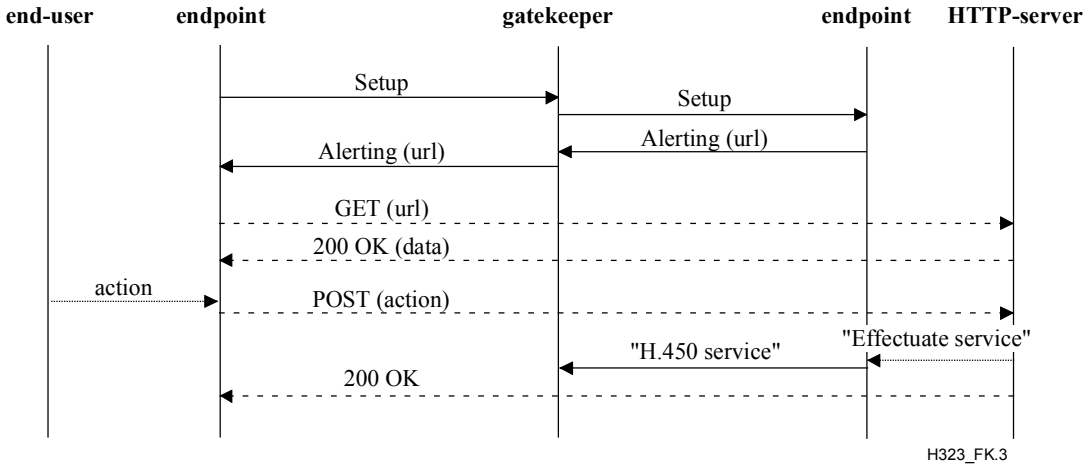


Figure K.3/H.323 – Example of call-related service control using URL in H.225.0 call signalling messages

---

[2] The term "HTTP user agent" used within this annex refers to a process that implements the client part of the HTTP protocol (normally represented with a web-browser).

1) The client sends a Setup that is routed via the gatekeeper to the endpoint representing the called party.

2) The called party may be in a state where specific call processing is programmed, e.g.:
   - Decide to reject the call by sending a Release Complete. The Release Complete could contain a URL to be displayed by an HTTP user-agent on the calling party. The URL could, for example, be a reference to the home page of the called party.
   - Decide to return a list of options for call setup options. In this case it returns Alerting with an URL that defines the options given to the calling party, for example, divert call to operator, secretary, voice-mail, email or intrusion on existing call session.

3) The calling party H.323 endpoint requests an HTTP user-agent to open the URL and the data is then rendered on the web interface of the calling party. The end-user can then dismiss the browser-window or interact with it by selecting a link/action.

4) Actions defined and executed by the contents of this URL could be executed locally (for example, mailto links) or remote on any linked HTTP server, e.g., being implemented or related to the endpoint or the gatekeeper. The remote endpoint or gatekeeper should analyse the given action and effectuate it by means of standard H.323/H.450 services. The result could, for example, be to divert the call to a voice-mail server.

K.4 Example scenarios

To illustrate the usage of the open service control a set of examples are given. These are:

- a simple example of usage of non-call-related service control;
- an example of call-related service control for gatekeeper routed calls;
- an example of call-related service control for non-gatekeeper routed calls;
- an example of non-call-related service control for script upload.

All examples here are only using one simultaneous service control channel. For simplicity, messages containing a **ServiceControlSession** structure are indicated with only the "url".

Example 1: Non-call-related service control

The example illustrates the control signals when a user registers with a gatekeeper, receives back an URL referencing a phone-book, updates the phone-book with a friends contact (alias) and then uses this updated phone book for making a call (these are not necessarily the same friends) by selecting an entry with a H.323 URL. See Figure K.4.

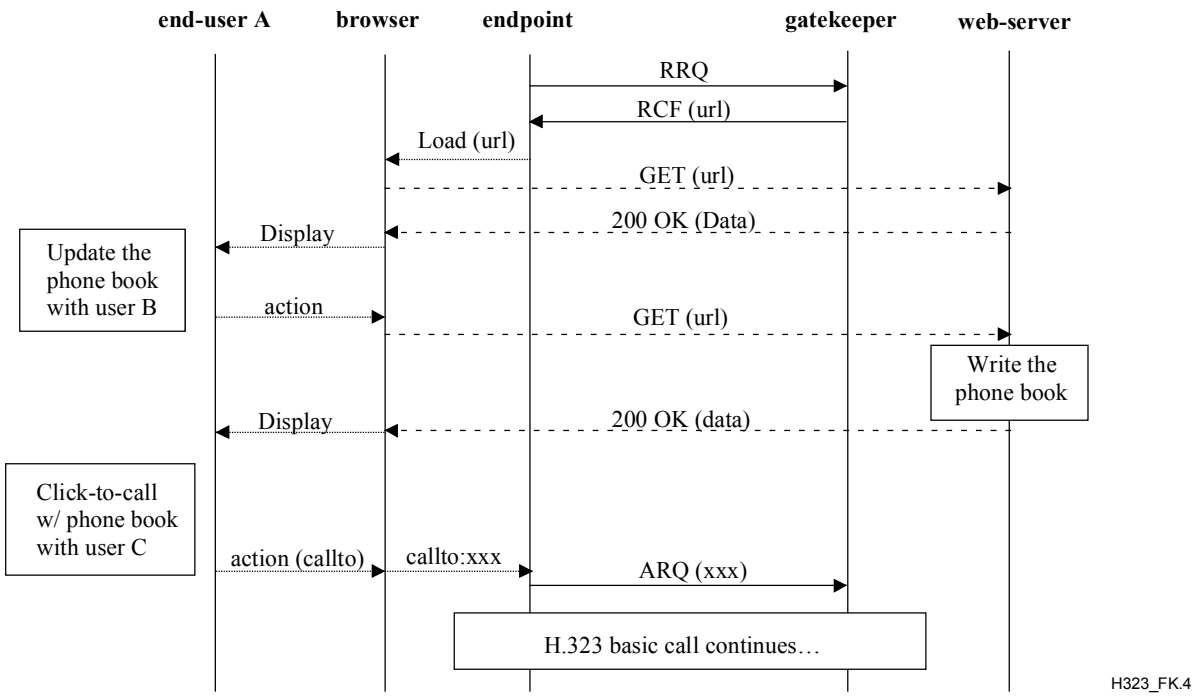


Figure K.4/H.323 – Non-call-related service control

Example 2: Call-related service control, gatekeeper-routed call

The example illustrates a variation of a "Call Waiting Service" with options for the calling party. The gatekeeper detects that the called party is busy and provides an URL to the calling party in an Alerting message (to prevent a timeout at the calling endpoint). The URL references a web page containing a set of options for the further processing of the call.

The user hears the audio alert and a web page with options is presented. The options could be divert to voice-mail, email or operator. The user selects the voice-mail and this selection is signalled to the HTTP-server that informs the endpoint about this.

The gatekeeper effectuates the diversion request as call forwarding on no reply (as Alerting have been sent) and informs the HTTP-server about the successful diversion. The HTTP server then responds to the browser with a new web-page saying, for example, that diversion was completed successfully and giving it some new options. See Figure K.5.

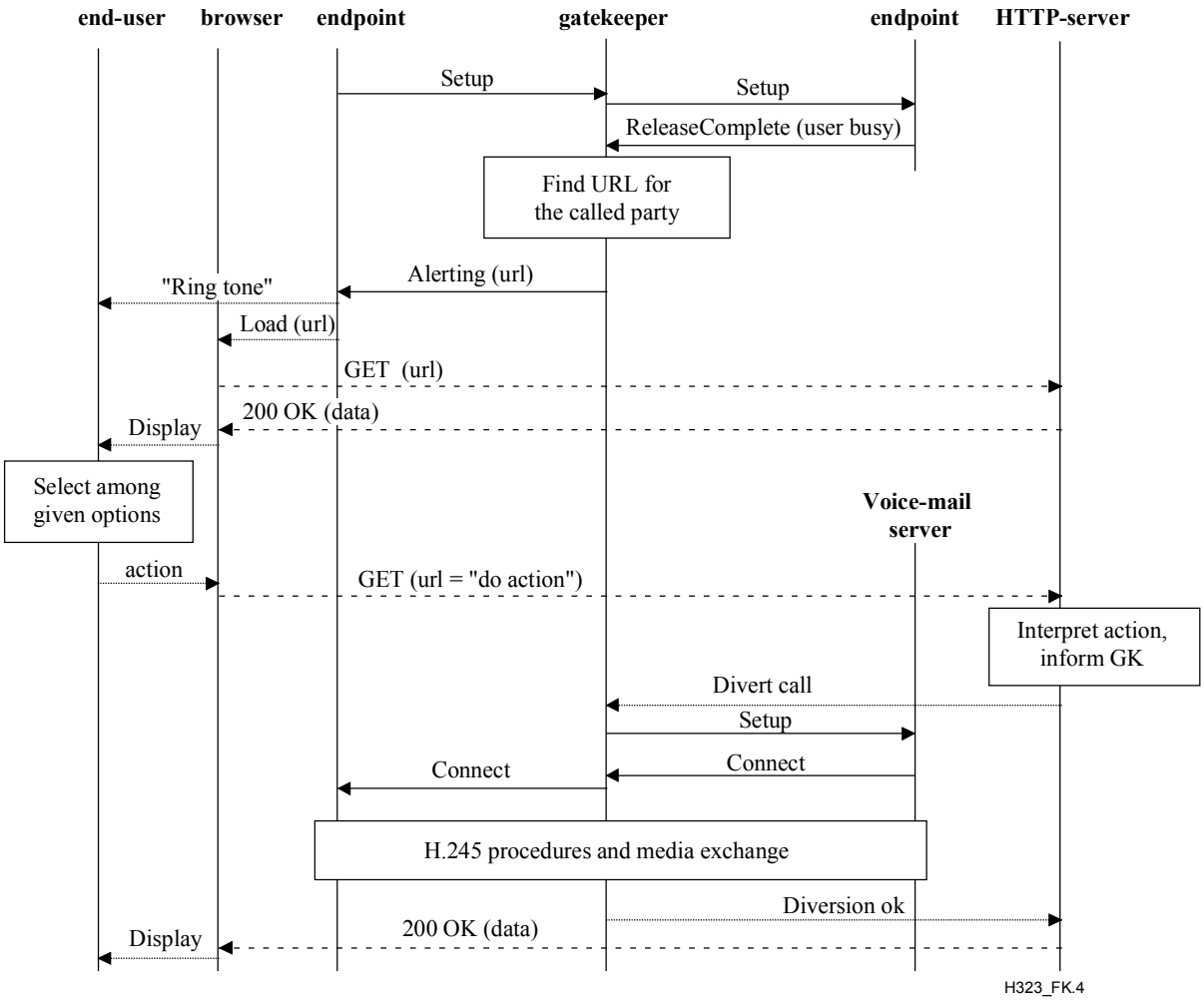


Figure K.5/H.323 – Call-related service control, gatekeeper-routed call

Example 3: Call-related-service control, non-gatekeeper-routed call

The example illustrates the same service as in Example 2, executed at the called endpoint. The called endpoint is busy in a call and returns an URL to the calling party in an Alerting message (to prevent a timeout at the calling endpoint). The URL references a web page containing a set of options for the further processing of the call.

The user hears the audio alert and a web page with options is presented. The options could be divert to voice-mail, email or operator. The user selects the voice-mail and this selection is signalled to the HTTP-server that informs the endpoint about this.

The endpoint effectuates the diversion request as call forwarding on no reply (as Alerting has been sent) and informs the HTTP-server about the successful diversion. The HTTP server then responds to the browser with a new web-page saying, for example, that diversion was completed successfully and giving it some new options. See Figure K.6.

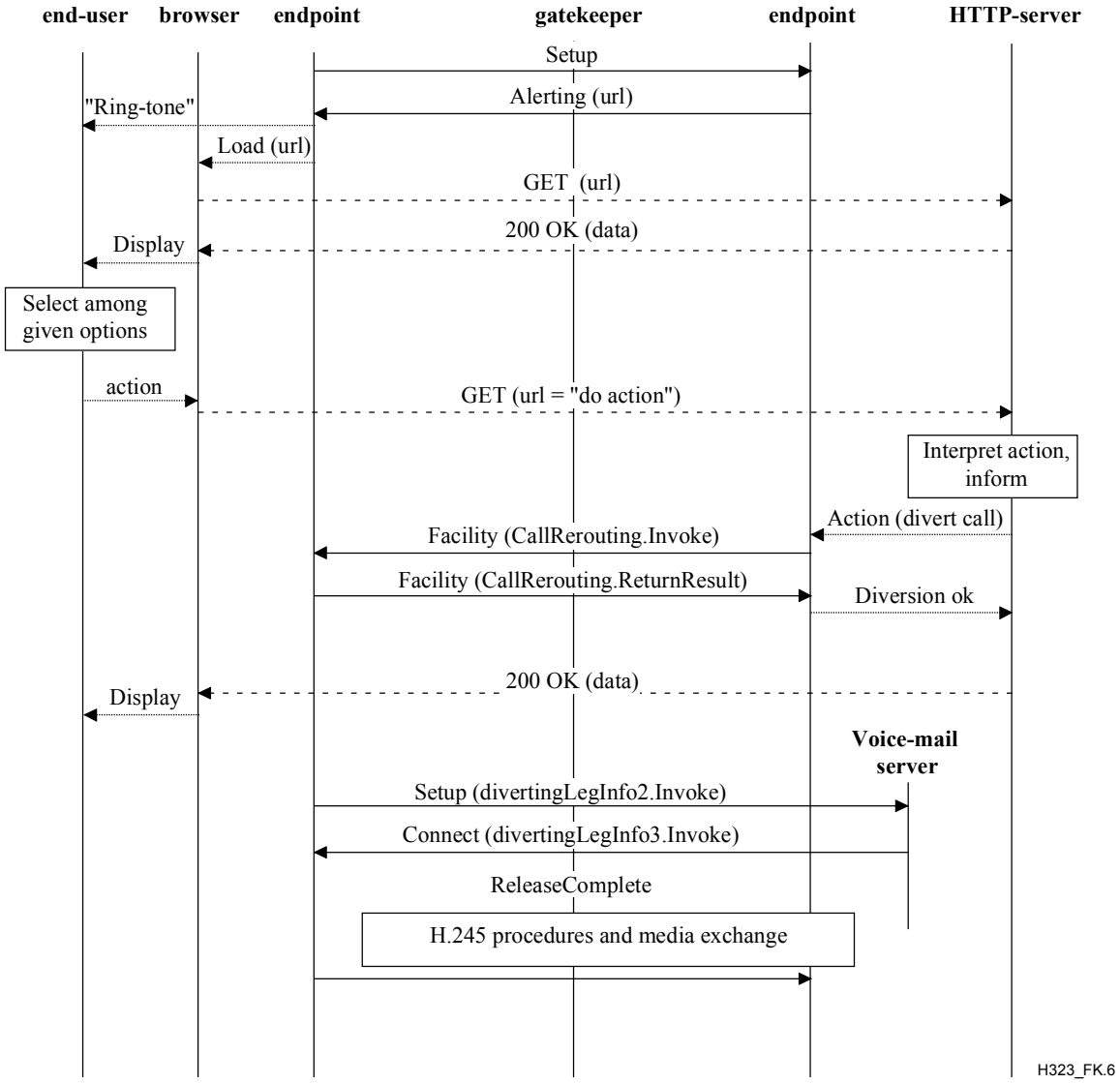


Figure K.6/H.323 – Call-related-service control, non-gatekeeper-routed call

Example 4: Non-call-related service control, script upload

Call processing scripts are also a form of service control. The example shows a terminal uploading a script after registration. The user prepares the script by a graphical builder in the endpoint or by other means, and decides to upload this to the server.

In this case the endpoint knows, when the user decides to upload the script, that it must utilize the POST scheme. The details of the script and impacts on further call signalling is dependent on the script. See Figure K.7.

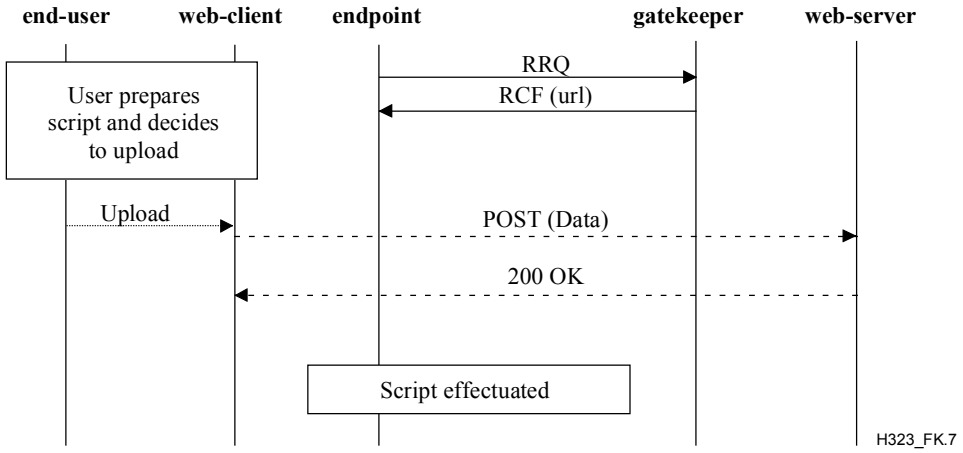


Figure K.7/H.323 – Non-call-related service control, script upload

K.5 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

K.5.1 Normative references

[H2250] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

[URL] BERNERS-LEE (T.) *et al.*: Uniform Resource Locators (URL), *RFC 1738, Internet Engineering Task Force*, December 1994.

[HTTP] FIELDING (R.) *et al.*: Hypertext Transfer Protocol – HTTP/1.1, *RFC 2068, Internet Engineering Task Force*, January 1997.

K.5.2 Informative references

[S-HTTP] RESCORLA (T.) *et al.*: The Secure HyperText Transfer Protocol, *RFC 2660, Internet Engineering Task Force*, August 1999.

[HTML] BERNERS-LEE (T.): Hypertext Markup Language – 2.0, *RFC 1866, Internet Engineering Task Force*, November 1995.

[MIME] FREED (N.), BORENSTEIN (N.): Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, *RFC 2045, Innosoft, First Virtual*, November 1996.

Annex L

Stimulus control protocol

L.1 Scope

This annex describes stimulus signalling procedures between H.323 terminals and a Feature Server functional entity. This stimulus method allows the network service provider to implement new supplementary services for the terminals without changes in the terminal software, which results in easier maintenance. An example of such terminals is a LAN attached feature phone. A Feature Server may be colocated with the Gatekeeper.

The H.323 stimulus protocol allows services to be provided by one or more Feature Servers. For interoperability, standard H.225.0 signalling is used for basic call control, and all manipulations of media streams are done using standard H.245 or Fast Connect procedures. Mechanisms based on ITU-T Rec. H.248 are used to manipulate physical terminations such as speaker or handset.

The protocol described by this annex can support both the direct signalling model and the Gatekeeper routed model.

The typical configurations illustrated by Figures L.1 and L.2 show functional signalling entities that may be involved in a call from an H.323 stimulus terminal to another endpoint in a different H.323 zone. Figure L.1 shows the Feature Server acting as a signalling proxy for the Annex L terminal. Figure L.2 shows the Feature Server colocated with the Annex L terminal's Gatekeeper. In both cases, the Feature Server has access to the H.323 signalling, which provides it with call state information which may be useful for particular services, as well as enabling the Feature Server to affect media streams using H.245 or Fast Connect signalling.

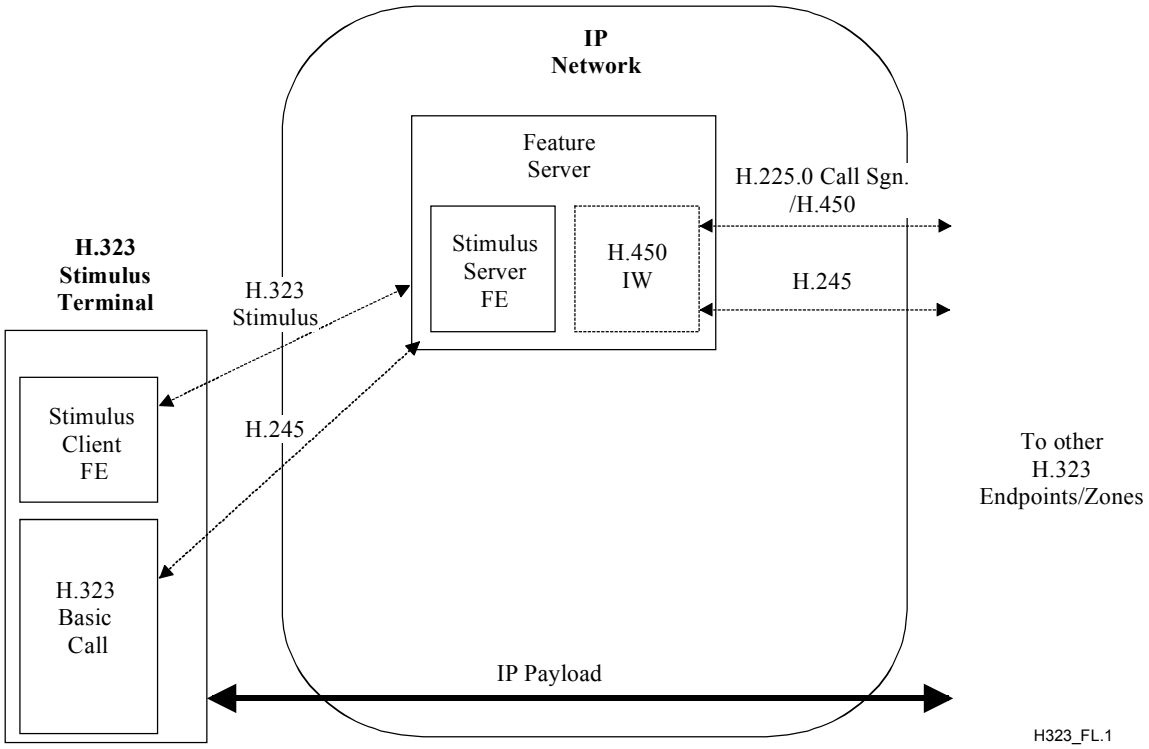


Figure L.1/H.323 – Example of Annex L in conjunction with direct signalling model

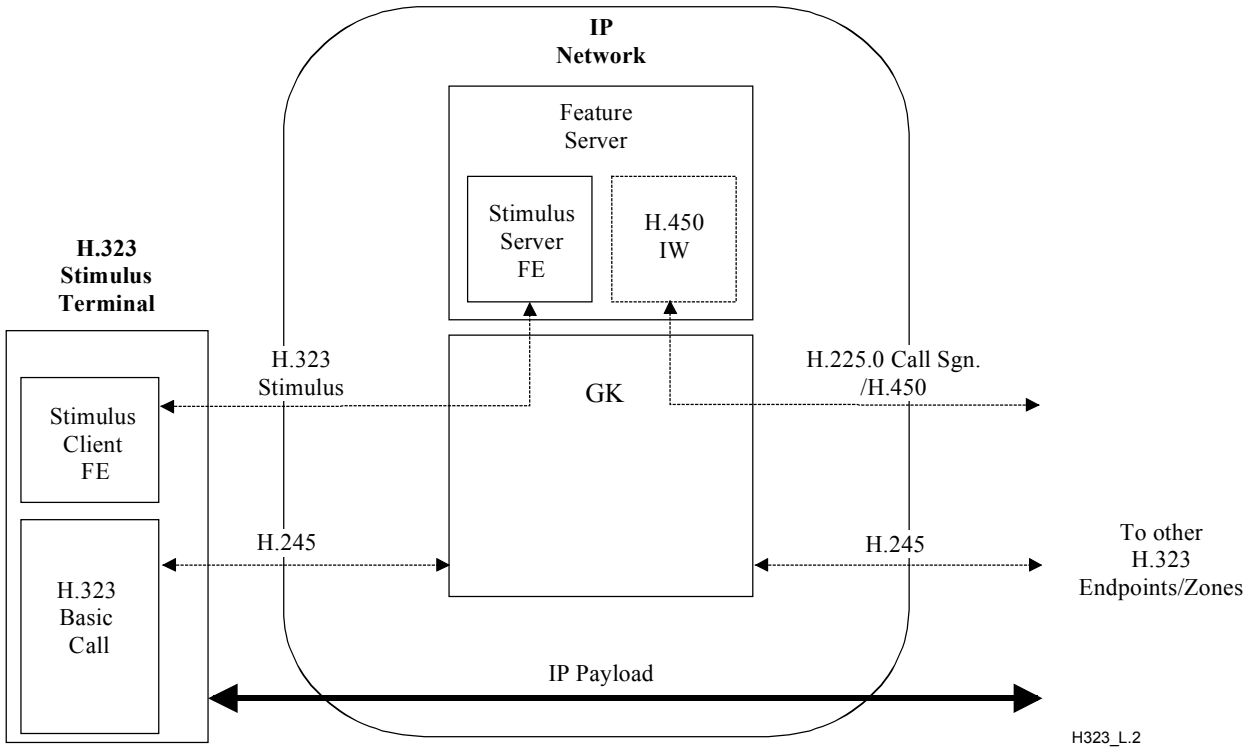


Figure L.2/H.323 – Example of Annex L in conjunction with GK-routed signalling model

L.1.1 Terminology

L.1.1.1 feature server: A Functional Entity that uses the method described in this annex to provide features to an Annex L Endpoint. A Feature Server may reside anywhere in the network. It may be colocated with a Gatekeeper, or reside on a gateway or other H.323 callable entity. A Feature Server may provide interworking between the stimulus protocol and H.450 services.

L.1.1.2 Annex L endpoint: An H.323 callable entity that can be controlled using the method described in this annex.

L.1.1.3 feature: A transaction that can affect the user interface and which may alter media streaming.

L.1.2 Relationship of H.323 stimulus to H.248

Since H.248 was developed for control of media gateways, it implies tight relationship between controller and the media gateway. Endpoints, such as telephones and residential gateways, can be included as controlled devices and treated as single line media gateways. However, these are tied to exactly one controller, which provides all connection control, features and services to the H.248 endpoints. A user can subscribe to features from one controller at a time.

This annex adopts the controller/endpoint model of H.248 for control of stimulus supplementary services, so that these procedures need to be defined only once. This annex explicitly excludes all parts of H.248 that are related to the control of media connections, which is done using standard H.245 or Fast Connect.

L.1.3 Relationship of H.323 stimulus to HTTP

Annex K allows third party control of an H.323 call based on a separate hypertext connection (using HTTP) for user interaction. There is no fixed set of capabilities for the user interface, as various types of text formats, images, and sounds will be utilized dynamically. The service provider (the HTTP server) is responsible for the mapping between HTTP events and call control actions (H.450 or other messages) for supplementary services, so the H.323 endpoint is unaware of the HTTP

application. The service provider may be associated with the local gatekeeper, or the remote endpoint, or remote gatekeeper within a call.

L.1.4 Relationship to H.450 supplementary services

Because a stimulus terminal does not perform H.450 supplementary services, the feature server or the gatekeeper is responsible for providing a proxy function for handling of the H.450 procedures over the network on behalf of the terminal.

In this case the feature server becomes an endpoint for all H.450 operations and implements all supplementary services and the state machines involved. The interaction with the user happens through the telephone user interface, which the gatekeeper is able to control via the H.323 stimulus signalling.

L.2 Introduction

The essential requirement for an H.323-based stimulus protocol is to provide a set of capabilities that allow supporting endpoints access to a potentially unlimited set of supplementary services. There are many benefits to such a protocol, such as allowing endpoints to remain relatively lightweight, and providing a degree of isolation from the effects of new feature introduction. These services themselves are typically controlled by a Gatekeeper, a proxy, or other network entity. This annex uses the term "Feature Server" to generically designate any network entity providing configuration or stimulus control of endpoints according to the protocol described.

The goals of the protocol described in this annex are:

- support for arbitrary (standard and non-standard) supplementary services;
- interoperability of these services between Feature Server and endpoint;
- backwards compatibility with endpoints using H.323 (version 2 or later).

This protocol achieves these goals by incorporating significant portions of the protocol described in ITU-T Rec. H.248. H.248 describes a purely stimulus model of endpoint control, whereas this annex must necessarily be a hybrid of both stimulus and H.323-based functional models. Annex L entities use H.248 PDUs in addition to standard H.323 messages to support this hybrid model.

This annex describes a framework that eases delivery of services into both H.323 and H.248-based systems, by allowing a high degree of commonality between an Annex L Feature Server and the components of an H.248 Media Gateway Controller (MGC) not related to media control. This framework enables the reuse of H.248-based packages in H.323-based systems, often with little or no modification. For example, suitably designed packages can allow a Feature Server to control various user interface elements of a compliant terminal, such as:

- write to a text display;
- provide hardware-independent indications to the endpoint, from which the endpoint may control its own indicators, such as message waiting or line lamps;
- receive user input such as digits, text, special keys (such as hookswitch and function keys);
- assign functions to soft keys and into an endpoint resident directory;
- request application of specific tones;
- specify tones dynamically.

Annex L terminals possess the above-listed control capabilities in common with H.248 terminals; the two types differ only in the means of managing media streams and their association with one or more calls or "contexts".

Use of the protocol described in this annex is suggested for, but is not restricted to Annex F Simple Endpoint Types.

L.3 Stimulus framework

L.3.1 Overview

Annex L terminals use the standard H.323 mechanisms for registration and signalling channel establishment. Normal H.225.0 call signalling is used for call establishment and termination. Media control may use the H.323 fast connect procedures (including repeated fastStart) or, optionally, H.245 signalling using procedures described in ITU-T Recs H.245, H.323 and its annexes. Use of these mechanisms may result in the creation of analogues to H.248 ephemeral terminations (which are not directly controllable using this annex).

Stimulus signalling capabilities of Annex L Endpoints will be specified in packages as in H.248. For example, an Annex L terminal might be described by a basic set package (for switchhook changes, etc.), a keypad package, an alerting package, a key package, and a display package. Additional packages might be included to permit modification of operational parameters and/or collect performance statistics.

As Annex L terminals are principally H.323 endpoints, H.323 procedures shall always apply and cannot be disabled by any H.248 signalling. For example, if an H.248 command results in the termination of a call, standard H.245 and H.225.0 signalling for call termination is still required.

L.3.2 Protocol signalling

The only form of signalling that all H.323 entities must support is H.225.0 Call Signalling. This is the most appropriate transport for the stimulus protocol as it allows a Feature Server to be co-located with a Gatekeeper or any other type of H.323 endpoint.

Annex L entities should support encapsulation of H.248 messages in the **StimulusControl** field, which is available in all H.225.0 call signalling messages. On every call on which it participates, an Annex L endpoint which supports H.248 encapsulation shall include a **StimulusControl** field in the first H.225.0 call signalling message that it sends to any other H.323 entity (the **StimulusControl** field may be empty).

When an endpoint registers with a Gatekeeper, the Gatekeeper may indicate an alias for the Feature Server in the **featureServerAlias** field of the RCF. When this alias is present, it should be used by an Annex L endpoint as the server destination for non-tunnelled H.248 signalling which is constrained to the functionality defined by this annex. Use of this alias address allows the Gatekeeper to associate or route the call to the Feature Server. Upon reception of a valid **featureServerAlias** in an RCF, a supporting endpoint shall immediately send an H.248 **ServiceChange** command containing the Root TerminationId to the indicated Feature Server address.

This allows two models of interaction between a Feature Server and an Annex L Endpoint:

- the Feature Server is present in the call signalling path for all H.225.0 call signalling messages for all calls originating and terminating on an Annex L Endpoint;
- a separate call signalling connection between the Annex L Endpoint and the Feature Server is established only when this feature is invoked.

L.3.3 Use of H.248

Annex L endpoints shall support the transaction level procedures of 7.2/H.248. Annex L signalling may include any of the commands defined in clause 7/H.248.

Because Annex L terminals do not use H.248 for media control, use of the following H.248 descriptors is not applicable to Annex L entities: ModemDescriptor, MuxDescriptor, StreamDescriptor, LocalControl Descriptor, Local Descriptor, Remote Descriptor, and TopologyDescriptor. These descriptors shall not be used for Annex L signalling and shall be ignored if received. Note that this annex cannot be used to explicitly address different media

streams; if an Annex L terminal supports multiple media streams (e.g., audio and video), the assignment of a termination to the call context (0xFFFFFFFD, see L.3.4, below) is implicitly assumed to refer to the stream carrying the appropriate medium.

The packages supported by the endpoint should be listed in the **supportedH248Packages** field of the RRQ when the endpoint registers with a Gatekeeper. If this field is present, but empty, a Feature Server can use an AuditCapabilities query to determine the supported packages.

L.3.4 H.225.0 encapsulation

All H.225.0 encapsulated Annex L related signalling uses a **StimulusControl** structure. Use of its fields is described in this clause. The use of Annex L by an endpoint is inferred from the presence of this structure in the first call signalling message sent by the endpoint to the feature server. If no H.248 message is encapsulated in this structure, then all of its optional contained fields may be omitted.

Encapsulated Annex L stimulus control shall be signalled using the **stimulusControl** field in the H323-UU-PDU element that is used for call signalling in H.323.

The H.248 message to be sent shall be encapsulated in the **h248Message** field in the **stimulusControl** sequence. The encapsulated message is a full MegacoMessage data type as defined in ITU-T Rec. H.248.

When an Annex L Feature Server becomes active within the context of an existing call, it may need to determine the state of that call, and/or the endpoint. This can be accomplished with the use of the H.248 AuditValue command.

The assignment of TerminationIds for physical terminations on the endpoint may be provisioned on the Feature Server and endpoint, predefined in a package, or obtained via AuditCapabilities.

H.248 signalling may be either binary (H.248 Annex A syntax, but using PER for encoding) or text (H.248 Annex B) based. The default is binary encoding. The presence of the **isText** field shall be used to indicate that H.248 Annex B encoding has been used for the H.248 descriptors in the **StimulusControl** structure. Annex L Endpoints may support only one form of encoding, and shall use the same form of encoding for all Annex L signalling to a Feature Server. Annex L Feature Servers should support both forms of encoding; communication from a Feature Server to an endpoint shall use only the form for which the endpoint has indicated support.

For H.225.0 encapsulated Annex L signalling, the special value "ANNEX-L", defined as 0xFFFFFFFD, shall be used as the ContextId for all call-related transactions. All commands shall apply to the current H.323 call (as represented by the **callIdentifier** of the H.225.0 call signalling message encapsulating the H.248 command). Commands not related to the call represented by the encapsulating H.225.0 message shall be associated with a ContextId value of NULL, as defined in ITU-T Rec. H.248.

Encapsulated Annex L transactions shall not use ContextId values other than NULL (as defined in ITU-T Rec. H.248) or ANNEX-L (as defined above).

Certain H.248 activities may not be associated with active H.323 calls. In this case, any existing call signalling channel between the endpoint and the Feature Server may be used, and the procedures of H.248 shall be used to associate the activity with the correct H.248 objects. For these activities, H.323 call independent signalling procedures may be used. For call independent signalling, the procedure of 7.2/H.450.1 shall be used.

For H.248 activities that can be associated with an active call with the desired Feature Server in the call signalling path, any appropriate H.225.0 call signalling message may be used to communicate between Feature Server and endpoint.

L.4 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

– ITU-T Recommendation H.248 (2000), *Gateway control protocol*.

– ITU-T Recommendation H.248 Annex G (2000), *User interface elements and actions packages*.

– ITU-T Recommendation H.450.1 (1998), *Generic functional protocol for the support of supplementary services in H.323*.

Annex M1

Tunnelling of signalling protocols (QSIG) in H.323

M1.1 Scope

The purpose of this annex is to give guidance on how the generic tunnelling mechanism described in 10.4 can be used to tunnel QSIG over H.323 networks. Other groups such as ISO/IEC are ultimately responsible for the QSIG procedures themselves. Information on QSIG (also known as PSS1) can be found in references [M1-1] and [M1-2] below.

M1.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[M1-1] ISO/IEC 11572:2000, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Circuit mode bearer services – Inter-exchange signalling procedures and protocol.*

[M1-2] ISO/IEC 11582:2002, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Generic functional protocol for the support of supplementary services – Inter-exchange signalling procedures and protocol*.

[M1-3] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

M1.3 Endpoint procedures

Endpoints supporting tunnelling of QSIG information shall use the procedures in 10.4, with the following OBJECT IDENTIFIER used as the TunnelledProtocol:

- **{iso (1) identified-organization (3) icd-ecma (0012) private-isdn-signalling-domain (9)}**

H.225.0 messages tunnel the entire QSIG message, unchanged, starting with the Protocol discriminator field, and ending with the other information elements. The binary content of the QSIG messages is encoded as an OCTET STRING in the **H323-UU-PDU.tunnelledSignallingMessage.messageContent**. Since the binary encoding of QSIG messages is what is tunnelled, the integrity of the QSIG messages is fully preserved, including any BER encoding of ASN.1 in Facility or Notification indicator information elements.

QSIG messages can, but need not, be tunnelled in the corresponding H.225.0 messages. For example, the QSIG SETUP message can be tunnelled in a H.225.0 SETUP message, and the QSIG RELEASE COMPLETE message can be tunnelled in an H.225.0 RELEASE COMPLETE message. For other messages, it is possible that there is no corresponding H.225.0 call signalling message (e.g., in the case of a QSIG DISCONNECT message) or the corresponding message is not available because it has already been sent. In those cases, the QSIG message may be tunnelled in an H.225.0 FACILITY message. A QSIG CALL PROCEEDING message should be tunnelled in an H.225.0 FACILITY message since the H.225.0 CALL PROCEEDING message does not have end-to-end significance. Also since the NOTIFY and PROGRESS messages are optional, they might not be delivered end-to-end and should be tunnelled in a FACILITY message unless tones or announcements are provided by the called side and no Progress indicator has been sent to the calling side so far. In this case a PROGRESS message (with Progress descriptor #1 or #8) should be used to tunnel a QSIG PROGRESS message. QSIG call clearing procedures may be supported by tunnelling the QSIG DISCONNECT and RELEASE messages in the H.225.0 FACILITY message. In the special case where a tunnelled QSIG RELEASE message is interpreted as a tunnelled QSIG RELEASE COMPLETE message (this happens when a QSIG RELEASE message is received when a RELEASE COMPLETE was expected), the H.323 call may be released by the side receiving the QSIG RELEASE message by sending an H.225.0 RELEASE COMPLETE with no tunnelled QSIG message.

A single QSIG call can be tunnelled in a single H.323 call. The relationship between QSIG call references and H.225.0 call references is outside the scope of this Recommendation.

Table M1.1 is indicative only and illustrates an example of the mapping between QSIG messages and H.225.0 messages.

Table M1.1/H.323 – Mapping between QSIG messages and H.225.0 messages

QSIG message	H.225.0 message
SETUP	SETUP
ALERTING	ALERTING
CONNECT	CONNECT
RELEASE COMPLETE	RELEASE COMPLETE
CALL PROCEEDING	FACILITY
FACILITY	
PROGRESS (Note)	
NOTIFY	
DISCONNECT	
RELEASE	
All other messages ...	
NOTE – If tones or announcements are provided by the called side this message should be tunnelled in a PROGRESS message rather than in FACILITY.	

M1.4 Tunnelling of QSIG connection oriented call independent signalling

For QSIG call independent signalling connections, no H.245 control channel and no media channels are required.

The call signalling procedures of H.225.0 may be used to establish a call independent signalling connection between the peer endpoints, as described in 10.4.

M1.5 Gatekeeper procedures

A gatekeeper participating in a call where QSIG tunnelling is used between the endpoints should pass along tunnelled QSIG messages unchanged unless it intends to terminate the tunnel. This may be the case when a gatekeeper is offering emulated QSIG services.

Annex M2

Tunnelling of signalling protocols (ISUP) in H.323

M2.1 Scope

The purpose of this annex is to give guidance on how the generic tunnelling mechanism described in 10.4 can be used to tunnel ISUP over H.323 networks. Other groups such as ITU-T are ultimately responsible for the ISUP procedures themselves. Information on ISUP can be found in references [M2-1] and [M2-2] below.

M2.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[M2-1] ITU-T Recommendation Q.763 (1999), *Signalling System No. 7 – ISDN user part formats and codes*.

[M2-2] ITU-T Recommendation Q.764 (1999), *Signalling System No. 7 – ISDN User Part signalling procedures*.

[M2-3] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

M2.3 Endpoint procedures

Endpoints supporting tunnelling of ISUP information shall use the procedures in 10.4. Endpoint shall identify the ISUP variant by either using the **tunnelledProtocolObjectID** or the **TunnelledProtocolAlternateIdentifier** structure. The **subIdentifier** may be used to identify the revision of the ISUP variant, e.g., "1988". See Table M2.1.

Table M2.1/H.323 – Examples of tunnelled protocols identified by tunnelledProtocolObjectID

Standard	tunnelledProtocolObjectID	subIdentifier
ITU-T Rec. Q.763 (1988)	{itu-t (0) recommendation (0) q (17) 763}	"1988"
ITU-T Rec. Q.763 (1992)	{itu-t (0) recommendation (0) q (17) 763}	"1992"

When using the **TunnelledProtocolAlternateIdentifier** structure the **protocolType** shall be set to "isup". The **protocolVariant** shall be a string identifying the ISUP specification used, e.g., a document number. See Table M2.2.

Table M2.2/H.323 – Examples of tunnelled protocols identified by TunnelledProtocolAlternateIdentifier

ISUP specification (Note)	protocolType	protocolVariant	subIdentifier
ANSI T1.113-1988	"isup"	"ANSI T1.113-1988"	"1988"
ETS 300 121	"isup"	"ETS 300 121"	"121"
ETS 300 356	"isup"	"ETS 300 356"	"356"
BELLCORE GR-317	"isup"	"BELLCORE GR-317"	"317"
JT-Q761-4 (1987-1992)	"isup"	"JT-Q761-4 (1987-1992)"	"87"
JT-Q761-4 (1993)	"isup"	"JT-Q761-4 (1993)"	"93"
NOTE – The ISUP specification may be a standard, a Recommendation or any other document specifying the ISUP protocol, e.g., an ISUP interconnection specification for a specific country.			

- **{ itu-t (0) recommendation (0) q (17) 763 }**

H.225.0 messages tunnel the entire ISUP message, unchanged, starting with the Message type code parameter, and ending with the other parameters. The binary content of the ISUP messages is encoded as an OCTET STRING in the **H323-UU-PDU.tunnelledSignallingMessage.messageContent**. Since the binary encoding of ISUP messages is what is tunnelled, the integrity of the ISUP messages is fully preserved.

For example, the ISUP IAM message can be tunnelled in a H.225.0 SETUP message, and the ISUP ANM message can be tunnelled in an H.225.0 CONNECT message. For other messages, it is possible that there is no corresponding H.225.0 message (e.g., in the case of an ISUP IDR message) or the corresponding message is not available because it has already been sent. In those cases, the ISUP message may be tunnelled in an H.225.0 FACILITY message.

A single ISUP call can be tunnelled in a single H.323 call.

Some information elements in the H.225.0 message may have been modified by the H.323 network and the gateway receiving the tunnelled ISUP message may need to override the corresponding ISUP parameters.

The **tunnellingRequired** flag shall be included in the Setup message when the ISUP required parameter in the IAM message indicates 'ISUP required'.

Table M2.3 is indicative only and illustrates an example of the mapping between ISUP messages and H.225.0 messages.

Table M2.3/H.323 – Mapping between ISUP messages and H.225.0 messages

ISUP message	H.225.0 message
IAM	SETUP
SAM	INFORMATION
CPG	CALL PROCEEDING, ALERTING, PROGRESS, NOTIFY or FACILITY
ACM	CALL PROCEEDING, ALERTING, PROGRESS, NOTIFY or FACILITY
ANM, CON	CONNECT
REL	RELEASE COMPLETE
All other messages	FACILITY

M2.4 Gatekeeper procedures

A gatekeeper participating in a call where ISUP tunnelling is used between the endpoints should pass along tunnelled ISUP messages unchanged unless it intends to terminate the ISUP tunnel. This may be the case when a gatekeeper is offering ISUP services.

A gatekeeper shall not select an endpoint that does not support ISUP when the **tunnellingRequired** flag is included the Setup message.

Annex M3

Tunnelling of DSS1 through H.323

M3.1 Scope

The purpose of this annex is to give guidance on how the generic tunnelling mechanism described in 10.4 can be used to tunnel DSS1 (Q.931) over H.323 networks. Other groups may adapt this procedure to accommodate national variants of DSS1.

M3.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[M3-1] ITU-T Recommendation Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control*.

[M3-2] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems*.

[M3-3] ITU-T Recommendation H.450.1 (1998), *Generic functional protocol for the support of supplementary services in H.323*.

M3.3 Endpoint procedures

Endpoints supporting tunnelling of DSS1 information shall use the procedures in 10.4, with the following OBJECT IDENTIFIER used as the **TunnelledProtocol.id. tunnelledProtocolObjectID** in a H.225.0 call signalling message or in the H.225.0 RAS message:

- **{itu-t (0) recommendation (0) q (17) 931}**

  Endpoints supporting tunnelling of DSS1 information and then acting as DSS1 User entity shall use the procedures in 10.4, with the following value used as the **TunnelledProtocol.subIdentifier**:

- **"User"**

  Endpoints supporting tunnelling of DSS1 information and then acting as DSS1 network entity shall use the procedures in 10.4, with the following following value used as the **TunnelledProtocol.subIdentifier**:

- **"Network"**

  When sending a H.225.0 RAS message requesting a specific tunnelled protocol (see 10.4.2) in the **desiredTunnelledProtocol** field an endpoint has to include the OBJECT IDENTIFIER and subidentifier of the protocol it expects from the other side to ensure proper gatekeeper functionality.

  DSS1 is an asymmetrical protocol and can only be used between one user and one network entity. By using different OBJECT IDENTIFIERS for user and network entities, the H.323 endpoints can ensure that no DSS1 tunnelling takes place between two user or two network entities.

  H.225.0 messages tunnel the entire message, unchanged, starting with the Protocol discriminator field, and ending with the other information elements. The binary content of the DSS1 messages is encoded as an OCTET STRING in:

- **H323-UU-PDU.tunnelledSignallingMessage.messageContent**

  Since the binary encoding of DSS1 messages is what is tunnelled, the integrity of the DSS1 messages is fully preserved, including any BER encoding of ASN.1 in Facility or Notification indicator information elements.

  DSS1 messages can be tunnelled in the corresponding H.225.0 message or in H.225.0 FACILITY messages. For example, the DSS1 SETUP message may be tunnelled in a H.225.0 SETUP message, and the DSS1 RELEASE COMPLETE message may be tunnelled in an H.225.0 RELEASE COMPLETE message. For other messages, it is possible that the corresponding H.225.0 message may not be supported (e.g., a DSS1 CONNECT ACK message), not available because it has already been sent or not transparently transported end-to-end. In those cases, the DSS1 message shall be tunnelled in an H.225.0 FACILITY message. In particular, the H.225.0 SETUP ACKNOWLEDGE or CALL PROCEEDING messages shall not be used for tunnelling of a DSS1 message, because it may not reach the originating H.225.0 endpoint, if an intermediate Gatekeeper has already sent such a message. Instead, for tunnelling of a DSS1 SETUP ACKNOWLEDGE or CALL PROCEEDING message, first a H.225.0 SETUP ACKNOWLEDGE or CALL PROCEEDING message without a tunnelled DSS1 message shall be sent, followed by a H.225.0 FACILITY message tunnelling the DSS1 SETUP ACKNOWLEDGE or DSS1 CALL PROCEEDING message. Also, DSS1 STATUS and STATUS ENQUIRY messages shall be tunnelled in a H.225.0 FACILITY message, to ensure, that the DSS1 messages reach the H.225.0 endpoint.

  DSS1 call clearing procedures may be supported by tunnelling the DSS1 DISCONNECT and RELEASE messages in the H.225.0 FACILITY message.

A single DSS1 call may be tunnelled in a single H.323 call. The DSS1 call reference is selected by the ingress endpoint and shall be the same in all tunnelled DSS1 messages for an H.323 call. However, the DSS1 call reference value in a TDM network is unique on a peer DSS1 entity basis. In an H.323 system, there is no peer DSS1 entity basis since any H.323 call may terminate on any endpoint. To ensure uniqueness, the H.323 call reference value should be used for identifying the H.323 call only.

The DSS1 tunnelling procedures shall not be used in conjunction with the H.450.1 procedures in the same call.

Table M3.1 illustrates the relationship between tunnelled DSS1 messages and enveloping H.225.0 messages.

Table M3.1/H.323 – Relationship between tunnelled DSS1 messages and enveloping H.225.0 messages

Q.931/Q.932 message	H.225.0 message	Remark
Call establishment messages		
ALERTING	ALERTING	
CALL PROCEEDING	FACILITY	
CONNECT	CONNECT	
CONNECT ACKNOWLEDGE	FACILITY	
INFORMATION	FACILITY	Support of H.225.0 INFORMATION message is optional
PROGRESS	FACILITY	Support of H.225.0 PROGRESS message is optional
SETUP	SETUP	
SETUP ACKNOWLEDGE	FACILITY	
Call clearing messages		
DISCONNECT	FACILITY	
RELEASE	FACILITY	
RELEASE COMPLETE	RELEASE COMPLETE	
Call Information messages		
RESUME	For further study	
RESUME ACKNOWLEDGE	For further study	
RESUME REJECT	For further study	
SUSPEND	For further study	
SUSPEND ACKNOWLEDGE	For further study	
SUSPEND REJECT	For further study	
USER INFORMATION	FACILITY	
Miscellaneous messages		
CONGESTION CONTROL	FACILITY	
NOTIFY	FACILITY	Support of H.225.0 NOTIFY message is optional
STATUS	FACILITY	
STATUS ENQUIRY	FACILITY	

Table M3.1/H.323 – Relationship between tunnelled DSS1 messages and enveloping H.225.0 messages

Q.931/Q.932 message	H.225.0 message	Remark
FACILITY	FACILITY	
HOLD	FACILITY	
HOLD ACKNOWLEDGE	FACILITY	
HOLD REJECT	FACILITY	
RETRIEVE	FACILITY	
RETRIEVE ACKNOWLEDGE	FACILITY	
RETRIEVE REJECT	FACILITY	
NOTE – DSS1 messages with global call reference e.g., RESTART, RESTART ACK and STATUS may be treated by the endpoints and therefore they may not be tunnelled.		

M3.4 Tunnelling of bearer-independent DSS1 signalling

For tunnelling of the bearer-independent transport mechanisms of DSS1 as described in 6.3.2/Q.932, no H.245 control channel and no media channels are required.

The call signalling procedures of H.225.0 may be used to establish a call independent signalling connection between the peer endpoints, as described in 10.4. For details on this call independent signalling connection, see also 6.2/H.450.1.

M3.4.1 DSS1 connectionless transport

The DSS1 connectionless transport mechanism as described in 6.3.2.2/Q.932 is based on FACILITY messages using the dummy call reference value.

Each such DSS1 FACILITY message shall be transported in a separate H.225.0 connection, which shall be cleared immediately after reaching the terminating side.

In particular, a DSS1 FACILITY message shall be transported in a H.225.0 SETUP message, as described in 10.4 and in 6.2/H.450.1. The terminating side (but no intermediate Gatekeeper) shall clear this connection immediately with a H.225.0 RELEASE COMPLETE message. Additionally, the entity sending the H.225.0 SETUP message shall clear the call after receiving expiry of an appropriately chosen timer which has been started after sending the H.225.0 SETUP message.

M3.4.2 DSS1 bearer-independent connection-oriented transport

The DSS1 bearer-independent connection-oriented transport mechanism as described in 6.3.2.1/Q.932 is based on connections initiated with REGISTER messages.

Here, the following message mapping shall apply:

Q.931/Q.932 message	H.225.0 message	Remark
REGISTER	SETUP	The H.225.0 SETUP message shall be used to set up a call-independent signalling connection as described in 6.2/H.450.1. The H.225.0 SETUP message shall be acknowledged with a H.225.0 CONNECT message in order to prevent call clearing after T303 expiry.
FACILITY	FACILITY	
RELEASE COMPLETE	RELEASE COMPLETE	

M3.5 Gatekeeper procedures

A gatekeeper participating in a call where DSS1 tunnelling is used between the endpoints should pass along tunnelled DSS1 messages unchanged unless it intends to participate in the DSS1 procedures and terminate the DSS1 protocol. This may be the case when a gatekeeper is offering DSS1 services.

Annex O

Usage of URLs and DNS

O.1 Scope

This Recommendation defines a means for building multimedia communication services over an arbitrary packet-based network, including the Internet. It is useful to take advantage of such services as the Domain Name System (DNS) [O-1] and ENUM [O-9] in order to help facilitate the completion of multimedia calls, especially when using H.323 over the Internet. This Recommendation defines the procedures for using DNS to locate gatekeepers and endpoints and for resolving H.323 URL aliases. This annex also defines parameters for use with the H.323 URL.

O.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[O-1] IETF RFC 1034 (1987), *Domain names – concepts and facilities*.

[O-2] IETF RFC 2396 (1998), *Uniform Resource Identifiers (URI): Generic Syntax*.

[O-3] IETF RFC 2782 (2000), *A DNS RR for specifying the location of services (DNS SRV)*.

O.3 Informative references

Note that this is an informative material and it is not required to implement this annex.

[O-4] ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan*.

[O-5] IETF RFC 768 (1980), *User datagram protocol*.

[O-6] IETF RFC 793 (1981), *Transmission control protocol*.

[O-7] IETF RFC 1006 (1987), *ISO transport services on top of the TCP: Version 3*.

[O-8] IETF RFC 2806 (2000), *URLs for Telephone Calls*.

[O-9] IETF RFC 2916 (2000), *E.164 number and DNS*.

[O-10] IETF RFC 2960 (2000), *Stream Control Transmission Protocol*.

O.4 H.323 URL

The H.323 Uniform Resource Locator (URL) describes a location for an H.323 entity or service reachable using standard H.323 procedures. The H.323 URL can include optional parameters that specify services and transport protocols that facilitate H.323 communications. The URL is suitable for usage within web pages, as input provided by the user, as output of an ENUM procedure, etc.

The H.323 URL has the general form of *user@hostport* where either both of the parts (i.e., *user* and *host*) or only one of the parts (i.e., *user* alone or *@host* alone*)* is present. The *user* part corresponds to an H.323 user or service name. The *host* part is a legal numeric IP address or a fully qualified domain name, thus providing means for address resolution using the DNS infrastructure.

Refer to 7.1.4 for the specific syntax of the H.323 URL.

This annex defines the H.323 URL parameters and the procedures for the H.323 URL usage.

O.5 Encoding of H.323 URL in H.323 messages

In general, procedures defined by this annex apply to an H.323 URL that is encoded with its scheme name. Processing of an URL/URI without an encoded scheme name is left for future study unless specified explicitly by this Recommendation.

An endpoint shall encode the H.323 URL with its scheme name in the **url-ID** field of **AliasAddress**.

During an address resolution procedure, a gatekeeper shall try to retrieve an H.323 URL from the **url-ID** field of **AliasAddress**. If unsuccessful, the gatekeeper should try to retrieve an H.323 URL from the **h323-ID** of **AliasAddress**. The latter is in order to support URL addressing even if an URL interface is not exposed to a user of earlier endpoint implementations. As a result the user will be able to convey the destination URL by inserting it with its scheme name manually as if it was a free format **url-ID**.

O.6 Non-H.323 URLs and URIs within the context of H.323

Non-H323 standard URL and URI schemes (such as *mailto*, *tel*, and *sip*) may be embedded into H.323 messages.

Non-H.323 URIs shall be inserted into H.323 messages in their full form (including the scheme name) in **url-ID** field of **AliasAddress** type.

An H.323 entity (such as a Gatekeeper) shall process any URI (embedded in an H.323 message) according to its syntax and semantics as pointed to by its scheme name.

O.7 H.323 URL parameters

The following table summarizes optional standard *url-parameters* of an H.323 URL. The valid parameter combinations are implied from the main text of Recommendation H.323.

Parameter	Brief description
user	Indicates that the *user* part of the H.323 URL contains a phone number.
service	Specifies the recommended type of service (i.e., one of the H.323 protocols) to be invoked first to reach the particular entity.
transport	Indicates the transport protocol to be used for the service above.

O.7.1 ABNF syntax

This annex specifies the following standard values for the **url-parameter** that is defined in 7.1.4:

```
user-parameter      =   "user=phone"
service-parameter   =   "service="("ls" | "rs" | "cs" | "be")
transport-parameter =   "transport="("udp" | "tcp" | "h323mux" | "sctp")
```

NOTE – These parameters may take on additional values in subsequent revisions of this Recommendation.

O.7.2 User parameter

Currently a single standard value for the *user* parameter is defined: *phone*.

Using *user=phone* allows to explicitly express that the user part of the H.323 URL carries a telephone number.

When encoding the *tel* URL scheme [O-8] within the H.323 URL, its scheme name (i.e., "tel:") shall be omitted and any of its used attributes (starting with ";") shall be placed in the *user* part of the H.323 URL. Note that each character occurring in the *tel* URL but is not allowed in the user part of the H.323 URL shall be escaped.

O.7.3 Service parameter

The *service-parameter* can have one of four values: *ls*, *rs*, *cs*, or *be* specifying RAS LRQ, RAS RRQ, call signalling messages of H.225.0, or the inter/intra-domain protocol defined in Annex G/H.225.0 correspondingly.

The value of *service-parameter* is that of the preferred service. In the process of connection establishment, the originating side may try using other services than the one specified by the *service-parameter*.

If *service-parameter* is absent, the H.323 entity may attempt each of the services in user-defined order. For specific guidelines, refer to O.9.

O.7.4 Transport Parameter

Signalling protocols defined in this Recommendation may be carried over different transports. Values *udp, tcp, h323mux,* and *sctp* specify UDP [O-5], TCP [O-6], Annex E/H.225.0, and SCTP [O-10] respectively. For each H.323 protocol, there are default values for both the transport protocol and the listening port (i.e., the well-known TSAP identifier) specified by ITU-T Recs H.323, H.225.0 and their corresponding Annexes. Default values may be specified by *transport-parameter* and/or *port* of an H.323 URL. Values, different from the defaults, shall be specified by the *transport-parameter* and/or *port* of an H.323 URL.

Note that inclusion of *port* parameter (including the default value) has special meaning. It is an indication to the resolving entity that the *host* points to a specific H.323 entity rather than a remote DNS domain containing H.323 SRV RRs. For details see O.9.

The value of *transport-parameter* is that of the preferred transport. In the process of connection establishment, the originating side may try using other transport protocols than the specified by the *transport-parameter*.

O.8 Usage of the H.323 URL

Currently there are two primary reasons for using an H.323 URL: to locate a callable H.323 entity and to locate a Gatekeeper with which an endpoint may register.

Additionally, ENUM [O-9] defines a system for storage of and access to mappings between E.164 numbers [O-4] and the services associated with them. The ENUM system is implemented using the Domain Name System (DNS) where the available services are represented by standard URIs [O-2].

Other uses for the H.323 URL are for further study.

O.8.1 Locating H.323 destination

When an H.323 URL is embedded in a web page or other hyperlink it means that a particular user or a service can be reached using the H.323 protocol.

Any H.323 entity may resolve the H.323 URL by utilizing DNS, including endpoints, gatekeepers, or border elements as a part of a call setup procedure defined in 8.1.

If an originating endpoint chooses to resolve the destination URL, it shall encode both the URL and the successfully resolved destination IP address (according to O.9) in the **destinationInfo** of ARQ RAS message or in the **destinationAddress** of Setup and continue the normal H.323 call setup. Otherwise, i.e., if the originating endpoint chooses not to resolve the destination URL or the DNS lookup fails, the endpoint shall encode the H.323 URL according to O.5 in the **destinationInfo** of ARQ RAS message or in the **destinationAddress** of Setup message and continue the normal H.323 call setup.

If the destination URL contains a *user* part only, a resolving H.323 entity shall logically act as if the *hostport* contained its own domain name.

Only a resolving entity that belongs to the URL domain (as specified by the *hostport*) shall interpret and process the *user* part of the H.323 URL based on its local policy. Such local policy may be based on (but is not limited to) procedures defined by H.225.0 RAS, Annex G/H.225.0, LDAP or local configuration.

If the *hostport* of the H.323 URL, is different from the DNS domain of the resolving entity it shall first perform the DNS procedure as specified in O.9. Only if the DNS procedure fails, the resolving entity may retreat to performing a different address resolution procedure based on its local policy.

O.8.2 Locating a Gatekeeper

This Recommendation defines a means of discovering a Gatekeeper via the GRQ RAS message. Generally, this entails sending out GRQ messages without any prior configuration required.

However, static provisioning of a Gatekeeper location within an endpoint is very common. It allows for better management and flexible security schemes to be implemented in the network.

Provisioning of a Gatekeeper location in terms of an H.323 URL and supporting DNS procedures for Gatekeeper discovery by the endpoints provide additional benefits. If SRV Resource Records are implemented, Gatekeeper redundancy and load-balancing schemes can be deployed transparently to the endpoints.

If an endpoint is provisioned for its Gatekeeper location with an H.323 URL in a form of "h323:*@hostport*" with no parameters it should use the *hostport* value for its Gatekeeper discovery. If an endpoint is provisioned for its Gatekeeper location with just a valid DNS domain name it is assumed that this DNS domain name is the value of the *hostport* of the H.323 URL above.

If an endpoint isn't provisioned with the H.323 URL for its Gatekeeper location but is provisioned with its own H.323 URL, it may use the *hostport* value of the endpoint's URL for the Gatekeeper discovery.

In order to discover its Gatekeeper the endpoint should use the provisioned *hostport* value and the implied **service** equals h323*rs* and **proto** equals *udp* as the inputs to the address resolution procedure defined in O.9.

If the procedure fails, the endpoint shall follow the normal Gatekeeper discovery procedures outlined in the main text of this Recommendation.

O.9 Resolving an H.323 URL to IP Address using DNS

The *host* part of the H.323 URL can specify one of the following:

– The IP numeric address of an H.323 entity.

– The DNS name of a host which is an H.323 entity.

– The remote DNS domain containing H.323 SRV RRs.

This clause defines the address resolution procedure covering these three cases.

When the *host* contains an IP numeric address, nothing needs to be resolved using DNS. The H.323 messages shall be sent directly to the specified IP address.

When the *hostport* part of the URL is present and contains a port number, it means that the *host* points to a specific H.323 entity (rather than specifying a DNS domain containing H.323 SRV RRs). This *port* value shall be assumed to be the port to which H.323 messages to be directed. Note that if the default port is to be used, the default port number shall be inserted in order to represent this case. The resolving entity shall attempt to retrieve Address Resource Record(s) ("A" RR or "AAAA" RR) for the domain name specified by the *host*. If more than a single record is retrieved, the resolving entity should select a single record based on its local policy (see also O.10.1). The H.323 messages shall be sent to the retrieved (and potentially selected) IP address and the port specified by the URL.

When the *hostport* part of the URL is present but doesn't contain a port number, it hints that the *host* most probably specifies a DNS domain containing H.323 SRV RRs. The resolving entity should attempt to locate the entity by performing a sequential SRV records retrieval within a subset of the possible H.323 services (i.e., *h323ls*, *h323rs*, *h323be*, and *h323cs*) and their corresponding possible transport protocols (i.e., *udp*, *tcp*, and *h323mux*) according to the procedure specified in O.10.4. The subset shall match the resolving entity capabilities and the purpose of the procedure (i.e., locating a Gatekeeper vs. locating a foreign border element vs. locating a destination). If the H.323 URL *service-parameter* is present or the SRV *service* (e.g., *h323rs*) is specified, the order of the SRV lookups should start based on its value. In the case that the *service-parameter* is not specified, the resolving entity may search for any or all SRV record types in any order.

For each successful retrieval, an additional DNS lookup needs to be performed to retrieve the Address Resource Records. If successful, the H.323 messages shall be sent to the retrieved and selected IP address and a default port number (corresponding to the transport protocol).

If the SRV RR retrieval procedure is not implemented or fails, the resolving entity may attempt to retrieve the Address Resource Record(s) for the domain name specified by the *hostport* even if the *port* hasn't been specified. If more than a single record is retrieved, the resolving entity should select a single record based on its local policy (see also O.10.1). If successful, the H.323 messages shall be sent to the retrieved (and potentially selected) IP address and a corresponding default port number.

O.10 Using DNS SRV Resource Records

O.10.1 Applicability

By using DNS SRV RRs (RFC 2782 [O-3]) it is possible to publish an address (i.e., an URI) for a specific service (*Service*) that can be reached over a specific protocol (*Proto*). "The SRV RR allows administrators to use several servers for a single [DNS] domain, to move services from host to host with little fuss and to designate some hosts as primary servers for a service and others as backups."

In the following clauses, this annex defines symbolic names for H.323 services and H.323 transport protocols to be registered with IANA and required for using the DNS SRV RRs. This annex also defines normative procedures for the SRV RRs usage in H.323 systems.

O.10.2 IANA registration

This specification defines the following symbolic names to be used in the *Service* field of the SRV record according to RFC 2782 [O-3].

Service	Name	Meaning
h323ls	Location Service	H.323 entity supporting H.225.0 LRQ procedure
h323rs	Registration Service	H.323 entity supporting H.225.0 RRQ procedure (i.e., a Gatekeeper that accepts registration of endpoints)
h323cs	Call Signalling	H.323 entity that performs H.225.0 call signalling
h323be	Border Element	H.323 supporting communication as defined in Annex G/H.225.0

This specification defines the following symbolic names to be used in the *Proto* field of the SRV record according to RFC 2782 [O-3].

Symbolic name	Meaning
udp	UDP as defined by RFC 768 "User datagram protocol" [O-5]
Tcp	TPKT [O-7] over TCP [O-6] in accordance with Appendix IV/H.225.0
sctp	SCTP as defined by RFC 2960 [O-10]
h323mux	As defined by Annex E, "Framework and wire-protocol for multiplexed call signalling transport."

O.10.3 SRV RR population

As defined in RFC 2782 [O-3], the DNS type code of SRV RR is 33 and its format is as follows:

_Service._Proto.Name TTL Class SRV Priority Weight Port Target

All the fields shall be populated in accordance with RFC 2782.

Service and *Proto* shall have one of the symbolic names defined above. *Port* shall have a value of a listening port on the H.323 host, which is defined by a *Target*.

If different forms of H.323 access (i.e., combinations of *Service* and *Proto*) are available for the DNS domain, all of them shall be published each using a separate SRV record.

Priority and *Weight* fields shall be used to express services local preferences' policy.

O.10.4 SRV RR retrieval and processing

This procedure does not define processing priority among H.323 *Service*s or among H.323 *Proto*s.

This procedure takes as an input a specific H.323 *Service* value and a specific *Proto* value only. Lookup in a form of _service.* is not allowed.

If no SRV records are retrieved, the procedure fails.

When the retrieved SRV records are locally processed, the selection algorithm based on *Priority* and described in RFC 2782 shall be followed. The selection algorithm based on *Weight* and described in RFC 2782 should be followed. The Priority and Weight values shall not be compared across different H.323 *Services* or different H.323 *Protos*.

The output of the process is an ordered list of SRV RRs (with or without corresponding address RR potentially provided in the Additional Data section of SRV RR).

O.10.5 Example 1

This example shows a fragment of a DNS zone or DNS domain file for **example.com**. All H.323 servers are listening on well-known TSAPs. There are two Gatekeepers installed in the domain. The **local-gatekeeper** provides registration services and can be "discovered" by its local endpoints. From the outside, the H.323 services can be reached through the external-gatekeeper by looking up the call signalling services of the domain. In addition, the external-gatekeeper would resolve its endpoint addresses by answering to the LRQ requests from the outside of its domain.

The functional separation between the two gatekeepers could be logical only and be useful in "NATted" environments where the two gatekeepers would represent local and external IP addressing.

```
$ORIGIN example.com.
_h323rs._udp            SRV 0 1 2517 local-gatekeeper.example.com.
_h323ls._udp            SRV 0 1 2517 external-gatekeeper.example.com.
_h323cs._tcp            SRV 0 1 1720 external-gatekeeper.example.com.
local-gatekeeper        A   172.30.79.11
external-gatekeeper     A   172.30.79.12
; NO H.323 access over H.323 Annex E is supported
*._h323mux              SRV  0 0 0 .
; NO other services are supported (including H.323 Border Element)
*._tcp                  SRV  0 0 0 .
*._udp                  SRV  0 0 0 .
```

O.10.6 Example 2

This example shows a fragment of a DNS zone or DNS domain file for **example.com**. All H.323 servers are listening on well-known TSAPs. The H.323 service is provided through both a Border Element and Gatekeepers. There is no priority defined or assumed between the Border Element and the Gatekeepers. It is a matter of application. For example, voice-only high quality service is provided through the Border Element while H.323 videoconferencing is provided through the Gatekeepers.

An H.323 voice phone residing in a domain would have the following URL: **h323:my-alias@example.com;service=be**. In this case a lookup for **_h323be._udp** will be performed first and succeed. Note that a lookup for **_h323cs._tcp** is allowed as well.

A videoconferencing service, provided by an H.323 MCU located in a zone of either a **main-gatekeeper** or **secondary-gatekeeper** would be published as **h323:conference-alias@example.com;service=cs**. This is due to the fact that the SRV records matching **_h323cs._tcp** will be retrieved, based on the *service-parameter*. Further, using the **Weight** field, actual access to the **main-gatekeeper** is three quarters that of the **secondary-gatekeeper** if either of the Gatekeepers is up and running.

```
$ORIGIN example.com.
_h323be._udp            SRV 0 1 2099 border-element.example.com.
_h323cs._tcp            SRV 0 1 1720 secondary-gatekeeper.example.com.
_h323cs._tcp            SRV 0 3 1720 main-gatekeeper.example.com.
border-element          A   172.30.79.10
main-gatekeeper         A   172.30.79.11
secondary-gatekeeper    A   172.30.79.12
; NO H.323 access over H.323 Annex E is supported
*._h323mux              SRV  0 0 0 .
; NO other services are supported (including H.323 Location Service)
*._tcp                  SRV  0 0 0 .
*._udp                  SRV  0 0 0 .
```

Annex P

Transfer of modem signals over H.323

P.1 Scope

The purpose of this annex is to describe the procedures for transferring modem signalling over an H.323-based network. The signalling procedures describe the use of H.245 (including Fast Connect and Extended Fast Connect), State Signalling Events (SSEs) to signal endpoint capabilities, to open and close logical channels, and to signal state changes. H.323 entities that support the carriage of modem signals of IP networks shall provide that functionality in accordance with this annex.

P.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[P-1] ITU-T Recommendation V.150.1 (2003), *Modem-over IP networks: Procedures for the end-to-end connection of V-series DCEs*.

[P-2] ITU-T Recommendation H.460.6 (2002), *Extended Fast Connect feature*.

[P-3] IETF RFC 2198 (1997), *RTP Payload for Redundant Audio Data*.

P.3 Definitions

This annex defines the following terms:

P.3.1 modem over IP: The transport of modem signals over an IP network as described in ITU-T Rec. V.150.1.

P.3.2 modem relay: The transportation of modem data across a packet network using modem termination at the network access points.

P.3.3 state signalling event: RTP-encoded event messages that coordinate switching between different media states as defined in Annex C/V.150.1.

P.3.4 voice band data: The transport of modem signals over an audio channel of a packet network with the encoding appropriate for modem signals.

P.4 Abbreviations

This annex uses the following abbreviations:

FEC Forward Error Correction

MoIP Modem over IP

MPS Multiple Payload Stream

OLC Open Logical Channel

RTP Real Time Protocol

SPRT Simple Packet Relay Transport

SSE State Signalling Event

VBD Voice Band Data

P.5 Introduction

H.323 systems have been widely deployed throughout the world for the carriage of audio, video and data traffic over packet-based networks, including IP networks. One of the applications of H.323 has been for transiting audio calls between two disjoint circuit-switched networks or two points on the same circuit-switched network. In such an application, the call is originated in a circuit-switched network and delivered to an H.323 Gateway. This Gateway then establishes communication with a remote Gateway that, in turn, delivers the call to a circuit-switched network.

In such applications, it is desirable to also allow calls between Gateways to be data calls, rather than audio or video calls only. Annex D introduced the signalling procedures necessary to facilitate the transport of facsimile data over an IP-based network between Gateways and other devices. The focus of this annex is to specify the procedures for transporting modem data over an IP-based network between two Gateways.

Figure P.1 graphically depicts H.323 Gateways that carry modem signals between modems over an IP network.

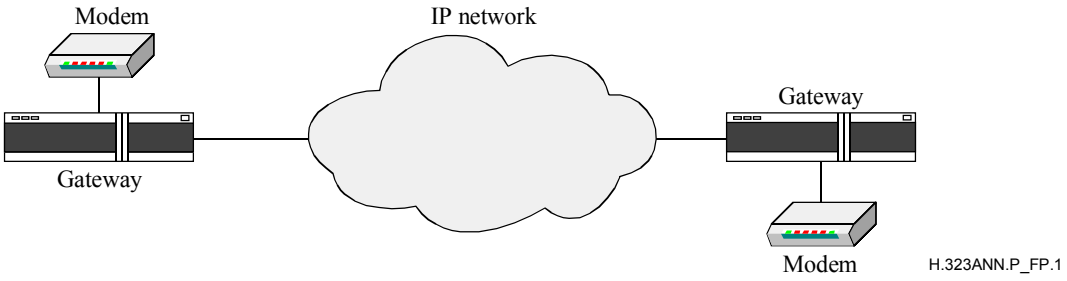


Figure P.1/H.323 – Typical modem over IP application

ITU-T Rec. V.150.1 defines the general procedures for carrying modem signals over IP-based networks between two Gateways and should be read in conjunction with this annex. Whereas ITU-T Rec. V.150.1 does not define the carriage of modem signals within the context of any particular call control protocol, this annex defines the procedures that are necessary and particular to this Recommendation.

Unless explicitly stated otherwise, references to H.323 endpoints throughout the remainder of this annex are to endpoints that are capable of carrying modem signals over an IP network.

P.6 Capability advertisement

As usual, endpoints advertise their capabilities using the **terminalCapabilitySet** message in H.245. The capabilities that are of particular importance and required for the application of modem over IP are the MoIP and SSE data application capabilities (defined in Annex F/V.150.1), RTP audio telephony events (see B.2.2.13/H.245), and the **vbd** audio capability. The **fecCapability** and/or **redundancyEncodingCapability** capabilities may be supported in order to improve the reliability of Voice Band Data (VBD) channel.

Endpoints shall also advertise support for the **multiplePayloadStream** (MPS) in the capability set transmitted to the other endpoint.

The MoIP and SSE capability definitions are in Annex F/V.150.1.

In accordance with ITU-T Rec. V.150.1, the list of codecs supported as VBD codecs shall include G.711 μ-law and A-law. Further, H.323 endpoints shall support G.711 for VBD at 64 kbit/s and, optionally, at 56 kbit/s.

P.7 Call establishment

Because of the time-critical nature of modem signalling, the calling endpoint should use the Fast Connect procedure to offer one or more channels suitable for MoIP operation. The calling endpoint should also include its terminal capabilities in the **parallelH245Control** field in order to facilitate the rapid negotiation of MoIP-related channels.

Likewise, the called endpoint should return a Fast Connect reply as quickly as possible. This reply may be an acceptance or a refusal of the offered channels. Additionally, if the **parallelH245Control** field is present in the Setup message, the called endpoint should acknowledge the receipt of that information as specified in 8.2.4.

In the case that media cannot be negotiated through Fast Connect for any reason, the endpoints shall begin logical channel signalling via the H.245 Control Channel as quickly as possible. Again, the implementor is cautioned about the time critical nature of MoIP and is encouraged to initiate this signalling well before the transmission of the Connect message.

P.8 Logical channel signalling

There are five types of streams of particular importance to an endpoint that supports MoIP. Those streams are: an audio stream, a VBD stream, RTP audio telephony events, State Signalling Events (SSEs), and an SPRT stream. An endpoint shall logically group streams necessary for MoIP together via an MPS channel. One exception to this requirement is that the SPRT stream may be signalled as a separate channel and associated to the audio/VBD channel using the **associatedSessionID** field.

Within the context of a MoIP session, the MPS channel that contains the audio and/or VBD streams and other streams for MoIP should be considered the primary audio session. As such, the H.245 **sessionID** should be set to 1. However, endpoints are at liberty to use dynamic session ID values, as prescribed by ITU-T Rec. H.245.

While there are no strict limitations on the number of streams that may be contained within any MPS channel, the MPS channel used for MoIP shall contain no more than one audio stream, no more than one VBD stream, no more than one SSE stream, and no more than one SPRT stream. If the SPRT stream is opened as a separate channel, the MPS channel shall not also include an SPRT stream. In addition, there may be one payload type for normal audio, one for the VBD stream, one for the SSE stream, and one for the SPRT stream. It is possible that more than four payload types may be utilized for those four streams. For example, if the VBD stream is protected with Forward Error Correction (FEC), and if those FEC packets are contained within a Redundancy Encoding packet, there may be not just one payload type value for the VBD stream, but three: one used in the RTP header to signify that the packet contains a redundantly encoded payload, one for the primary payload (the VBD data), and one for the FEC data carried as the secondary encoding.

To optionally protect the VBD stream, an endpoint may utilize forward error correction and/or redundancy encoding. A stream that utilizes forward error correction shall be signalled via the **fec** field of the **DataType** structure within the **MultiplePayloadStreamElement** structure. A stream that utilizes redundancy encoding shall be signalled via the **redundancyEncoding** field in the **DataType** structure within the **MultiplePayloadStreamElement** structure.

To illustrate the usage of the MPS for MoIP, consider an OLC that has a G.729 audio stream, a G.711 A-law VBD stream that is protected with redundancy encoding, an SSE stream, and an SPRT stream. The **OpenLogicalChannel** would essentially have a composition similar to that shown in this abbreviated example:

```
{
     forwardLogicalChannelNumber 1,
     forwardLogicalChannelParameters {
          dataType : multiplePayloadStream {
               element {
                    dataType : audioData : g729 2
               },
               element {
                    dataType : redundancyEncoding {
                         primary {
                              dataType : audioData : vbd : g711Alaw64k 160
                         },
                         secondary {
                              dataType : audioData : vbd : g711Alaw64k 160
                              payloadType 97 -- The PT for the redundant encoding
                         }
                    },
                    payloadType 101 -- The PT for the RFC 2198 packet
               },
               element {
                    dataType : data {
                         application : genericDataCapability {
                              -- SSE capability
                              capabilityIdentifier : standard {
                                   itu-t(0) recommendation(0) v(22) 150 sse(1)
                              },
                              nonCollapsing {
                                   {
                                        parameterIdentifier : standard 0,
                                        parameterValue : octetString "3,5"
                                             -- A comma-separated string
                                             -- of supported events (this string
                                             -- illustration of syntax and is not
                                             -- necessarily an appropriate list)
                                   },
                                   {
                                        parameterIdentifier : standard 1,
                                        parameterValue : logical
                                   }
                              }
                         }
                    },
                    payloadType 102     -- The PT for the SSE packets
               },
               element {
                    dataType : data {
                         application : genericDataCapability {
                              -- MoIP capability
                              capabilityIdentifier : standard {
                                   itu-t(0) recommendation(0) v(22) 150 moip(0)
                                   major-version-one(1) minor-version-one(1)
                              },
                              nonCollapsingRaw '0000'H
                                   -- This value shown is only presented
                                   -- for illustration and is not
                                   -- a valid value
                         }
                    },
```

```
                    payloadType 103     -- The PT for the MoIP packets
                }
            }
        },
        multiplexParameters : h2250LogicalChannelParameters {
            sessionID 1
        }
}
```

P.8.1 Extended fast connect

Extended Fast Connect [P-2] should be used in order to reconfigure logical channels, as it is much faster than exchanging a series of H.245 messages. If an endpoint needs to transition from audio operation to MoIP operation and does not presently have an open channel suitable for usage with MoIP, it should first attempt to reconfigure channels using Extended Fast Connect.

Extended Fast Connect should also be the first preference for logical channel signalling even when existing channels do support MoIP. For example, if the endpoint wishes to exchange the G.729 audio codec within an MPS with the G.723.1 audio codec, it should attempt to reconfigure the logical channels via Extended Fast Connect, as opposed to using H.245 signalling.

P.8.2 H.245 signalling

H.245 logical channel signalling via the H.245 Control Channel may be employed to configure or reconfigure media streams as necessary. MoIP-capable endpoints shall support H.245 Tunnelling when there is a need to utilize an H.245 Control Channel. However, it should be understood that support for H.245 Tunnelling does not guarantee that it will be utilized and that a separate connection may be necessary, though discouraged.

While signalling the opening of new channels is typically not an issue for H.323 endpoints, the possibility does exist that two endpoints may attempt to open channels independently which results in an incompatible configuration. To resolve such issues, the master shall reject the OLC proposals from the slave device with the reason **masterSlaveConflict**. The master should then send a **RequestMode** message to the slave device to propose a compatible mode of operation.

If an endpoint determines that it is necessary to switch modes of operation, for example, to switch from audio-only mode to a mode that supports MoIP, the endpoint shall send a **RequestMode** message to the other endpoint. For example, assume that two endpoints open G.729 audio in each direction and, then one endpoint determines a need to change the mode of operation from audio to MoIP. The endpoint shall send a **RequestMode** message over the H.245 Control Channel indicating the desired mode of operation. The receiving endpoint shall respond with an acknowledgement or a rejection message, as appropriate, but it should make every effort to accept the requested mode of operation. The endpoints should exchange messages in a manner similar to that shown in Figure P.2. As much as possible, messages should be exchanged in parallel to reduce mode transition delays.

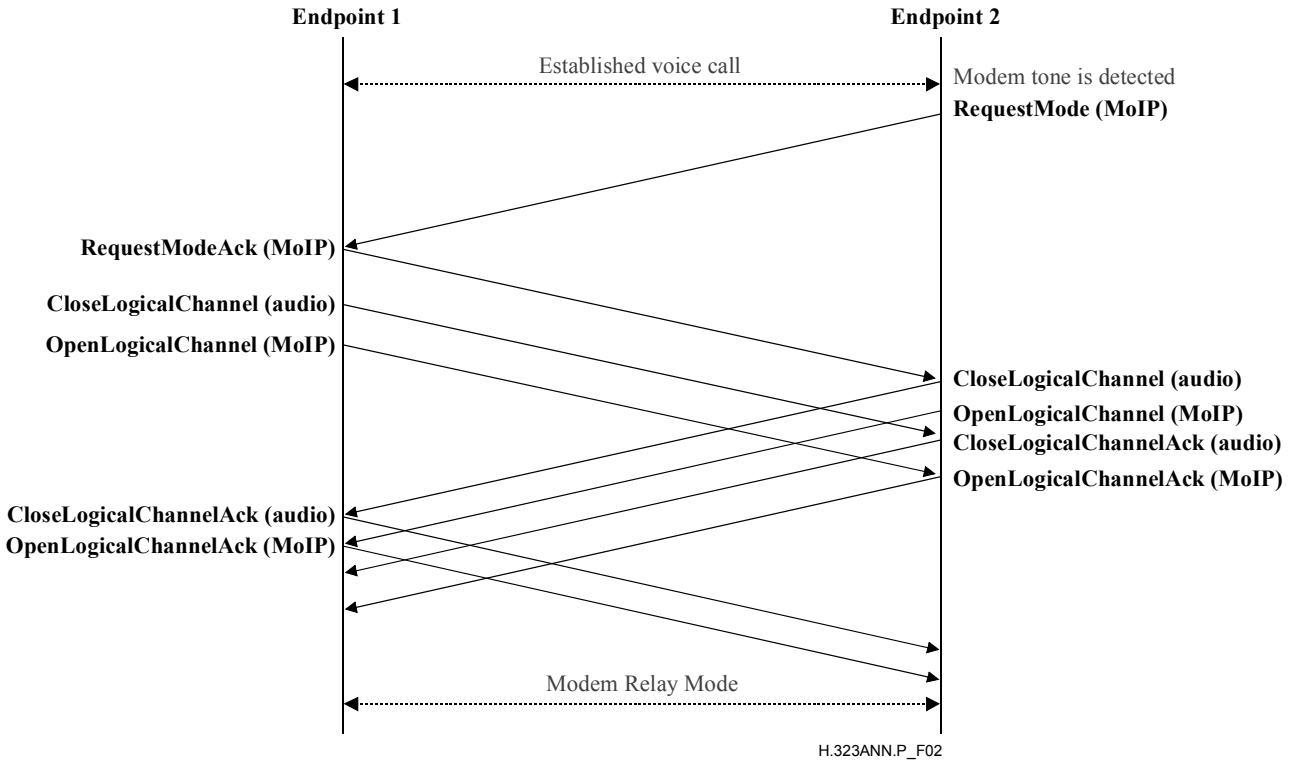


Figure P.2/H.323 – Successful switch between audio and MoIP modes

Annex Q

Far-end camera control and H.281/H.224

Q.1 Scope

The purpose of this annex is to provide far-end camera control protocol based on H.281/H.224. It also permits an H.323 endpoint to run any H.224 application using the IP/UDP/RTP/H.224 protocol defined in this annex.

Q.2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[Q-1] ITU-T Recommendation H.224 (2000), *A real time control protocol for simplex applications using the H.221 LSD/HSD/MLP channels*.

[Q-2] ITU-T Recommendation H.281 (1994), *A far end camera control protocol for videoconferences using H.224*.

[Q-3] ITU-T Recommendation T.140 (1998), *Protocol for multimedia application text conversation*.

Q.3 Introduction

The protocol described in this annex may be used to support far-end camera control (FECC) in this Recommendation using the stack IP/UDP/RTP/H.224/H.281. This protocol supports both point-to-point and multipoint scenarios.

This method may be used as a "simple" FECC scheme when the more sophisticated features of H.282/H.283 are not needed.

This method shall be used for FECC thru H.320-H.323 and H.324-H.323 gateways when the H.320 or H.324 endpoints do not support ITU-T Rec. H.282.

The requirements given below apply only in the case that the protocol described in this annex has been selected, using the normal procedures of ITU-T Rec. H.245.

It is allowed to run any H.224 application using the IP/UDP/RTP/H.224 protocol defined in this annex. The only other currently standardized H.224 application is ITU-T Rec. T.140.

Q.4 Far-end camera control protocol

Q.4.1 General

This protocol is based on ITU-T Rec. H.281 running over ITU-T Rec. H.224 in an RTP/UDP channel.

On IP transport networks, the H.224 protocol octet structure shall be the same as Figure 2/H.224 except that the HDLC bit stuffing, HDLC flags and HDLC Frame Check Sequence shall be omitted. The entire remaining content of each frame shall be placed in a single RTP packet.

References in ITU-T Rec. H.224 to the LSD channel of ITU-T Rec. H.221 shall be interpreted as referring to the H.224 logical channel as described in this annex. The maximum transmission time requirements of ITU-T Rec. H.224 shall be met, with the H.224 logical channel considered as operating at 4800 bit/s, regardless of the actual bit rate of the channel.

This protocol shall run over RTP in a unidirectional unreliable H.245 logical channel. The RTP payload value shall be dynamic. The payload descriptor field of H.245 **RTPPayloadType** shall use the H.224 Object ID.

Terminal numbering according to the procedures in ITU-T Rec. H.243 shall be used in order to support the data link layer in multipoint. The MCU/Terminal address pair <M><T> shall be used to uniquely identify each terminal in a conference. The special destination address of <0><0> shall be used as the broadcast address. The special source address <0><0> shall indicate that the sender does not know its address. An address with the terminal number set to 0 indicates the MC. For example, <n><0> indicates MC number n.

In a point-to-point call, when only two terminals are involved, then the terminals do not have an <M><T> address. In this case, the <M><T> source and destination addresses shall be always <0><0>.

In a centralized conference, an H.224 channel shall be opened between each terminal and the MC. When a terminal sends an H.224 packet, the MC shall forward the packet to the destination terminal by either retransmitting each packet to all other connected terminals or, by selectively retransmitting each packet only toward the destination terminal. The decision which method to use is up to the MCU manufacturer.

In a decentralized multicast conference, each terminal shall multicast the FECC packet to all other terminals. The MC is not involved in forwarding the packets. Terminal numbers per ITU-T Rec. H.243 shall be used to identify the source and destination terminals.

In decentralized multi-unicast conferences, each terminal shall use a separate logical channel to each far-end terminal to which it wants to send H.224 packets.

Q.4.2 H.320 to H.323 gateways

H.320-H.323 gateways shall insert and remove HDLC flags, HDLC bitstuffing, and HDLC Frame Check Sequence(s) as appropriate in each direction, so that the bitstream on the H.320 side shall conform with ITU-T Rec. H.224, and the bitstream on the H.323 side shall conform with the paragraphs above.

Q.4.3 H.324 to H.323 gateways

H.324-H.323 gateways shall insert and remove HDLC flags, HDLC octet-stuffing, and HDLC Frame Check Sequence(s) as appropriate in each direction, so that the bitstream on the H.324 side shall conform with the use of ITU-T Rec. H.224 as described in ITU-T Rec. H.324, and the bitstream on the H.323 side shall conform with the clauses above.

Q.4.4 H.245 signalling

The use of this protocol shall be signalled by the **GenericCapability** part of the **DataApplicationCapability** sequence in H.245. The Generic Capability for H.224, described in ITU-T Rec. H.224, shall be used. This shall be placed in the **receiveAndTransmitDataApplicationCapability** part of the **Capability** choice.

This protocol shall not be signalled in the **receiveDataApplicationCapability** or **transmitDataApplicationCapability** parts of the **Capability** choice.

Q.5 RTP header information

The following fields shall be filled in the RTP header:

V:	2
M:	0 NA
PT:	The same number sent in the OLC dynamicRTPPayloadType field
Sequence number:	Filled, incremented by one for each RTP packet sent
Timestamp:	Filled with 8 kHz clock rate
SSRC:	Filled with the synchronization source

Annex R

Robustness methods for H.323 entities

R.1 Introduction and scope

This annex specifies methods that can be used by H.323 entities to implement robustness or tolerance to a specified set of faults. Methods for recovery of call signalling (ITU-T Rec. H.225.0) and call control signalling (ITU-T Rec. H.245) channels are specified. RAS (ITU-T Rec. H.225.0) does not involve a connection and recovery, involving registering with an alternate gatekeeper, is covered elsewhere and so not specified in this annex. Recovery of Annex G service relationships is for further study.

H.323 calls require the cooperation of two or more H.323 entities. Call state information is distributed among the various entities involved in the call. Call signalling may depend on persistent connections between some of the entities involved. If any entity fails and does not have a backup peer, it may not be possible to establish new calls. If any entity involved in an active call fails and the entity does not have a backup peer or that peer does not have a method of retrieving sufficient call state information, it may not be possible to continue with the call. This Recommendation

provides some support for building robust systems but the mechanisms are distributed throughout this annex and few, if any, procedures for using them are given.

This annex describes two alternative methods consisting of sets of mechanisms along with procedures for using them to build systems that can recover from a significant set of specified failures. One method is more appropriate for small-scale systems, uses simpler entities and does not recover as much call state information. The other method is appropriate for large-scale systems and can recover as much state information as desired but requires more complex entities. The two methods share several mechanisms and can be used concurrently in different parts of a system.

R.2 Normative references

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

[R-1] ITU-T Recommendation H.225.0 (2003), *Call signalling protocols and media stream packetization for packet-based multimedia communication systems.*

[R-2] ITU-T Recommendation Q.931 (1998), *ISDN user-network interface layer 3 specification for basic call control*.

[R-3] ITU-T Recommendation X.680 (2002), *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation.*

R.3 Definitions

In addition to terms defined in ITU-T Rec. H.323, the following terms are used:

R.3.1 backup entity or backup peer: A peer of an entity that can take over the entity's functions if the entity fails.

R.3.2 peer entities: Two entities of the same type in an H.323 system, e.g., two gatekeepers. The two entities may cooperate in a call (e.g., originating and terminating gatekeepers in gatekeeper-routed call signalling) or one may provide backup for the other.

R.3.3 robustness methods: Procedures and mechanisms that allow recovery from the failure of one or more H.323 entities. The extent of recovery varies between different robustness methods and may include preservation of active calls in a stable state or only the ability to place new calls. Methods described in this annex are usually able to preserve active calls.

R.3.4 signalling neighbour: Other entities to which a specific entity has direct call signalling or call control signalling connections for a specific call. For example, a gatekeeper using the gatekeeper-routing model may have a direct call signalling connection for a specific call to a gateway and to another gatekeeper. These two other entities would be the gatekeeper's signalling neighbours for that call.

R.3.5 stable calls: A call is considered stable or in a stable state after Connect has been sent or received and media channels in both directions are established (though H.245 or fast connect procedures). A call becomes unstable when Release Complete has been sent or received. Certain Facility commands used to change call signalling connections may also cause a call to be considered unstable. This version of this Recommendation offers methods to preserve only stable calls during recovery.

R.3.6 tandem entities: Two (or more) peer entities, where all but one act as backup entities for one active entity.

R.3.7 virtual entity: Two (or more) closely-coupled peer entities that collectively appear as a single entity to the rest of an H.323 system while providing fault recovery.

R.4 Abbreviations

This annex uses the following abbreviations:

CRV Call Reference Value

GK Gatekeeper

GW Gateway

RAS Registration, Admission and Status

SCTP Stream Control Transmission Protocol (IETF RFC 2960) (Informative use only)

SDL Specification and Description Language

TCP Transmission Control Protocol

UDP User Datagram Protocol

R.5 Overview of the two methods

Two robustness methods are offered by this version of this annex.

The problem we are trying to solve is to recover from a failed H.323 entity. The goal is to preserve as many active calls as possible. As a minimum we wish to preserve all calls in a "stable" state. Calls not yet fully connected or in the process of tear-down may be lost. It is also a goal to preserve most relevant billing information, such as call start time, stop time, etc., even if maintained in the failed entity (e.g., routing gatekeeper).

It is assumed that the failed entity has one or more designated backup entities, although the small-scale solution may allow recovery when the failed entity returns to service quickly. Two basic problems must be solved to recover signalling for active calls:

1) Redirecting/re-establishing signalling to the backup entity.
2) The backup entity must recover sufficient call state information that had resided in the failed entity.

The two methods are primarily distinguished by the method of recovering state information about the active calls and the amount of information recovered.

R.5.1 Method A: State recovery from neighbours

In Method A, each entity is aware of the signalling transport addresses for backup entities for each upstream and downstream signalling neighbour. When entities become aware of the failure of their upstream or downstream signalling neighbour, they attempt to connect to one of the backup entities. The backup entity recovers minimal call state from its signalling neighbour using Status and StatusInquiry messages (enhanced with additional fields). Note that in some cases it may be necessary for the neighbour to query its neighbour for call state if it has not kept all the necessary information locally (e.g., a routing gatekeeper may not have cached open logical channel information).

The recovered call state is sufficient to continue the call (forward call signalling and call control signalling and know of open logical channels) but not sufficient to allow the recovered entity to participate in billing and some other services.

R.5.1.1 Partial method A

There is also a case where an H.323 entity does not itself have a backup entity but it still implements robustness procedure so it can help preserve calls, if its signalling neighbour that does have a backup entity fails.

The H.323 entity that participates in the recovery of stable calls with the backup entity of its signalling neighbour, but does not itself have any backup, is said to implement Partial Method A.

R.5.2 Method B: State recovery from a shared repository

The second architecture depends on a fault-tolerant pseudo-entity. This may be implemented by:

1) Using a fault-tolerant platform/OS.
2) A pool of non-fault-tolerant entities that share call state information through shared-memory, shared-disks, or through messages. The mechanism for sharing is not specified in this Recommendation.

The real entities in this fault-tolerant pseudo-entity must share sufficient state information with its peer entities to allow recovery of the desired call-state without any assistance from its signalling neighbours. This Recommendation will define the minimum information entities that must be shared. Any additional information that is desired to be recoverable can be shared. We note that method B will require that all entities in the pool constituting the pseudo-entity be from the same vendor since the sharing mechanism is not standard. The group would suggest one or two possible solutions and will consider recommending a standard sharing mechanism in H.323 versions beyond version 4.

More details of this architecture will be given below.

R.5.3 Comparison

Each of these two architectures has advantages, which makes the choice less than obvious. Some of the issues are listed below.

The Recovery from Neighbour approach:

1) allows simpler entities;
2) adds less overhead before a failure (still need keepAlive messages in some cases),

but:

1) requires more changes to H.323 messages;
2) makes recovery somewhat slower (due to the Status and StatusInquiry messages);
3) is non-scalable; only suitable for small-scale systems.

The Shared Repository approach:

1) hides most of the recovery process from H.323 and so requires fewer changes to existing messages;
2) makes recovery faster;
3) allows future use of state-maintenance protocols that might be implemented below the H.323 application layer. (See Informative Note 2 in R.13.);
4) can support recovery of billing information and other desired state information,

but:

1) adds significant overhead to all signalling (before failure);
2) requires more complex entities or pseudo-entities.

R.6 Common mechanisms

The two methods share several common mechanisms.

R.6.1 Detection of TCP based connection lost

In case of Network failure, the first "automatic" attempt would be on the IP routing protocol level. If it does not succeed, the TCP failure will be reported to both sides (entity and its signalling neighbour, e.g., gatekeeper and endpoint). Either a network failure or failure of the signalling neighbour will appear as a TCP failure.

When the call was set up, it was determined whether the entity's neighbour supported robustness procedures.

In the case where one of the sides does not support the defined Robustness Procedure, it is suggested to release the call because of TCP connection failure.

On the Endpoint side, in case both sides support the Robustness Procedure, it is suggested to maintain a reasonable timeout for the robustness procedure to be initiated by the other side. This timeout is necessary in order to address a potential Network connectivity problem. After the timeout is expired, internal resources (consumed by the call) should be released.

R.6.2 Protocol failure handling

For entities that use this annex, if a protocol failure occurs in an H.245 Control Channel and both signalling neighbours support robustness, the channel and all associated logical channels are **not** closed (contrary to 8.6). Recovery procedures of this annex are instead attempted.

R.6.3 Detecting failure – KeepAlive

Without a keepalive mechanism, entity failure or failure of the signalling connection will be known only when the connection is used. Annex E provides a keepAlive mechanism to detect the failure even with little traffic. TCP's keepAlive mechanism has too long a timeout to be of use and so with TCP failure might not be detected for an extended time under conditions of low traffic sent to the failed entity. Our small-scale solution depends on failure being detected by both signalling neighbours (connections are always established from the neighbour toward the recovered entity) and so we need keepAlive messages at the H.323-level that can be used with TCP connections. KeepAlive messages are available to be optionally used in H.245. We would specify that Status/Status Inquiry be used periodically over TCP connections to provide this keepAlive mechanism. Although this issue is common, we will see that it is only a significant problem for the Method A, the state recovery from neighbour method.

The entity closer to the called party (destination side of connection or side that uses call reference flag = 1 in CRV used on connection – See ITU-T Rec. Q.931 for definition of the call reference flag) shall send StatusInquiry periodically (this is the direction of least traffic during established calls). The period should vary randomly from a configurable maximum value to one half that value in order to avoid congestion. Two seconds is the recommended default maximum, in order to allow detection of failure before other messages timeout. The maximum value shall be included in the StatusInquiry as timeToLive, so that the recipient can also monitor failure without an additional StatusInquiry/Status exchange in the opposite direction. The recipient system needs only to maintain a timer using the indicated maximum value as a timeout.

When multiplexed channels are used, it is not necessary to send StatusInquiry/Status for each call signaled on the channel. A StatusInquiry or Status message with a CRV IE of 0 (zero) and with the field callIdentifier of 0 (zero) applies to all calls using the channel.

KeepAlive messages, especially at the H.323-level, can add significant signalling overhead. But note that only Method A with TCP connections uses these KeepAlives and Method A is for the small-scale case where the number of connections per entity is low. To minimize the overhead, the

use of TCP should be avoided. StatusInquiry/Status keepAlives are **not** needed in our large-scale solution.

In order to further minimize the impact of exchanging keepAlives, if there are several calls between the same two entities, StatusInquiry/Status messages need be sent on any one of the connections between the two entities. In order to associate each active call with the correct set of entities, an endpoint GUID shall be included by the originating entity in the Setup message and another by the destination entity in the Connect message. These GUIDs shall be unique to each entity and, in case any entity has more than one signalling interface, shall be generated per interface. If there are multiple H.323 instances on the entity, each instance shall generate a unique GUID. KeepAlive timers shall be maintained on each unique GUID pair. Upon the expiry of the keepAlive timer, any entity may send a StatusInquiry message with a CRV IE of 0 (zero) and with the field callIdentifier of 0 (zero) using any available connection. The signalling neighbour shall respond with a keepAlive Status message.

Detection of failures for Annex E connections will be made using the existing I-Am-Alive messaging. The procedure described above defines keepAlive messages between the signalling entities based on a timer. This timer uses a value defined by T-IMA1 timer, by default set to 6 seconds. However, in the case where the two entities also implement Annex R, this timer shall be configurable in accordance with recommended values as above. The I-Am-Alive messaging also uses the a counter defined by the N-IMA1, that defines the number of consecutive retries of I-Am-Alive messages before which the signalling neighbour is assumed to have failed. For Annex R enabled entities, this counter is recommended to have a maximum value of two (2).

R.6.4 Transport address and re-established connections

Both of these solutions (with the possible exception of some fault-tolerant platform solutions) must deal with recovery of the signalling channel using a backup transport address. These must be exchanged when call signalling is established, using the backupCallSignalAddresses fields in Setup and Connect. An entity sends the call signalling address of its backup in both Setup and Connect. An entity receives the call signalling address of the backup entity from its origination-side neighbour when it receives Setup and from its termination-side neighbour when it receives Connect.

An entity that implements Partial Method A shall send an empty **backupCallSignalAddresses** to indicate that it does participate in robustness procedure but it does not itself have a backup.

All entities shall add their own call signal address as the first entry in the **backupCallSignalAddresses** list including the port number on which they are listening. This is required for the signalling neighbour (or its backup) to re-establish connection with the entity.

R.6.4.1 Establishment of a new TCP connection

An entity that detects loss of a call signalling channel with a signalling neighbour shall try to re-establish the channel using the backup transport address. Alternatively, the entity detecting failure may attempt to probe its original signalling neighbour using methods outside the scope of this Recommendation (e.g., ping) and, if it believes the original signalling neighbour maybe usable, it may try to re-establish the channel to the original signalling neighbour before trying the backup transport address. Implementers choosing this option should be aware that attempting to establish a TCP connection to a non-responding entity may cause significant delays.

The re-established call signalling channel will assume the state of the previous – not behave as a new channel (it will **not** begin with Setup). See further detail below for ensuring synchronization of state between signalling neighbours.

INFORMATIVE – An alternative is to use SCTP for transport rather than TCP. SCTP channels are associated with a list of alternate transport addresses that can be used as needed to maintain the channel with no intervention by the application layer. Note that more information about using SCTP is given in Informative Note 2 in R.13.

R.6.4.2 Association between the call and the new TCP connection

The association between the Call and the new TCP connection (in the endpoint side) shall be done by retrieving the callIdentifier value from the messages received on the new TCP connection.

R.6.4.3 An old TCP connection closure

After the new connection is opened, there might be two TCP connections opened, belonging to the same call on the side that did not fail. There are two options in this case:

1) The TCP connection was lost after the SETUP message was sent (and received). In this case the side that did not fail shall identify the situation and close the connection. This should be accomplished by detecting an identical callIdentifier for both connections.

2) The TCP connection was lost before the first message was transferred.

In that case, the side that did not fail has no way to find the relation between the first (old) and the second (new) TCP connections. This may be solved by a procedure that will enable the receiving side to close a connection if it is opened for a while and no message was received on it within a pre-defined timeout. (This procedure is not described in this annex.)

R.6.5 Support for extended status

To enhance interoperability between the two methods, all entities supporting robustness shall support the extended Status message including the fastStart field. This will allow an entity with a shared-repository to cooperate with a neighbour requiring Status for state recovery.

R.7 Method A: State recovery from neighbours

R.7.1 Introduction

Currently ITU-T Recs H.323 and H.225.0 do not explicitly define procedures for connection failure detection and recovery. The purpose of this method is to introduce a procedure for:

- detection of TCP based connection failure;
- synchronization between the two sides of the connection in terms of Call State;
- definition of recommended behaviour on each side in order to renew the Call Signalling connection and proceed with the call as normal in each State of the Call.

The main motivation to sustain a call (when a connection is lost) is in situations where a gatekeeper, that handles a big number of calls, fails due to hardware or software problem. In this case a control can be transferred to a standby gatekeeper (this gatekeeper may hold all the information about the calls by means of some common database). The procedure defined and presented in this annex addresses this case of Gatekeeper failure and enables the managed calls to proceed without any interruption.

This procedure does not address all the aspects of TCP based connection failure and recovery in other possible cases and topologies. Nevertheless, it is possible to address additional cases in a similar manner in the future.

R.7.2 Scope

This proposal addresses TCP based connections only (H.225.0 call signalling and H.245 call control channels). UDP (RAS) channels will not be discussed since their failure situations are already covered using the retries mechanism defined for UDP channels.

R.7.3 The robustness procedure

After a failure the H.323 Entity shall re-establish the Call Signalling connection and shall send both STATUS INQUIRY and STATUS messages to the other H.323 Entity. The other H.323 Entity shall respond with a STATUS messages, thus reaching a state where both sides are aware of the Call

State of the other side. If the receiving entity is unaware of the call, it shall respond with a STATUS message with CallState IE set to NULL. The Call Signalling connection should be established to one of the entries in **backupCallSignalAddresses** in the order of preference defined by the order of elements in **backupCallSignalAddresses** structure.

In the event that both entities simultaneously initiate Call Signalling connection, the entity with the numerically smaller value of TransportAddress used from **backupCallSignalAddresses** shall close the TCP connection it opened and use the connection opened by the other endpoint. For purposes of comparing the numeric values of TransportAddress from **backupCallSignalAddresses**, each octet of the address shall be individually compared beginning with the first octet of the OCTET STRING and continuing through the OCTET STRING left to right until unequal numeric octet values are found. Comparison shall first be performed on the network-layer address element of the TransportAddress from **backupCallSignalAddresses**, and, if found to be equal, then on the transport (port) address element. See Figure R.1.

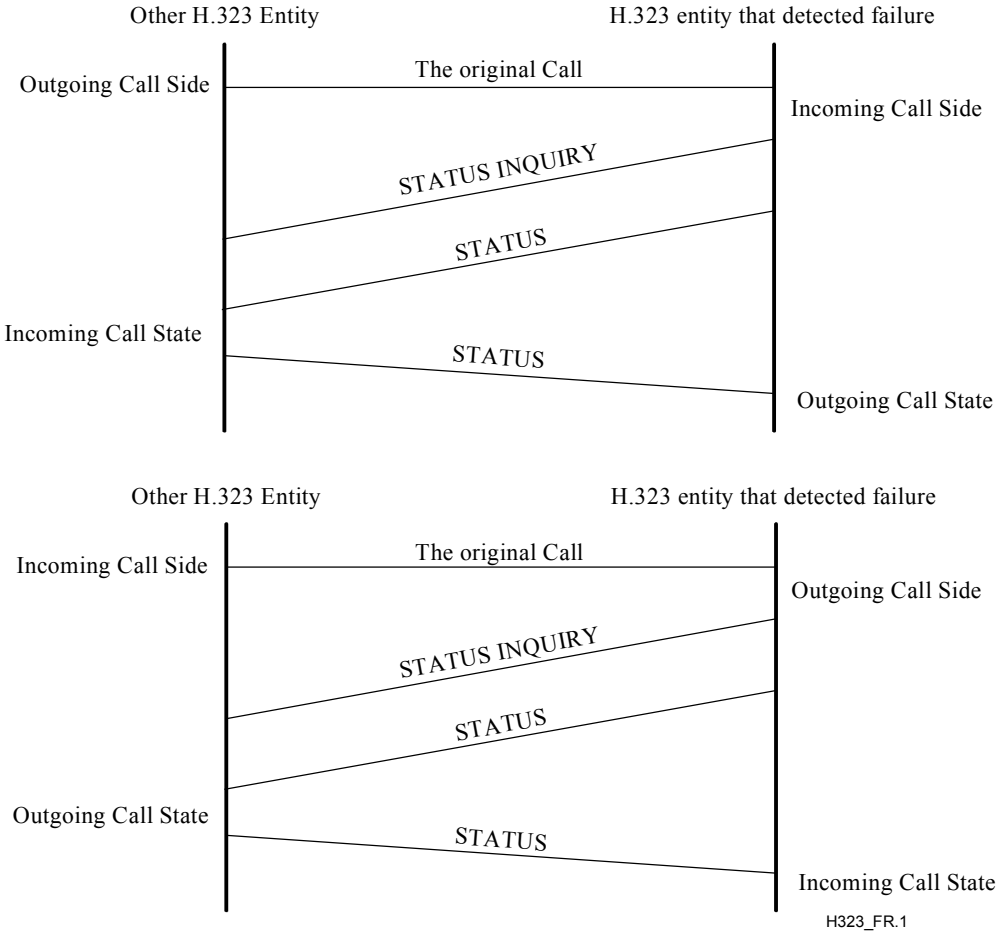


Figure R.1/H.323 – Robustness procedure

Any previous connections that might be still open for the call shall be closed, this applies both to the Call Signalling connection and the Call Control connection.

To facilitate synchronization of the logical channels state, the new fields **IncludeFastStart** in STATUS INQUIRY and **RobustnessFastStart** in STATUS message may be used. Sender of the STATUS message should include the **RobustnessFastStart** field containing the currently active receive and transmit channels with the receive addresses for media and media control streams. Sender of the STATUS INQUIRY message may request inclusion of the **RobustnessFastStart** field in the STATUS message by setting **IncludeFastStart** to TRUE.

If an intermediate entity needs to synchronize the logical channel state it should send the STATUS INQUIRY message to one of the call legs, should wait to STATUS message with fast start field, should issue the STATUS and STATUS INQUIRY message to the other leg of the call, should wait for STATUS message with the second leg logical channel information and the should send the STATUS message to the first leg of the call.

This procedure is used to synchronize the states of the logical channels that were opened through both fast start procedure and H.245 logical channel establishment procedure.

In situations where the call before failure has not reached the active state, the call should be dropped.

Both the recovering H.323 entity and its signalling neighbour shall implicitly reset their H.245 state machines for the call as the recovering entity is not aware of any remote terminal capabilities or the knowledge of the result of MSD negotiations. Moreover, the recovering entity's capabilities may differ from the failed entity. Before any H.245 messages are sent, both entities shall exchange TCS messages and make the Master/Slave determination.

R.7.4 SDL for Method A state machine

See Figure R.2.

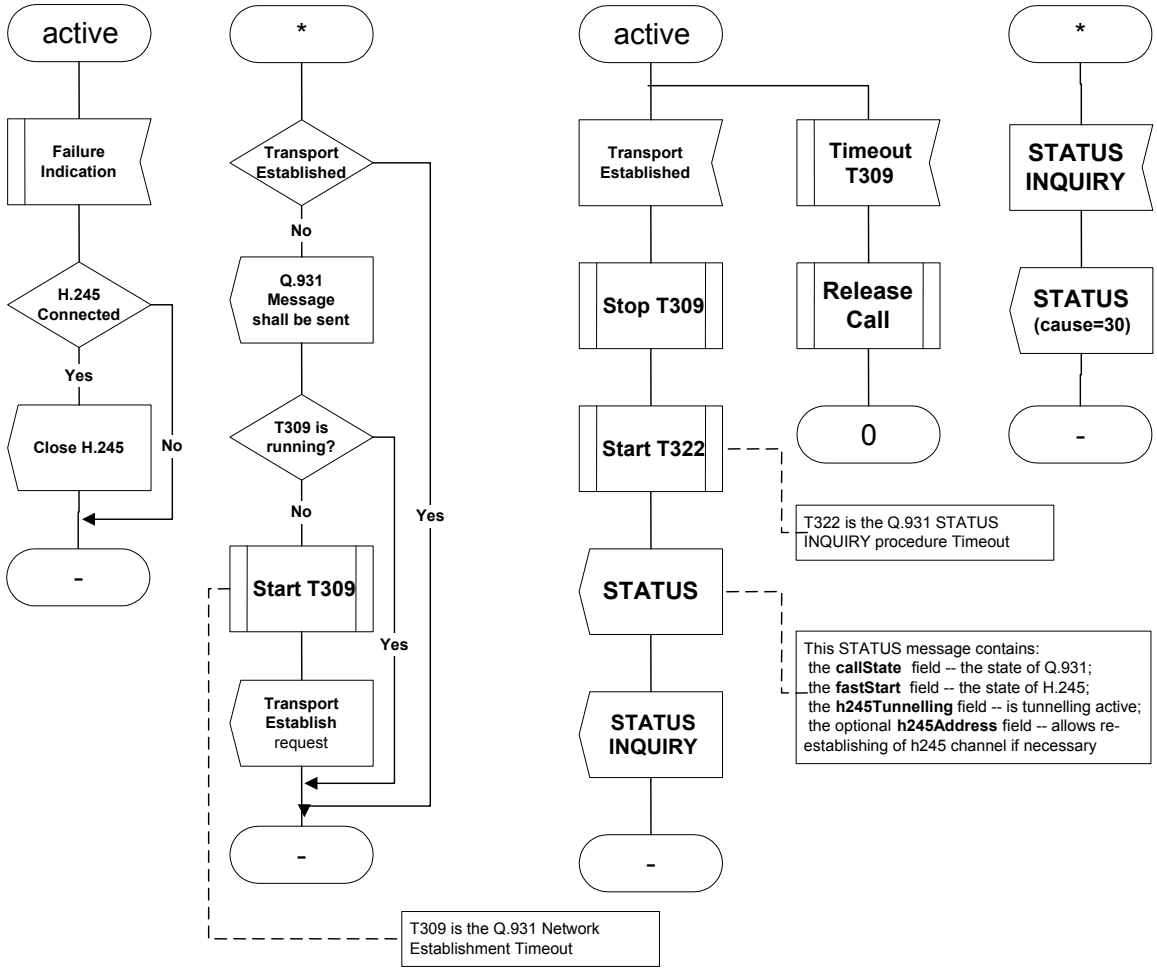


Figure R.2/H.323 – Method A state machine ***(sheet 1 of 2)***

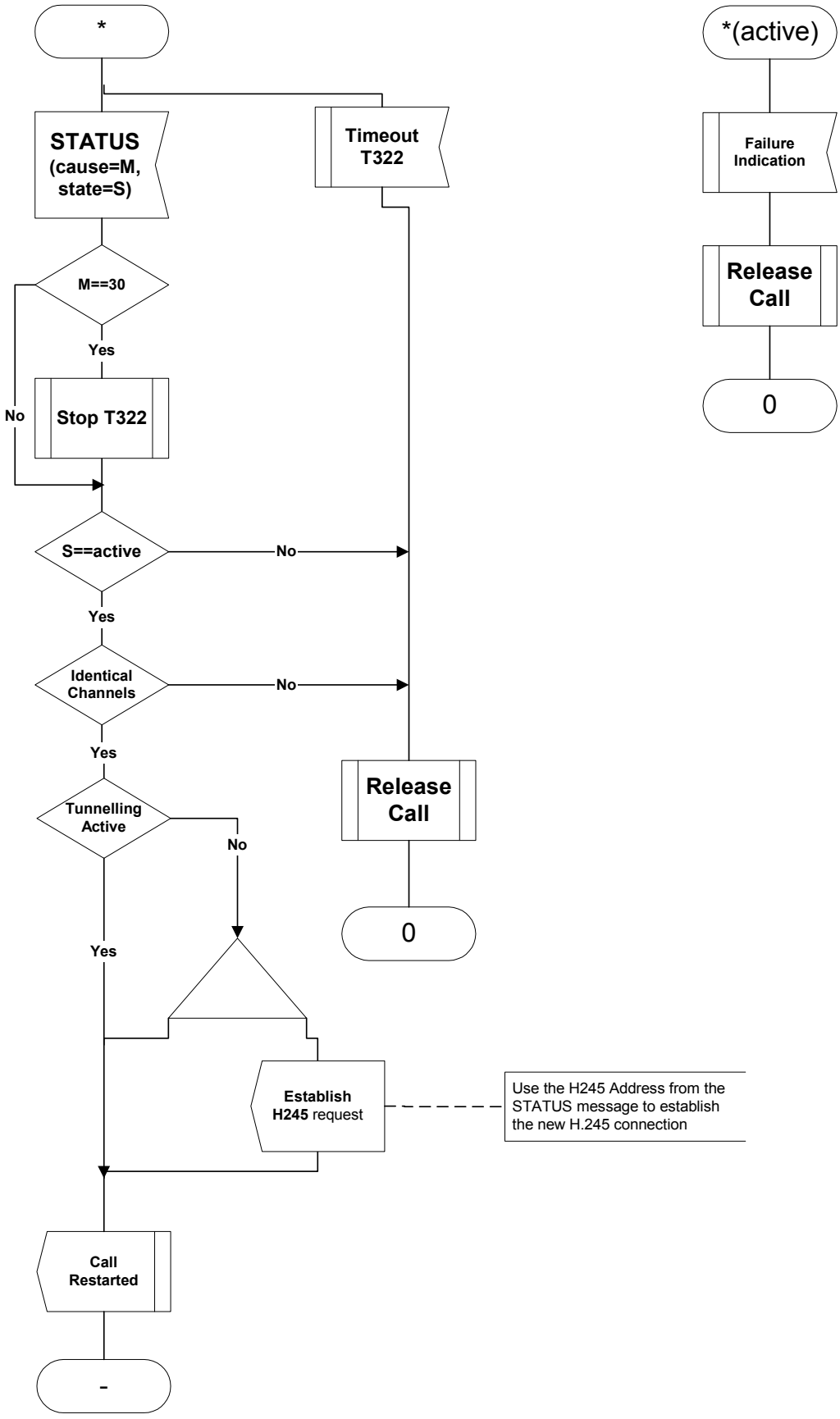


Figure R.2/H.323 – Method A state machine ***(sheet 2 of 2)***

R.8 Method B: State recovery from a shared repository

This method depends on a fault-tolerant entity or pseudo-entity and (if the backup entity requires a different signalling address) a mechanism to re-establish call signalling to the backup. There are several ways this can be done. The fault-tolerant mechanism will not be standardized in this version of this Recommendation but we will suggest some solutions. We may recommend standardizing the solution in a future version of this Recommendation. There are some emerging IETF protocols that may help solve this problem but these are not yet in a state that could be referenced by H.323v4 (November, 2000).

R.8.1 Fault-tolerant platform

One solution is to implement the robust entity on a fault-tolerant platform that uses hardware and OS support. Such a solution would make state recovery completely transparent to H.323. If the platform also maintains a constant transport address then it is a fault-tolerant virtual entity, the signalling channel will not fail and no application level procedures are needed. If the transport address changes, then the mechanism of this clause will be needed.

R.8.2 Fault-tolerant cluster

Another solution is to establish a cluster (two or more) of non-fault-tolerant tandem entities, that collectively behave as a fault-tolerant pseudo-entity. The entities of the cluster would arrange to share specified call state information sufficient to allow a peer to take over in the event of the failure of the active entity. Solutions can include:

1) active/spare ("1+1");
2) single spare shared by several active entities (spare sharing state information with each active entity that it might substitute for) ("N+1");
3) and others configurations.

Although state information is shared, allowing the cluster to appear like a fault-tolerant virtual entity, it will not be able to maintain a constant call signalling transport address and so must use one of the mechanisms of R.8.3 to re-establish the call signalling channel.

One key problem for the cluster model is how to share state. State information must be synchronized at key times in the call, to which the system can safely fall back. We will call these times *checkpoints*. This Recommendation specifies the checkpoints and the minimal data items that must be shared. We do not suggest a standard solution for sharing in this version of this Recommendation but in Informative Note 2 in R.13, we will discuss several solutions to illustrate the practicality of this model.

R.8.3 Call signalling connection re-establishment

Sharing of backup signalling addresses is the same as with Method A. Re-establishment of call signalling connections is similar but has differences since the backup entity has sufficient information to re-establish the connection on the second side rather than wait for the other neighbour to also detect failure.

When a backup entity takes over for a failed peer and receives a message over a new connection, it would retrieve the call state (using the callIdentifier as key). This will allow it to continue supporting the call, including routing signalling, maintaining billing information, etc. An entity detecting a failure shall not re-establish the connection until it has a message to send over the connection. The backup entity will have new channels for each call that used the failed peer, unless multiplexed channels are used. The policy to re-establish only when needed will spread the re-establishments over time.

Delaying re-establishment until the channel is needed for a message and the fact that the backup entity has sufficient information to establish the new channel on the other side means that a keepAlive mechanism is not needed for Method B.

Since both the recovered entity and its signalling neighbour may re-establish the connection, there is a potential race condition but we avoid the need for keepAlive messages with TCP connections. Since traffic is greater in one direction than the other and re-establishment occurs only when there is message traffic, the race condition will be rare. We can resolve the race condition by the same methods used for H.245 channel establishment. The entity with the numerically smaller h245Address shall close the TCP connection it opened and use the connection opened by the other endpoint.

For multiplexed signalling channels, detecting a failure on any call shall imply failure of the channel. When a new channel is established it shall be used for the same set of calls as the failed channel. Note that this implies that the list of calls sharing a channel must be part of the data shared between an entity and its backup entity or entities through the shared repository. After a failure, the multiplexed channel is re-established when there is a message to send for any of the calls sharing the channel. A similar race condition exists to that in non-multiplexed channels. If two signalling channels are found handling the same set of calls or any calls from the same set, it shall drop one connection.

If an entity receives a new signalling connection with a callIdentifier matching that of an existing connection, it shall verify that the connection is from either the same entity as the earlier connection or the backup call signalling address for that same entity. If either is true, the entity receiving the new connection shall consider the earlier connection as failed and close it.

R.8.4 H.245 connection re-establishment

After the Call Signalling channel has been re-established and the robustness procedure has reached a stable state, if H.245 tunnelling was in use, the entities can continue tunnelling H.245 messages using the new Call Signalling channel.

If a separate H.245 connection was being used it may have also failed alone or along with the Call Signalling channel. If the entity has detected failure on an H.245 channel, it shall drop its connection without closing it (not sending EndSessionCommand, which would indicate to the other end that the call was over). It shall then attempt to establish a new connection by sending its h245Address in a Facility message to its signalling neighbour. An entity receiving Facility with an h245Address for a call for which it already has an H.245 channel (possibly failed but not detected) shall close that existing channel and open the new one. Neither entity shall perform H.245 initialization procedures (master slave determination and terminal capability exchange) for the new channel.

The recovering entity may have a different set of capabilities than that of the failed entity. In this case and especially when H.245 procedures were started between the signalling neighbours, the entities should restart their H.245 state machines and begin anew. This is done by using the **resetH245** flag in the STATUS robustness-data. After transmission of this flag, the entities should follow it up by exchanging TCS and MSD messages.

R.8.5 Data items shared through shared repository

As a minimum, the following data shall be shared through a shared repository:

1) backupCallSignallingAddresses;
2) hasSharedRepository;
3) callIdentifier;
4) openLogicalChannel structures, from H.245 or fastStart.

Additional data may be shared to support recovery of unstable calls or to allow recovery of additional data that changes during stable calls (e.g., call detail records, call timing data, billing data, authorization tokens).

R.8.6 Checkpoints

In this version of this Recommendation we only maintain calls that are in the stable state. Thus the only checkpoint needed is when entering the stable state. This occurs when Connect has been sent or received and media channels in both directions are established (through H.245 or fast connect procedures).

Entities may use additional checkpoints to support recovery of unstable calls or to allow recovery of additional data that changes during stable calls.

R.9 Interworking between robustness methods

Signalling neighbours must agree on the robustness method to be used between them. It is **not** necessary that the same method be used end-to-end.

Support for robustness (any of the methods) is indicated by the originating side entity by including RobustnessGenericData field in Setup. In addition support for Method B (Shared Repository) is indicated in hasSharedRepository field of Setup. The terminating side entity indicates its support for robustness and Method B by the same fields in Connect. The choice of Method A versus Method B is then made as indicated in R.10, Procedures for recovery.

If an entity routing call signalling supports Method B (has a shared repository), it may be required to use Method B on one connection and Method A on the other connection it has for the same call. In this case, it follows the rules in R.10 independently on the two connections. If a backup entity with a shared repository receives StatusInquiry, it may reply with Status using information in the shared repository.

R.10 Procedures for recovery

1) If neighbour does not support Method B (Shared Repository) and TCP signalling is used, then StatusInquiry keepAlives shall be used. If entity has Shared Repository (even though neighbour does not), then it shall send StatusInquiry periodically. If entity does not have a Shared Repository, then only the entity nearer the called party shall send StatusInquiry periodically.
2) If an entity has a message to send on a call signalling channel (including a keepAlive StatusInquiry) and it detects failure, then it shall attempt to establish a channel to the first address in backupCallSignalAddresses (backup entity).
3) After a call signalling channel is re-established, if the neighbour does not have Shared Repository, Method A shall be used and the establishing entity shall send a Status (with the fastStart field) before the message waiting to be sent.
4) The establishing entity may also send a StatusInquiry before the message, if it wishes to audit state consistency.
5) If an entity with Shared Repository receives StatusInquiry, it shall send StatusInquiry to its neighbour on the other side to retrieve necessary state information (including fastStart data) unless it keeps all such data in its repository.
6) If an entity not having Shared Repository receives a StatusInquiry wait until it receives a Status from its neighbour on the other side (sending StatusInquiry, if necessary, to the other neighbour if signalling channel on other side is available).

R.10.1 Recovery procedures with conflicting CRV values

It is possible that at the time of failure, the active entity and its backup peer are both simultaneously in calls to the same signalling neighbour. In this case there is the remote possibility that both of these calls use the same CRV values with the signalling neighbour and backup peer is not able to continue the call from the failed entity keeping the same CRV. Assignment of a new CRV and communicating it to the signalling neighbour is required.

If the failed entity implements Method A, the signalling neighbour re-establishes a call signalling connection with the backup entity of the failed entity. Then the signalling neighbour shall send StatusInquiry and Status messages to the backup entity. But before sending the StatusInquiry and Status messages, the entity shall check if it is the one that originated (caller side of) the call and if it already has prior calls to the backup entity. If the signalling neighbour is on the caller side of the call and it has prior calls to the recovered entity as shown in Figure R.3, then the signalling neighbour shall assign a new unique CRV value for this call to the recovered entity and use it (in CRV IE) in all subsequent H.225.0 call signalling and RAS messages. The recovered entity shall assign a unique CRV value for this call and use it in its communication with the gatekeeper. If the signalling neighbour is on the called side of the call and it has prior calls from the recovered entity as shown in Figure R.4, then the entity shall assign a new unique CRV value in StatusInquiry message with CRV flag = 1 because it is the destination side of the call. The recovered entity shall adopt this new CRV for this call. All the subsequent H.225.0 call signalling messages for this call shall use this new CRV value. The recovered entity, if required, shall assign a unique CRV value for this call to be used in RAS messages.

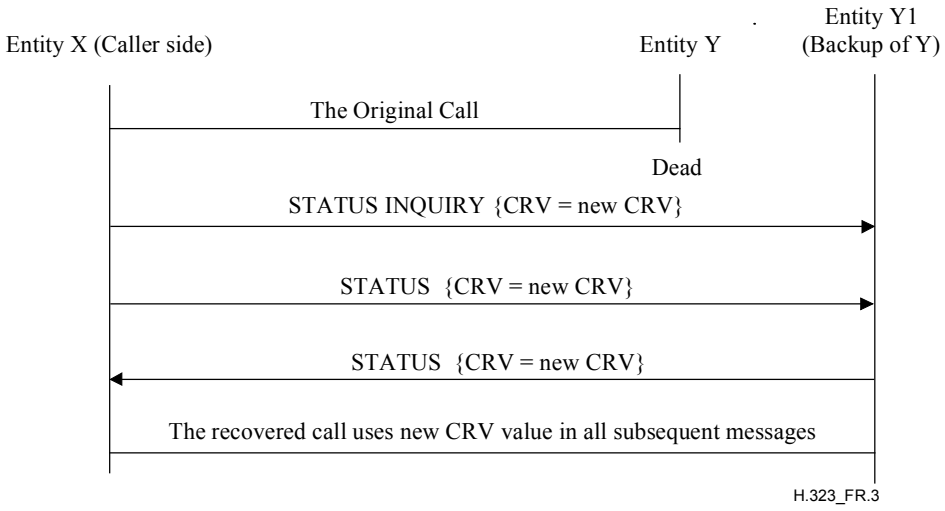


Figure R.3/H.323 – Failed entity is of Method A and called side

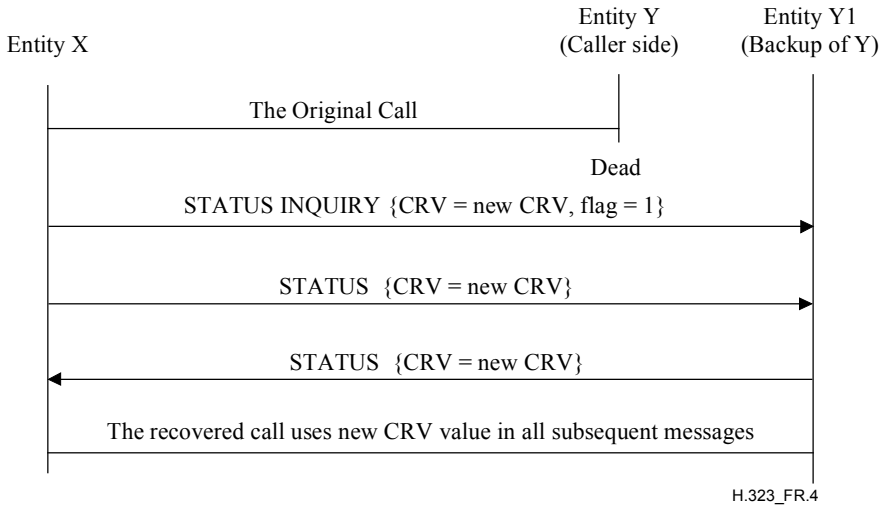


Figure R.4/H.323 – Failed entity is of Method A and caller side

If the failed entity implements Method B, the signalling neighbour or the backup entity of the failed entity can re-establish the call signalling connection. Whoever re-establishes the call signalling connection, before sending any H.225.0 call signalling messages, the entity shall check if it is the one that originated (caller side of) the call and if it already has prior calls to the recovered entity. If the entity that is re-establishing the connection is on caller side of the call and it has prior calls to the signalling neighbour as shown in Figure R.5, then the entity shall assign a new CRV value for this call and use it in all subsequent H.225.0 call signalling and RAS messages. The signalling neighbour shall assign a unique CRV value for this call and use it in RAS messages in its communication with the gatekeeper. If the entity that is re-establishing the connection is on called side of the call and it has prior calls from the signalling neighbour entity as shown in Figure R.6, then the entity shall assign a new unique CRV value and use it in a H.225.0 call signalling message with CRV flag = 1 because it is the destination side of the call. The signalling neighbour entity shall adopt this new CRV for this call. All the subsequent H.225.0 call signalling messages for this call shall use this new CRV value. The signalling neighbour entity, if required, shall assign a unique CRV value for this call to be used in RAS messages.

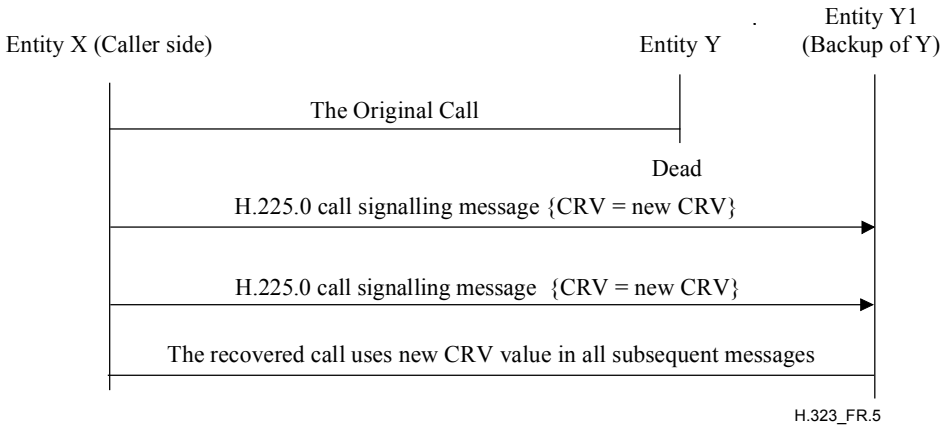


Figure R.5/H.323 – Failed entity is of Method B and called side and Survived entity initiates re-establishment

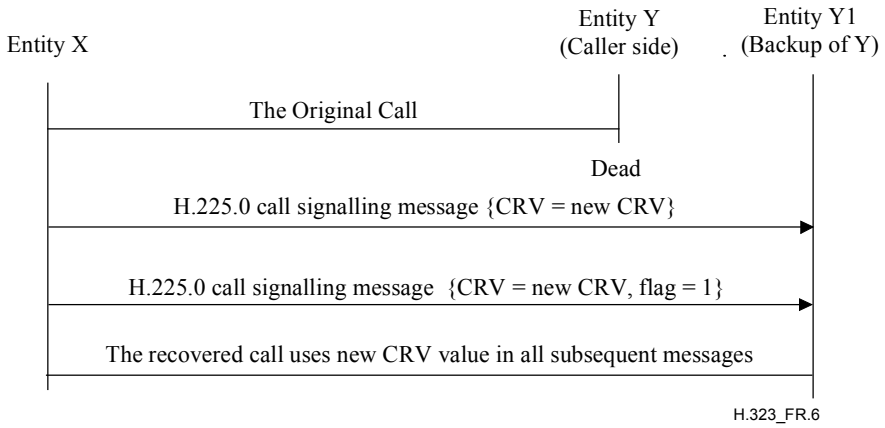


Figure R.6/H.323 – Failed entity is of Method B and caller side and survived entity initiates re-establishment

R.11 GenericData usage

The data fields necessary to implement this annex's features are carried in GenericData fields of various messages as defined below. `RobustnessData` shall be encoded and the resulting binary data carried as a raw instance of GenericData in the specified messages.

```
RobustnessData ::= SEQUENCE
{
     versionID          INTEGER (1..256),
     robustnessData     CHOICE {
          rrqData            Rrq-RD,
          rcfData            Rcf-RD,
          setupData          Setup-RD,
          connectData        Connect-RD,
          statusData         Status-RD,
          statusInquiryData  StatusInquiry-RD,
          ...
     },
     ...
}

BackupCallSignalAddresses ::= SEQUENCE OF CHOICE {
     tcp                TransportAddress,
     alternateTransport AlternateTransportAddresses,
     ...
}

Rrq-RD ::= SEQUENCE
{
     backupCallSignalAddresses   BackupCallSignalAddresses,
     hasSharedRepository         NULL OPTIONAL,
     ...
}

Rcf-RD ::= SEQUENCE
{
     hasSharedRepository     NULL OPTIONAL,
     ...,
     irrFrequency            INTEGER (1..65535) OPTIONAL     -- in seconds;
                                                             -- not present
                                                             -- if GK does not
                                                             -- want IRRs for
                                                             -- recovered calls
}
```

```
Setup-RD ::= SEQUENCE
{
     backupCallSignalAddresses   BackupCallSignalAddresses,
     hasSharedRepository         NULL OPTIONAL,
     endpointGuid                GloballyUniqueIdentifier OPTIONAL,
     ...
}

Connect-RD ::= SEQUENCE
{
     backupCallSignalAddresses   BackupCallSignalAddresses,
     hasSharedRepository         NULL OPTIONAL,
     endpointGuid                GloballyUniqueIdentifier OPTIONAL,
     ...
}

Status-RD ::= SEQUENCE
{
     h245Address    TransportAddress OPTIONAL,
     fastStart      SEQUENCE OF OCTET STRING OPTIONAL,
     ...,
     resetH245      NULL OPTIONAL
}

StatusInquiry-RD ::= SEQUENCE
{
     h245Address         TransportAddress OPTIONAL,
     timeToLive          TimeToLive OPTIONAL,
     includeFastStart    NULL OPTIONAL,
     ...
}
```

The GenericIdentifier shall be 1:

```
robustnessId GenericIdentifier ::= standard:1
```

In addition a featureDescriptor carrying the robustnessId shall be included in desiredFeatures of messages specified below.

R.11.1 GenericData usage in H.225.0 messages

RRQ, RCF, ARQ, ACF, Setup, Connect, Status, and StatusInquiry shall include RobustnessData in GenericData as per the data definitions for the respective messages.

All messages (RRQ, RCF, ARQ, ACF, Setup, and Connect) excluding the Status and StatusInquiry shall include the robustness FeatureDescr in desiredFeatures of featureSet. Note that the desiredFeatures is not inside featureSet in Setup.

The version of this data (versionID field in RobustnessData) shall be set to 1.

R.12 Informative Note 1: Background on robustness methods

This clause describes types of system failures and types of robustness from a general viewpoint. Not all types of system failures described are addressed by robustness methods in the current version of this annex. This more general view is provided to give context to the methods currently defined and help the reader understand which types of system failure are addressed. It also serves as a list of failures that might be addressed in future versions of this annex.

R.12.1 Types of robustness methods

System robustness can be provided in several ways:

1) hardware/operating system redundancy methods (possibly including several NIC cards);
2) tandem entities;
3) virtual entities.

R.12.2 Robust entities

Entities to be considered for robustness include essentially all H.323 entities:

1) Gatekeepers;
2) Border Elements;
3) Multipoint Controllers;
4) Possibly Multipoint Processors (for media stream failure);
5) Gateways (including IP-to-IP Gateways);
6) Firewall proxies; and
7) certain types of endpoints.

Not all robustness models may be suitable for all system components.

R.12.3 Robust system scope

The scope of robustness or the part of a system implementing robustness can include one or more of:

1) H.323 Zones (intra-zone, with one or more Gatekeepers).
2) H.323 Intra-Domain (intra-domain, inter-zone with several Gatekeepers).
3) H.323 Inter-Domains (inter-domain, with several Gatekeepers and Border Elements).

R.12.4 System termination and failures

Orderly system termination (such as an MC leaving a conference) should be catered for as well as system failure. Terminating orderly in principle allows the terminating endpoint to notify its peers thereby potentially simplifying detection but also requiring additional/slightly different mechanisms. It should be noted that the notification may not succeed due to repeated packet loss, so that the border to system failures is almost seamless.

System failure aspects are addressed in the following clauses:

R.12.4.1 Types of failures

The methods of this annex only address failures that can be detected from a protocol "on the wire" point of view. Failure of a processor on a multi-processor system with otherwise shared memory is not visible to the outside and hence is not a failure addressed by these methods. Failure of a NIC card on the other hand requires the use of a different transport address and hence is visible and is to be dealt with. The following types of failures will be visible to signalling neighbours and are targets for this work:

1) Full system component failure (power failure, software crash);
2) Partial system component failure (failure of one out of many communication interfaces);
3) Full network link failure (a system component is no longer reachable); and
4) Partial link network failure (not all system components can reach each other, but some can still communicate; this particularly includes partial connectivity and half-link failure).

It should be noted that various of these failure modes may be not only hard to detect (symmetrically) and may be indistinguishable from one another (see below).

5) Malicious attacks on the system – should be looked at in the context of the H.323 security work.

R.12.4.2 Failure detection

1) Time to detect a failure.
2) Ways of detecting a failure (explicit permanent surveillance vs detection upon invoking a function).
3) Entities responsible for/involved in detecting a failure.
4) Appearance of a failure to a system component/a set of system components.
5) Possibility to determine the type of failure.
6) Consistency/timing of failure detection among various system components.
7) Failure detection may not be transitive, i.e., from "A can/cannot talk to B" and "B can/cannot talk to C" cannot necessarily be concluded that also "A can/cannot talk to C".
8) How much overhead is acceptable?

R.12.4.3 Failure handling

1) Time to repair.
2) Entity initiating the repair process.
3) Possibility to repair the failure.
4) Consequences if the failure cannot be repaired.
5) How to ensure consistent handling of a failure by all involved entities?
6) How to deal with inconsistent views/detection of failures by various components (failed vs not)?
7) How to deal with different timing of failure detection?
8) How to deal with inconsistent state when handling a failure?
9) How to deal with gaps in state information when handling a failure?
10) Consequences on the overall system operation (e.g., an ongoing call).
11) How much overhead is acceptable?
12) How to deal with multiple simultaneous failures?

R.12.4.4 Failure scenarios

This clause lists many identified failure scenarios of H.323 systems. The robustness methods of this annex do not allow recovery from all of these failures but they are listed here for completeness and to give context to the list of failures that are covered by the robustness methods.

1) (Gatekeeper – endpoint): No relationship yet/anymore.
2) (Gatekeeper – endpoint): discovered but not registered.
3) (Gatekeeper – endpoint): discovered and registered.
4) In the process of call establishment:
   a) direct;
   b) gatekeeper-routed.

5) During a call/conference: ITU-T Rec. D.160: "stable state" – discuss what this means for the various protocols:
   a) direct;
   b) gatekeeper-routed.
6) In the process of call teardown:
   a) direct;
   b) gatekeeper-routed.

Consider the implications that arise from the various new protocols under development (H.450.x family, Annex K, Annex L of this Recommendation etc.)

Consider media streams as well as RAS/call signalling/conference control communication relationships.

R.13 Informative Note 2: Call state sharing between an entity and its backup peer

This Note suggests ways to implement call state sharing between an entity and another that serves as its backup peer. The selection of a method is not part of this Recommendation. Since the method is not standardized, peers from different vendors may not be able to serve as robust backup peers.

R.13.1 Shared-memory

If members of the cluster are physically located in the same cabinet, they may be able to use a shared (or reflective) memory device. This is similar to many fault-tolerant platforms, but might simply write to shared memory at each checkpoint rather than running a fault-tolerant OS.

R.13.2 Shared-disk

If the members of the cluster are physically located near each other, they can use a shared-disk and write state information at each checkpoint.

R.13.3 Message passing

The active entity can send a message updating the shared state to each of the other members of the cluster at each checkpoint. This implements a distributed shared memory sometimes referred to as a *bulletin board*. The messages can be sent using distinct UDP messages, multicast messages, persistent TCP links or fault-tolerant message passing protocol such as ASAP (which supports a send-to-group multicast mechanism not requiring multicast IP). This is discussed in more detail and with some suggested checkpoints in APC-1772.

R.13.3.1 SCTP/ASAP

This clause will illustrate, with an H.323 call example, the use of ASAP and SCTP for robustness purposes in an H.323 system. It will give in brief:

1) an architectural overview of an H.323 system using ASAP/SCTP;
2) a view of the necessary protocol stacks in the respective H.323 nodes; and
3) fail-over scenarios of an example H.323 call with two gatekeepers and two endpoints.

R.13.3.1.1 References

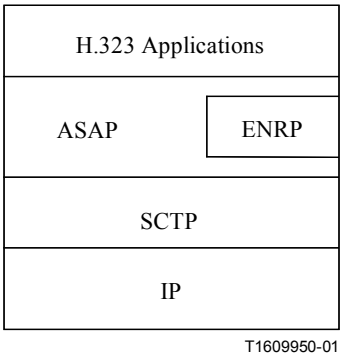
[R.13-1] IETF RFC 2960 (2000), *Stream Control Transmission Protocol*.

[R.13-2] STEWART (R.R.) et al.: *Aggregate Server Access Protocol (ASAP)*, <draft-ietf-serpool-asap-07.txt>, IETF, May 2003.

[R.13-3] XIE (Q.) et al.: *Endpoint Name Resolution Protocol (ENRP)*, <draft-ietf-rserpool-enrp-06.txt>, IETF, May 2003.

R.13.3.1.2 Protocol stacks

In general, an H.323 application using ASAP/SCTP [R.13-1] to [R.13-3] for fault tolerance will have the following protocol stack:



This can provide fast fail-over, transparent to the upper layer application, at both link and session levels:

1) link level (SCTP) – multi-homing support, surviving network failures;
2) session level (ASAP) – server pool support (2N, N+K, etc.), surviving process/node failures.

In addition, ASAP provides:

- location transparency;
- load sharing;
- Plug-n-Play, i.e., hot scalability;
- avoiding single point of failure.

R.13.3.1.3 An architectural view of an H.323 system

Figure R.7 shows an H.323 system built upon ASAP/SCTP model.

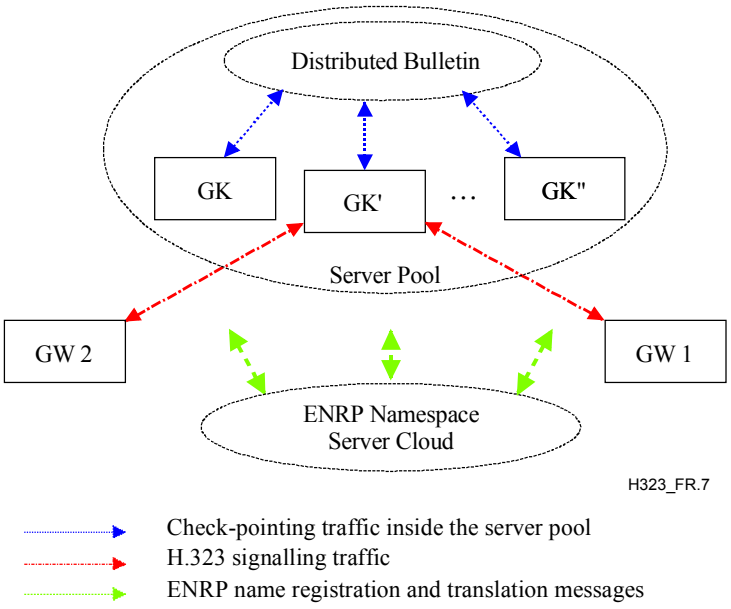


Figure R.7/H.323 – H.323 system built upon ASAP/SCTP model

In the system, all the H.323 components, including the GW 1, GW 2, and GKs employ the ASAP/SCTP stacks as shown in the previous clause. In this example, we assume that the H.323 gatekeeper is implemented as a server pool (the figure depicts the internals of the server pool), while the gateways may or may not be implemented as server pools.

As shown in the figure, inside the gatekeeper server pool we have multiple instances of functionally identical H.323 gatekeepers, GK, GK', … GK". The GK instances share call state and other call recovery critical information among themselves using an internal distributed bulletin board. The mechanism and implementation of the distributed bulletin board is vendor specific and thus out of the scope of either ASAP or SCTP (the bulletin board, however, can use ASAP/SCTP to gain fault tolerance and scalability for itself).

All the ASAP/SCTP nodes, including GWs and GKs, rely on either a single ENRP namespace server cloud or a group of bridged ENRP clouds for name registration and name translation services [R.13-2]. To form the gatekeeper server pool, all GK instances register to the ENRP namespace under the same name. However, each individual GK instance may choose to register with a different load handling capability.

Each H.323 call message will be delivered by ASAP to one of the GK instance in the server pool. The selection of the receiver GK instance is based on both the load sharing policy in effect and the current status of each GK instances in the server pool. It is sometimes very desirable to have all the H.323 signalling messages related to a call be handled by the same GK instance for the entire life cycle of the call, and only let another GK instance take over the call in case the original handler dies. We call this relationship between the call and the server instance "loose binding". ASAP is designed to support this type of "loose binding" relationship very easily [R.13-2] and [R.13-3].

Moreover, when a GK instance is handling a call, it should publish to the distributed bulletin board (i.e., "checkpoint") all critical call state information every time the call reaches a certain stage in its life cycle. This information will help the alternate GK instance to recover the call in case the original call handler crashes.

R.13.3.1.4 An example H.323 call

For the purposes of describing a call, signalling flows are used in Figure R.8.

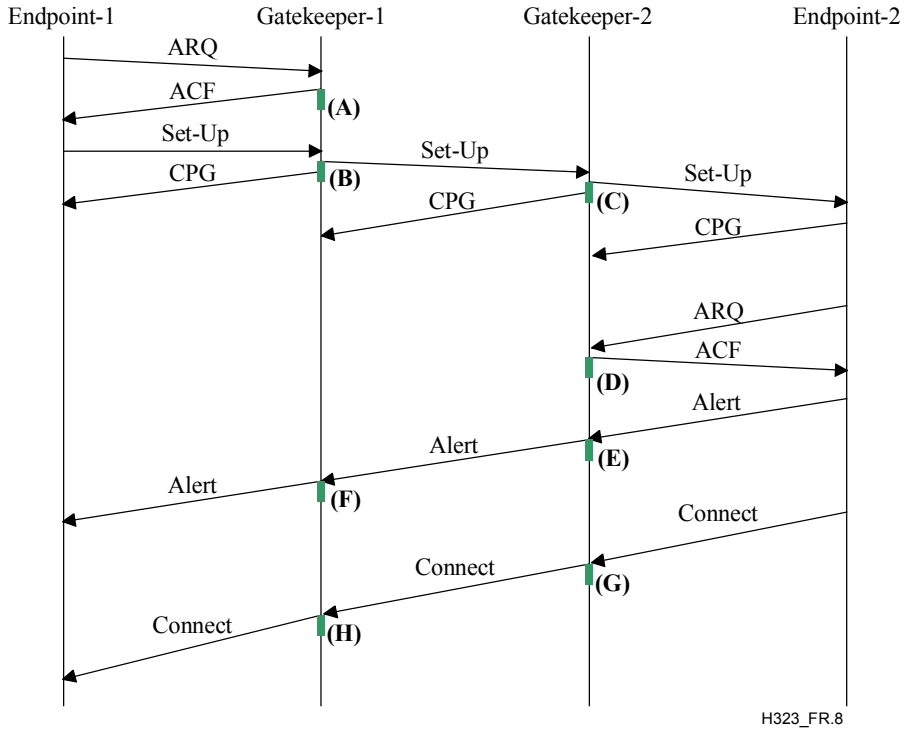


Figure R.8/H.323 – Example H.323 call

Please note that the references in this call flow are quite old and the second gatekeeper is extrapolated. There may be some differences in the way the current H.323 specification would have a call flow, but the point here is to emphasize how ASAP/SCTP would be used. Even if minor items are incorrect in the above figure, this does not invalidate the example.

R.13.3.1.4.1 General description

The call starts with Endpoint-1 requesting bandwidth. The endpoint would in this case use ASAP to query a gatekeeper, known by a name or possibly by a well-known IP address and port. In either case, an ENRP name translation query (not shown) would propagate to the endpoint the set of all gatekeepers (primary and any redundant) in the server pool. This information would be populated into a ASAP layer local cache in Endpoint-1 for future reference in case of a failure. This same caching would occur in all ASAP endpoints in the chain transparent to the call itself. Note that caching is an optional feature. Being that it is an option, endpoints not implementing it can still obtain an alternate gatekeeper, an additional query would be needed to the ENRP server at the time of failure detection.

Note we now hit point (**A**), at this step the gatekeeper allocates bandwidth and checkpoints this bandwidth utilization information message into a "bulletin board" area. This "bulletin board" area could be any of the following:

- a piece of distributed shared memory being maintained by a separate subsystem;
- a piece of reflective memory specifically built for this purpose;
- a distributed commercial database;
- some other creative invention.

Please note that the point here is that the redundant/peer gatekeepers have to share call states in some way. Any existing or future mechanism conceived to share call state can be used.

Gatekeeper-1 populates its ARQ related state and pushes this information to the "bulletin board" and responds to the request in the normal manner, i.e., with an ACF.

Endpoint-1 now reacts and sends the set-up message to Gatekeeper-1. Upon reception of the set-up message Gatekeeper-1 selects the next gatekeeper, Gatekeeper-2, and forwards its set-up, pushing the state information about the call (point (**B**)), possibly tied in some way to the previous information (perhaps with some form of cross-reference, i.e., Call-X is using Y bandwidth represented by the ARQ information). After pushing the information at point (**B**), Gatekeeper-1 sends out the call proceeding message to Endpoint-1.

Gatekeeper-2 receives the set-up message from its peer gatekeeper, selects the destination endpoint forwards the set-up and pushes state information at point (**C**) for the call. After pushing its state to the bulletin board, it sends Gatekeeper-1 a call proceeding message.

Endpoint-2, upon reception of the set-up, sends back a call proceeding message and asks its gatekeeper for bandwidth with its own ARQ message.

This causes Gatekeeper-2 to allocate bandwidth, push state at point (**D**) and send back the ACF message. Upon reception of which, Endpoint-2 sends an Alerting message to Gatekeeper-2.

Upon reception of the Alerting message, Gatekeeper-2 would push a small update to its bulletin board (point (**E**)), i.e., that the call is in Alerting, and forward an Alerting message to Gatekeeper-1.

Gatekeeper-1 will repeat the same procedure, updating its state at point (**F**) and forwarding the Alerting message.

Endpoint-2 at some point answers the call, sending a Connect message to Gatekeeper-2. Gatekeeper-2, upon reception of the Connect message, will push another small update to the state at point (**G**) indicating that the call is now in an answered state and forward the connect message to Gatekeeper-1.

Upon reception of the Connect message, Gatekeeper-1 will perform the same operation, saving its state at point (**H**) and sending the connect message on to Endpoint-1.

R.13.3.1.4.2 Failure scenarios

The above descriptions assume the maximum level of redundancy and state/call preservation. In this scenario any failure of either Gatekeeper becomes transparent to either endpoint. If a failure occurs, the message would be re-routed by ASAP to an alternate. The alternate would need to take the following actions on any message it received that it did not have a call object/block for:

- look up the call in the "bulletin board";
- pull the state information and construct a call control block or object to the call;
- continue processing the message on behalf of the dead peer.

Endpoints become completely transparent to failure scenarios. No knowledge is placed in the endpoint itself (other than ASAP) to recover from a Gatekeeper failure.

R.13.3.1.4.3 State saving issues

As stated above, the example assumes a maximum state saving model. In this mode updates to state would need to be minimized to the smallest amount of information possible. In particular, state should be limited to the smallest set of information necessary to re-construct the call AND updates should be as small as possible. In some cases an operator may not wish to have this level of redundancy. To achieve a robust system with less state, the following state sharing points could be eliminated:

- At points (**A**) and (**D**) – If the gatekeeper uses some other methodology to calculate bandwidth utilization (besides tracking the number of calls by count) these steps could be completely skipped with no harm. It may be that the operator has NO concern for admission control and its gatekeepers do not perform this, in these cases this step is not necessary.
- At points (**F**) and (**E**) – These points are optional in that they may not provide any information worth saving, i.e., the call is ringing versus still setting up.
- At points (**B**) and (**C**) – If the operator is NOT interested in saving anything but stable calls, these points can be eliminated. In this case, any calls that were being set up would be lost if a failure did occur.

Trade-offs, such as the above, are outside the scope of using ASAP/SCTP and are strictly an operator/manufacture decision as to how much state may be saved by a given implementation and what controls/options the operator may have.

Appendix I

Sample MC to terminal communication mode command

I.1 Sample conference Scenario A

Endpoints A, B and C are in an audio and video distributed conference using multicast. The MC (which could be anyone of the nodes) has decided to place the media and media control channels on the following multicast addresses:

Stream	Multicast address
Audio for all endpoints	MCA1
Audio Control for all endpoints	MCA2
Video from endpoint A	MCA3
Video Control data about endpoint A	MCA4
Video from endpoint B	MCA5
Video Control data about endpoint B	MCA6
Video from endpoint C	MCA7
Video Control data about endpoint C	MCA8

I.2 CommunicationModeTable sent to all endpoints

All entries are commands for endpoints to open a logical channel for transmission. **terminalLabel** is only present when the entry is specific to a single endpoint in the conference.

```
ENTRY 1 – AUDIO & AUDIO CONTROL FOR CONFERENCE
sessionID                       1
sessionDescription              Audio
dataType                        Audio Capability
mediaChannel                    MCA1
mediaControlChannel             MCA2

ENTRY 2 – VIDEO & VIDEO CONTROL FOR NODE A
sessionID                       2
associatedSessionID             1
terminalLabel                   M/T for A
sessionDescription              Video for Node A
dataType                        Video Capability
mediaChannel                    MCA3
mediaControlChannel             MCA4

ENTRY 3 – VIDEO & VIDEO CONTROL FOR NODE B
sessionID                       3
associatedSessionID             1
terminalLabel                   M/T for B
sessionDescription              Video for Node B
dataType                        Video Capability
mediaChannel                    MCA5
mediaControlChannel             MCA6

ENTRY 4 – VIDEO & VIDEO CONTROL FOR NODE C
sessionID                       4
associatedSessionID             1
terminalLabel                   M/T for C
sessionDescription              Video for Node C
dataType                        Video Capability
mediaChannel                    MCA7
mediaControlChannel             MCA8
```

I.3 Sample conference Scenario B

Endpoints A, B and C are in a multipoint conference where audio is unicast from each endpoint and centrally mixed, but video is multicast from the endpoints. The MC may send a unique CommunicationModeCommand to each endpoint, or it may send the same message to all endpoints if the table entries are identified by the destination endpoint's label. For this example, assume that the same message is sent to all endpoints.

Stream	Multicast Address
Audio from endpoint A	UCA1
Audio Control data about endpoint A	UCA2
Audio from endpoint B	UCA3
Audio Control data about endpoint B	UCA4
Audio from endpoint C	UCA5
Audio Control data about endpoint C	UCA6
Video from endpoint A	MCA1
Video Control data about endpoint A	MCA2
Video from endpoint B	MCA3
Video Control data about endpoint B	MCA4
Video from endpoint C	MCA5
Video Control data about endpoint C	MCA6

I.4 CommunicationModeTable sent to all endpoints

All entries are commands for endpoints to open a logical channel for transmission. **terminalLabel** is only present when the entry is specific to a single endpoint in the conference.

```
ENTRY 1 – AUDIO & AUDIO CONTROL FOR NODE A
sessionID                       1
sessionDescription              Audio
terminalLabel                   M/T for A
dataType                        Audio Capability
mediaChannel                    UCA1
mediaControlChannel             UCA2

ENTRY 2 – AUDIO & AUDIO CONTROL FOR NODE B
sessionID                       2
sessionDescription              Audio
terminalLabel                   M/T for B
dataType                        Audio Capability
mediaChannel                    UCA3
mediaControlChannel             UCA4

ENTRY 3 – AUDIO & AUDIO CONTROL FOR NODE C
sessionID                       3
sessionDescription              Audio
terminalLabel                   M/T for C
dataType                        Audio Capability
mediaChannel                    UCA5
mediaControlChannel             UCA6

ENTRY 4 – VIDEO & VIDEO CONTROL FOR NODE A
sessionID                       4
associatedSessionID             1
terminalLabel                   M/T for A
sessionDescription              Video for Node A
dataType                        Video Capability
mediaChannel                    MCA1
mediaControlChannel             MCA2
```

```
ENTRY 5 – VIDEO & VIDEO CONTROL FOR NODE B
sessionID                       5
associatedSessionID             2
terminalLabel                   M/T for B
sessionDescription              Video for Node B
dataType                        Video Capability
mediaChannel                    MCA3
mediaControlChannel             MCA4

ENTRY 6 – VIDEO & VIDEO CONTROL FOR NODE C
sessionID                       6
associatedSessionID             3
terminalLabel                   M/T for C
sessionDescription              Video for Node C
dataType                        Video Capability
mediaChannel                    MCA5
mediaControlChannel             MCA6
```

Appendix II

Transport level resource reservation procedures

II.1 Introduction

H.323 recommends the use of transport level resource reservation mechanisms to fulfil the QOS requirements of real-time video and audio streams. Although the transport level resource reservation mechanisms themselves are beyond the scope of this Recommendation, the general method and coordination of these transport level mechanisms between H.323 entities is described in this appendix to prevent conflicting interoperability issues.

This appendix describes the use of RSVP (Resource reSerVation Protocol) as a possible mechanism for providing transport level QOS over IP-based networks. Other protocols may be used; however, the basic procedures defined in this appendix should still apply. Participants in a conference should be able to signal their intentions, capabilities, and requirements in a standard, protocol-specific manner. In addition, the signalling sequence of the resource reservation mechanisms must be specified such that the call establishment interval is minimal.

RSVP is the transport level signalling protocol for reserving resources in unreliable IP-based networks. Using RSVP, H.323 endpoints can reserve resources for a given real-time traffic stream based on its QOS requirements. Only best-effort delivery of the packets is possible if the network fails to reserve the required resources or if RSVP is absent.

II.2 QOS support for H.323

When an endpoint requests admission with a Gatekeeper, it should indicate in the ARQ message whether or not it is capable of reserving resources. The Gatekeeper should then decide, based on the information it receives from the endpoint and on information it has about the state of the network, either:

- to permit the endpoint to apply its own reservation mechanism for its H.323 session; or
- to perform resource reservation on behalf of the endpoint; or
- that no resource reservation is needed at all. Best-effort is sufficient.

This decision is conveyed to the endpoint in the ACF message. The endpoint shall accept the Gatekeeper's decision in order to place a call.

The Gatekeeper should reject an endpoint's ARQ, if the endpoint does not indicate that it is capable of resource reservation, and the Gatekeeper decides that resource reservation must be controlled by the endpoint. In this case, the Gatekeeper should send an ARJ back to the endpoint.

The specific field in H.225.0 RAS signalling to permit this functionality is the **transportQOS** field.

In addition to **transportQOS**, an endpoint should also calculate and report the bandwidth it currently intends to use in all channels of the call. This bandwidth should be reported in the **bandWidth** field of the ARQ message independent of the decision by the endpoint to use RSVP signalling or not. In addition, if bandwidth requirements change during the course of the call, an endpoint should report changes in bandwidth requirements to the Gatekeeper using BRQ independent of the decision to use RSVP.

RSVP reservations can only be made by network entities which are in the path of media flow between endpoints. It is possible through Gatekeeper routed call signalling to route media streams through a Gatekeeper. However, most of the time media channels will be routed between endpoints without passing through the Gatekeeper. If a Gatekeeper decides to route media streams, then the procedures followed should be identical to those for RSVP signalling directly from the endpoints. It is best if RSVP reservations are made directly by the endpoints since this will reserve resources along the entire routed path of the call. The remainder of this appendix discusses the use of RSVP by the H.323 endpoints.

Some of the salient points of RSVP are as follows:

- RSVP supports both unicast and multicast environments;
- RSVP is tied to specific streams (i.e., specific Transport Address pairs);
- RSVP is soft-state based, and therefore adapts dynamically to changing group membership and routes;
- RSVP is unidirectional;
- RSVP is receiver-oriented – the recipient of the media stream makes the reservation (scalable).

II.3 RSVP background

In the following description, the high-level usage of RSVP in a simple H.323 conference will be outlined.

In Figure II.1, Endpoint A wishes to send a media stream to Endpoint B. Therefore, it has to open a logical channel to B. RSVP signalling for resource reservation should be a part of the opening logical channel procedure. Endpoint A would cause RSVP *Path* messages to be sent out to B. These *Path* messages go through routers and leave "state" on their way tracing towards B. *Path* messages contain the complete source and destination addresses of the stream and a characterization of the traffic that the source will send. Endpoint B would use the information from the *Path* to make the RSVP *Resv* request for the full length of the path. *Resv* messages contain the actual reservation and will generally be the same as the traffic specification in the *Path* message.

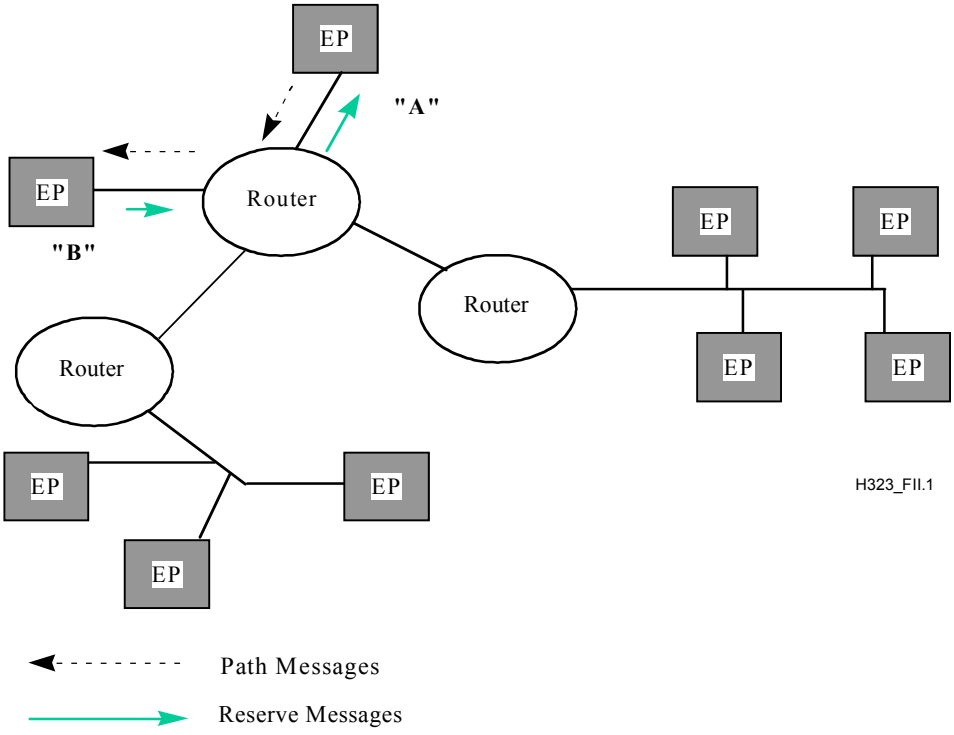


Figure II.1/H.323 – Resource reservation for a point-to-point connection

In Figure II.2, a multipoint conference is shown. The *Path* messages are utilized in the same manner as the simpler point-to-point case. It should be noted that the *Resv* requests are aggregated by the routers to keep redundant reservation requests from occurring upstream.

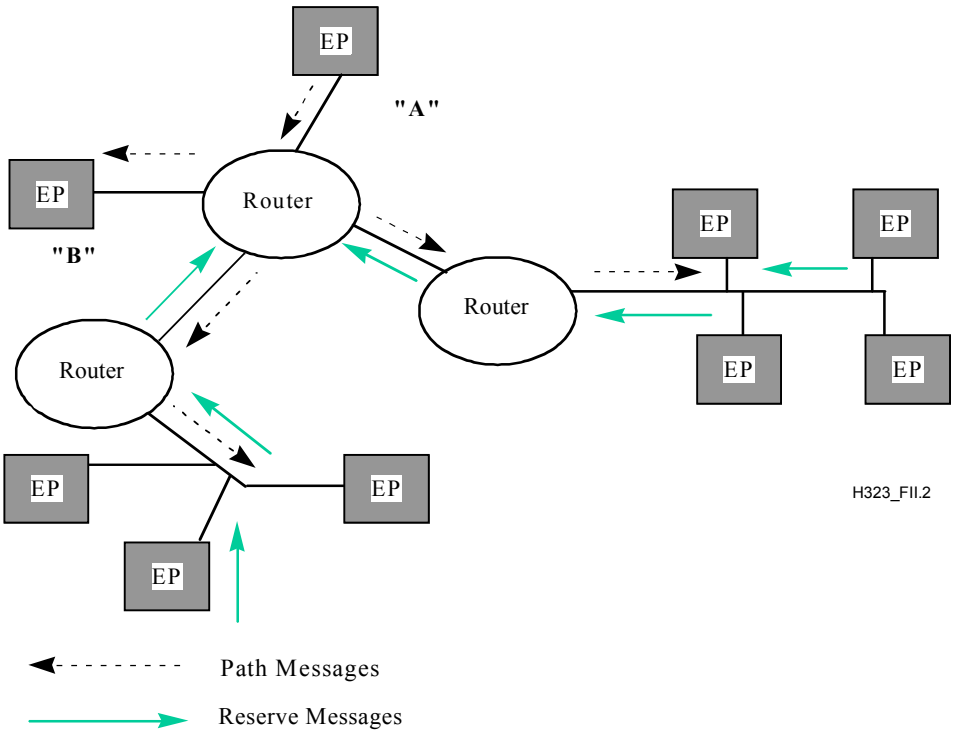


Figure II.2/H.323 – Resource reservation for a point-to-multipoint connection

Path messages must contain the complete destination/source addresses and a traffic specification. *Resv* messages contain the reservation parameters and the required service. *Path* and *Resv* messages for a given traffic stream should be sent as part of the **openLogicalChannel** procedure for that particular stream. The reservation should be released during the **closeLogicalChannel** procedure using the RSVP *PathTear* and *ResvTear* messages.

Note that RSVP Path and Resv messages use the same IP address/port pair as the media to be delivered between endpoints. This means that these messages must be filtered out of the media stream by the endpoints. This is not an issue for endpoints which do UDP filtering since RSVP messages themselves are not UDP messages. Even so, the sender of a media stream should not use RSVP when the receiver is not capable of it. RSVP capabilities are exchanged as part of the capability exchange and open logical channel procedures.

RSVP is only a signalling protocol. Together with the appropriate QOS services (e.g., guaranteed QOS or controlled-load service), scheduling mechanisms (e.g., weighted fair queuing), and policy-based admission control module (e.g., local policy manager), RSVP is capable of satisfying the QOS requirements of H.323 conference participants. In addition, RSVP is designed for point-to-point links. If a path traverses a shared link, RSVP invokes the appropriate resource reservation mechanism for the specific shared medium, e.g., SBM (Subnet Bandwidth Management) in case of Ethernet. All the mechanisms mentioned in this paragraph are controlled completely from within RSVP. Therefore, all that an H.323 endpoint needs is RSVP signalling.

II.4 The H.245 capability exchange phase

During the H.245 capability exchange phase, each endpoint indicates its transmit and receive capabilities to the other endpoint. The **qOSCapability** is part of the capability exchange. However, it is not stream-specific. Therefore, the RSVP parameters if specified in the **qOSCapability** would represent an aggregate for all streams (either those to be transmitted or those to be received). Such parameters will not be of any use to the other endpoint. Therefore, the only RSVP-related information an endpoint should convey to the other endpoint in the capability set is whether or not it is RSVP-capable.

To signal RSVP capability, an endpoint shall set the appropriate available **qOSMode** fields within the capability PDU during capability exchange. Endpoints which do not receive RSVP capabilities from the receiving endpoint shall not use RSVP when opening logical channels.

II.5 Open logical channel and setting up reservations

In this clause, we describe the steps that should be followed for opening an H.245 logical channel and reserving resources for a given traffic stream. Reservations are established only if both endpoints indicate that they are RSVP enabled during capability exchange. We consider only the point-to-point case. The case of point-to-multipoint (multicast) connections will be discussed in II.7.

The sender shall specify the RSVP parameters of the stream to be transmitted and the integrated services the sender supports in the **qOSCapability** field of the **openLogicalChannel** message. In case of a point-to-point stream, the sender does not specify a receiver port ID in the **openLogicalChannel** message. This ID is selected by the receiver after receiving the **openLogicalChannel** and is returned to the sender in the **openLogicalChannelAck** message. Only then can the sender create an RSVP session for that stream (to create an RSVP session for a given stream means that the endpoint registers with RSVP to get notified when messages arrive that may affect the state of the RSVP reservation for that stream) and start emitting RSVP *Path* messages. The receiver has sufficient information to create an RSVP session for the same stream before sending the **openLogicalChannelAck** message. The information needed to create an RSVP session and initiate RSVP processing are: the receiver IP address in case of point-to-point or the group

multicast IP address in case of point-to-multipoint, the receiver port ID, and the protocol (always UDP in case of H.323 audio and video streams on IP networks).

A receiver may not want to start receiving stream packets until the RSVP reservations are in place. To achieve this, the receiver may set the Boolean **flowcontrolToZero** field of the **openLogicalChannelAck** message to TRUE to indicate that it does not wish to receive any traffic on that channel before the resource reservations are complete. When a sender receives an **openLogicalChannelAck** message with **flowControlToZero** set to TRUE, the sender shall not transmit any traffic on that channel.

When the receiver starts receiving the sender's *Path* messages, it should start sending RSVP *Resv* messages. When the receiver receives an RSVP *ResvConf* message confirming that reservations have been established, it may send a **flowControlCommand** to the sender unrestricting the bit rate of the traffic stream, i.e., cancelling the effect of the previous **flowcontrolToZero** field in the **openLogicalChannelAck** message. When the sender receives the **flowControlCommand** it starts transmitting packets.

Note that the *ResvConf* message and similarly all other RSVP messages are transmitted unreliably. As a result, they may get delayed or even lost. An endpoint should be aware of that fact and set timers with appropriate value while waiting for a *ResvConf*. The action taken if the endpoint times out without receiving a *ResvConf* is up to the individual endpoint vendors.

The behaviour of an endpoint if RSVP reservations fail at any point during an H.323 call is not specified in this appendix and is left to the individual vendors. However, if an RSVP reservation fails and the receiving endpoint decides that best-effort level of service is not acceptable, it may request to close its logical channel using the **requestChannelClose** message. The **closeReason** field is available in the **requestChannelClose** message to allow the receiver to signal to the sender that the RSVP reservation has failed. Along with the failure indication, **requestChannelClose** includes **qOSCapability** which can be used by the receiver to tell the sender the resources which are actually currently available on the path from the sender to the receiver. At this point, the sender can decide to try to reopen the channel with a lower bandwidth codec and/or data format and go through the Open Logical Channel procedure again.

All RSVP *Resv* requests shall use the same reservation style, the **Fixed Filter** style, for the following reasons:

- Shared filter styles reduce to fixed filters in case of point-to-point calls.
- Different reservation styles for the same session cannot be merged in the network. For example, if in a multipoint call some of the receivers request fixed filter reservations while the rest request shared explicit reservations, then either the fixed filter reservations or the shared explicit reservations will fail.
- Shared reservations, created by wildcard filter and shared explicit filter styles, are appropriate for those multicast applications in which multiple data sources are unlikely to transmit simultaneously. In distributed multipoint H.323 calls, there is no mechanism to permit only one source to transmit at a specific time. On the other hand, in centralized multipoint H.323 calls, the MCU is the only multicast source. Shared reservation styles are not suited for either case.

It is up to the endpoint vendors to choose which intserv QOS service (guaranteed QOS or controlled-load) to use. However, any RSVP-enabled H.323 endpoint shall support the controlled-load service as a least common service. This requirement is necessary to avoid interoperability problems that may arise from RSVP-enabled H.323 endpoints which do not support a common intserv QOS service.

Figure II.3 shows the sequence of messages in case of successful RSVP reservation.

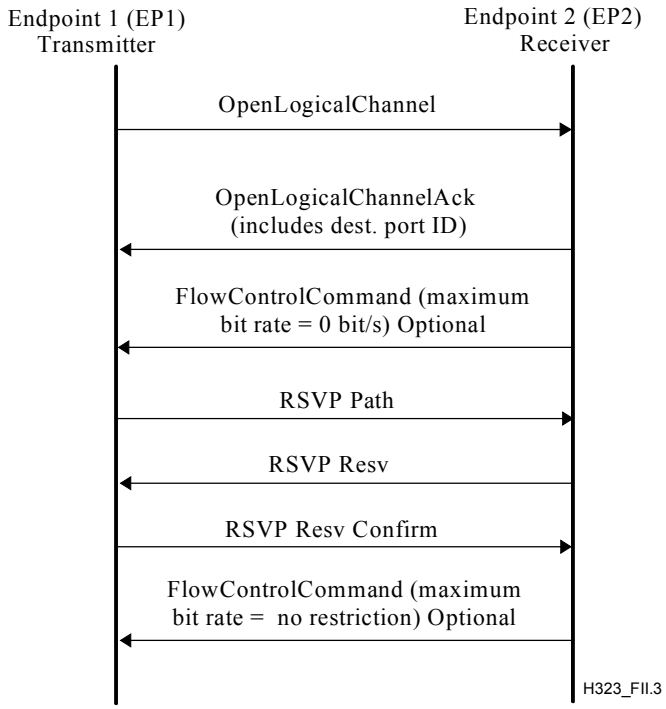


Figure II.3/H.323 – Message sequence for opening a unicast logical channel with RSVP

II.6 Close logical channel and tearing down reservations

Before sending out a **closeLogicalChannel** message for a given traffic stream, a sending endpoint should send a *PathTear* message if an RSVP session has been previously created for that stream. When a receiving endpoint receives a **closeLogicalChannel** for a given traffic stream, it should send a *ResvTear* message if an RSVP session has been previously created for that stream.

II.7 Resource reservation for multicast H.323 logical channels

The H.245 **openLogicalChannel** procedure is point-to-point even if the traffic stream involved is a multicast stream. However for the receiving endpoint to start receiving packets of a multicast stream, it has to join the multicast group and get connected to the source's multicast tree. When a receiver receives an **openLogicalChannel** message, it joins the multicast group and the source's multicast tree using standard IGMP procedures. The IGMP join (using IGMP *Report* message) takes place before the receiver sends an **openLogicalChannelAck** back to the sender.

In case of a multicast stream, the sender specifies the receiver port ID in the **openLogicalChannel** message instead of receiving the receiver port ID in the **openLogicalChannelAck** message.

The receiver may set the **flowControlToZero** field of the **openLogicalChannelAck** message to TRUE, similar to the unicast case. However, the sender (an endpoint in a distributed conference or an MCU in centralized conference) should decide not to interrupt the data stream on the opened channel, if it determines this interruption may affect other receivers of the same multicast group which are already receiving that stream. As a result, in the multicast case, the receiver may initially receive the data at best-effort until the RSVP reservations are established.

Figure II.4 shows the sequence of messages required to open a logical channel and to join the multicast tree and to reserve resources for a multicast stream.

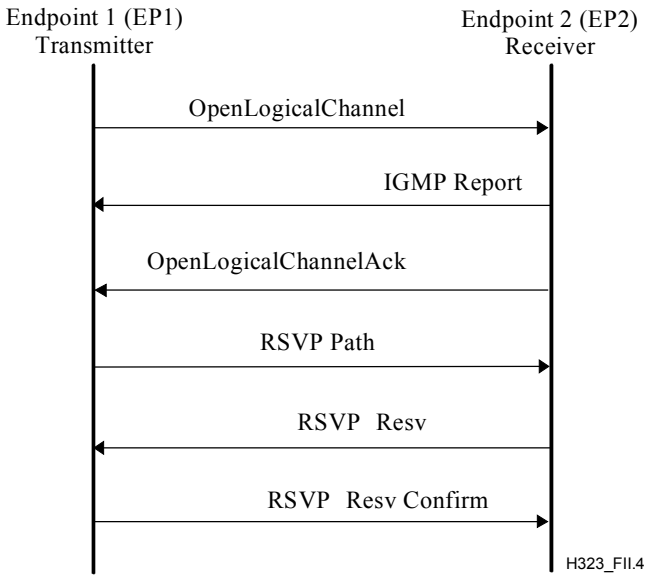


Figure II.4/H.323 – Message sequence for opening a multicast logical channel with RSVP

Before sending out a **closeLogicalChannel** message for a given multicast stream, a sending endpoint should send an RSVP *PathTear* message if the logical channel being closed is the last channel carrying that multicast stream and if an RSVP session has been previously created for that stream. When a receiving endpoint receives a **closeLogicalChannel** for a given multicast stream, it should send an RSVP *ResvTear* message and an IGMP *Leave* message, if an RSVP session has been previously created for that stream.

II.8 Synchronized RSVP

Synchronized RSVP is defined as the process of reserving resources with RSVP prior to transitioning to the Alerting phase of the call. Details of synchronizing RSVP without Fast Connect and with Fast Connect, respectively, are discussed in the following two subclauses. This clause introduces the general concept of a prioritized list of QOS levels, expressed by each endpoint from which a new set of QOS levels 'D' is derived. This derived set 'D' comprises the intersection of the two preferred **QOSMode** sets. The two endpoints can attempt to establish RSVP reservations based on a QOS level in the derived set starting with the most preferred QOS level.

Upon deriving the QOS set, the called endpoint suppresses the Alerting phase of the call until reservations are established in both directions. On successful reservation establishment, the Alerting can proceed, and call setup is resumed. In the event of failures, the lowest QOS level in the derived set is examined. If this is indicated to be "best effort", the call setup procedures are resumed; otherwise, the call is released. Sending a **QoSCapability** structure with an empty **QOSMode** element in the **rsvpParameters** block shall indicate a "best effort" level of QOS. The **QOSMode** sequence is prioritized by the **QOSMode** element of the **rsvpParameters** block with the priority decreasing from the first element to the last. **GuaranteedQoS** is the highest level of QOS that an endpoint can receive, and "best effort" is the lowest. If the preferred QOS that the calling endpoint wishes to receive is higher than "best effort", the endpoint should start RSVP procedures by listening for PATH messages from the called endpoint.

The called endpoint shall examine the sequence of **QoSCapability** structures, if present, and compare it to its own preferred set of QOS levels based on **QOSMode**. It then derives a new set of QOS levels 'D' based on **QOSMode** that represents the intersection of QOS levels from the preferred sets of the two endpoints. This new set denotes the different QOS levels in a prioritized order based on **QOSMode** that are supported by both endpoints. For example, if the calling endpoint's preferred set of QOS levels is {**GuaranteedQOS**, **ControlledLoad**} and that of the

called endpoint is {**ControlledLoad**, "best effort"}, the derived set representing the intersection is {**ControlledLoad**}. Based on the preferred QOS levels of the two endpoints, different general cases are possible. The different cases and the corresponding call handling are shown in the Table II.1.

Table II.1/H.323 – Handling calls for various QOS classes

QOS Scenario	Example	Call handling
1) The Derived QOS Set 'D' is empty	Preferred set of Calling Endpoint : {GQ} Preferred set of Called Endpoint : {CL,BE} Derived QOS Set 'D' : {}	The called endpoint shall release the call.
2) The Derived QOS Set 'D' has just one QOS level: "best effort"	Preferred set of Calling Endpoint : {BE} Preferred set of Called Endpoint : {CL,BE} Derived QOS Set 'D' : {BE}	The called endpoint shall not attempt RSVP procedures. It shall, however, continue with call setup procedures.
3) The Derived QOS Set 'D' has at least one QOS level higher than "best effort"	Preferred set of Calling Endpoint : {GQ,CL,BE} Preferred set of Called Endpoint : {CL,BE} Derived QOS Set 'D' : {CL,BE}	The called endpoint shall suppress Alerting and attempt Synchronized RSVP. The detailed procedures are described in the individual subclauses below.
BE "Best Effort" CL ControlledLoad GQ GuaranteedQoS		

In the event of failure in RSVP procedures, the called endpoint shall examine the next most preferred QOS, if present, in the derived set 'D'. If a QOS level other than "best effort" exists, the called endpoint should reinitiate RSVP reservations with that QOS level. In the event of successive failures, it is possible to reattempt RSVP reservation procedures for all QOS levels (other than "best effort") in the derived set. On expiry of the reservation timer on the called endpoint or, if the called endpoint fails to establish RSVP reservations with the lowest non-"best effort" level of QOS in the derived set, the called endpoint shall examine the lowest level of QOS in the derived set. If this QOS level is not "best effort", the called endpoint shall release the call; otherwise, the call setup is resumed with a QOS level of "best effort". Reservation failures and expiry of the reservation timer are handled similarly on the calling endpoint.

The following two subclauses discuss Synchronized RSVP and Synchronized RSVP with Fast Connect, respectively, using the concept of the prioritized **QOSMode** derived list.

II.8.1 Synchronizing RSVP when not using Fast Connect

A calling endpoint that wishes to reserve resources via synchronized RSVP when not placing a Fast Connect call shall, as a prerequisite, include an H.245 address in the Setup message. Likewise, a called endpoint that wishes to reserve RSVP resources prior to call setup completion shall retrieve the calling endpoint's H.245 address, if present, from the incoming Setup message. Subsequently, the called endpoint shall establish the H.245 Control Channel and commence H.245 procedures. Until H.245 and RSVP procedures have completed, the called endpoint shall not continue with the H.225.0 call setup phase. It is recommended, however, that the called endpoint return a Call Proceeding message to the calling endpoint to prevent any H.225.0 timer on the originating side from expiring.

If the called endpoint desires to attempt synchronized RSVP, yet the calling endpoint does not include its H.245 address in the incoming Setup message, then the calling endpoint shall assume that the originating endpoint will not accept or initiate synchronized RSVP procedures. It is then the responsibility of the called endpoint to decide on the appropriate action to take, based on the derived QOS mode as discussed in II.8. Similarly, if the calling endpoint desires to attempt synchronized RSVP and has included its H.245 address in the Setup message, yet the called endpoint has failed to establish the H.245 Control Channel and has resumed with H.225.0 procedures, then it is up to the calling endpoint to determine which action to take, based on the derived QOS mode as shown in Table II.1.

Otherwise, if the calling endpoint has offered its H.245 address in the Setup message and the called endpoint has established the H.245 Control Channel, H.245 procedures will progress as usual through master-slave determination and capability exchange.

During the H.245 capability exchange, endpoints wishing to attempt RSVP are required to include a sequence of **qOSCapabilities** (as part of the **transportCapability** element of the **H2250Capability** structure), prioritized by the **qosMode** (e.g., **guaranteedQOS**, **controlledLoad**) element of the **rsvpParameters**.

Likewise, when opening logical channels using H.245, each endpoint shall specify the RSVP parameters of the stream to be transmitted in the **qOSCapability** field of the **openLogicalChannel** message.

Upon receiving an OLC message from its peer and under the condition that the peer has indicated during capability exchange that it is RSVP enabled, the endpoint shall start listening for incoming Path messages. When it receives a Path message, the endpoint shall respond by sending a Resv message along the receive stream.

Upon receiving an OLC ACK message from its peer, the endpoint shall start sending Path messages to its peer along its transmit stream. RSVP procedures have successfully completed when the endpoint has received a Resv Confirm in response to its Resv message transmission and a Resv message in response to its Path message. If multiple streams are involved (e.g., voice, video, and data), then the endpoint must wait for reservation confirmation for all streams requiring RSVP-based QOS.

It is recommended that the endpoint start a timer for a short amount of time (e.g., five or six seconds), once it has attempted RSVP. If the timer expires before the RSVP reservations have completed, then the endpoint can determine appropriate action to take.

In the case that RSVP procedures (and therefore H.245 procedures) have successfully completed before the timer expires, the called endpoint may then resume normal call setup procedures by returning an Alerting message to the calling endpoint. If, however, the attempt to reserve RSVP resources fails, then it is the individual endpoint's responsibility to decide on appropriate action to take, based on the derived **QOSMode** set, as described in II.8. In any case, it is recommended that if the call has reached the Alerting phase of the call and RSVP reservations have failed, then the call is allowed to proceed.

Figure II.5 illustrates the modified call flow for a successful synchronized RSVP when not using Fast Connect.

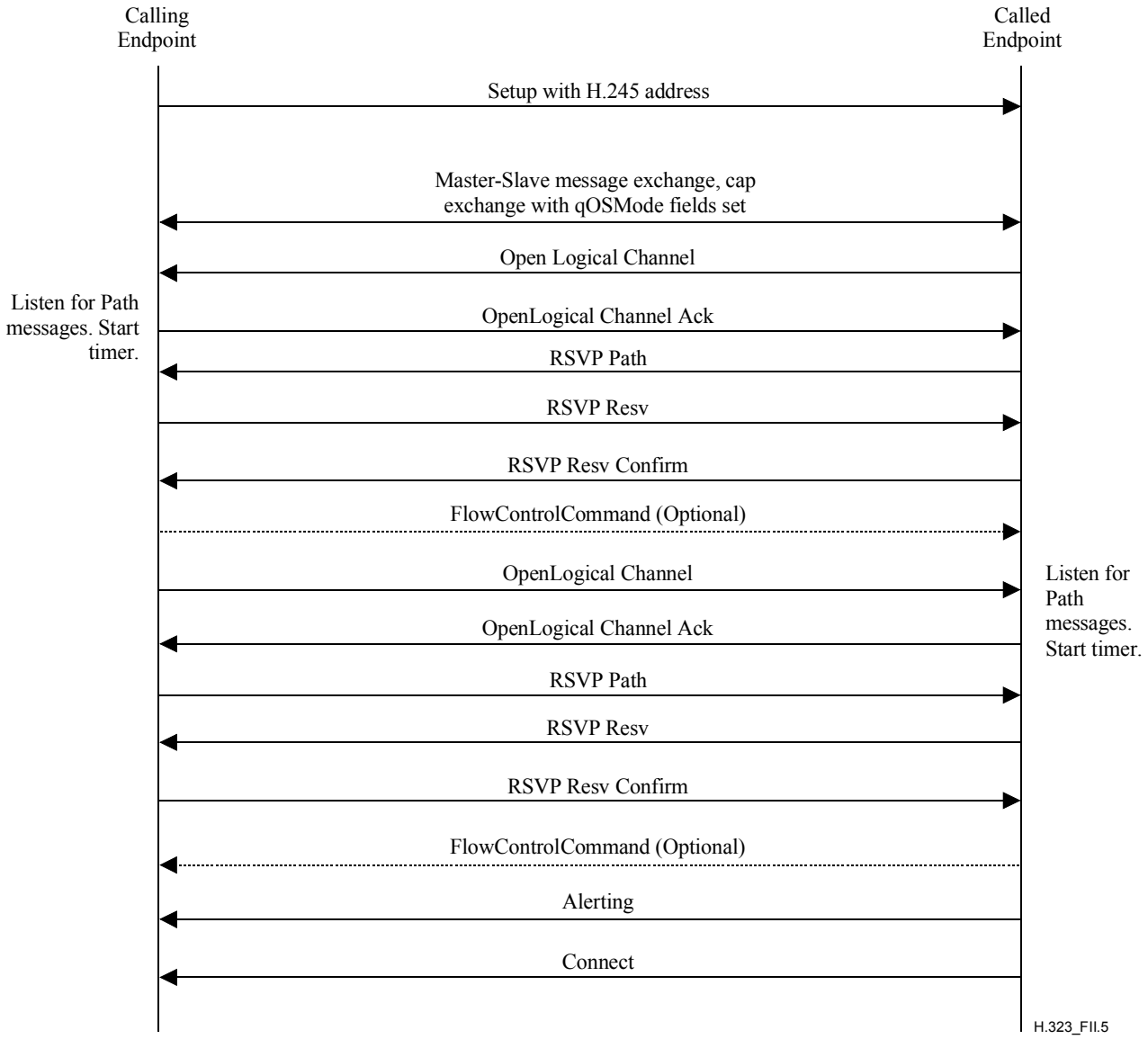


Figure II.5/H.323 – Synchronizing RSVP when not using Fast Connect

II.8.2 Synchronizing RSVP with fast connect

This clause describes synchronizing Fast Connect call setup procedures with RSVP reservation procedures in order to eliminate transporting in-band ringing before the reservations have been established.

A calling endpoint that wishes to use RSVP in a Fast Connect procedure shall send a sequence of prioritized **QoSCapability** structures in the **OpenLogicalChannel** structures contained in the **fastStart** element of the Setup message.

Upon receiving the Fast Connect Setup message, the called endpoint shall derive the **QOSMode** set using the mechanism described in Table II.1. Assuming that the derived set contains a valid (i.e., non-best effort intersection), the called endpoint shall respond to the Setup message from the calling endpoint by sending a **fastStart** element including only the **QoSCapabilities** indicated in the derived QoS set. The **fastStart** element shall be sent as soon as possible (e.g., in a Call Proceeding message) to expedite the resource reservation. The calling endpoint's set will be a subset of the list sent by the calling endpoint in the **OpenLogicalChannel** structures and will, similarly, be a sequence in decreasing order of priority by **QOSMode**. Each **QoSCapability** included in the **OpenLogicalChannel** in the response message indicates an acceptance of the corresponding

QOS level by the called endpoint. The **OpenLogicalChannel** structures in the **fastStart** element also contain information about the media ports used on the called endpoint.

The called endpoint shall initiate RSVP procedures by sending a PATH message to its peer along the transmit stream. In addition, the endpoint may use a reservation timer which would represent the total time available to establish synchronized RSVP reservations for any QOS level (other than "best effort") in the derived set. Furthermore, the called endpoint shall respond to an incoming PATH message with a RESV message along the receive stream. Note that the called end should suppress the Alerting phase of the call and not send an Alerting message to the calling endpoint until reservations are established in both directions. After RSVP procedures are established, the called endpoint shall continue with the H.225 call setup procedures.

When the calling endpoint receives the **fastStart** element, it shall extract the media port information in the **OpenLogicalChannel** and also record the prioritized list of **QoSCapabilities** returned by the called endpoint. The endpoint shall start sending PATH messages to its peer along the transmit stream. Also, when it receives a PATH message from the called endpoint, it shall respond with a RESV message along the receive stream. The calling endpoint may start a reservation timer that would represent the total time available to establish synchronized RSVP reservations.

The establishment of RSVP reservations is said to have successfully completed when the called endpoint receives a RESV message in response to its PATH message and a RESV CONFIRM message in response to its RESV message. As soon as the RSVP procedures are completed successfully, the called endpoint shall stop the reservation timer and resume with the call setup procedures. It subsequently sends Alerting/Connect messages to the calling endpoint. Figure II.6 illustrates the call flow for a successful synchronized Fast Connect call.

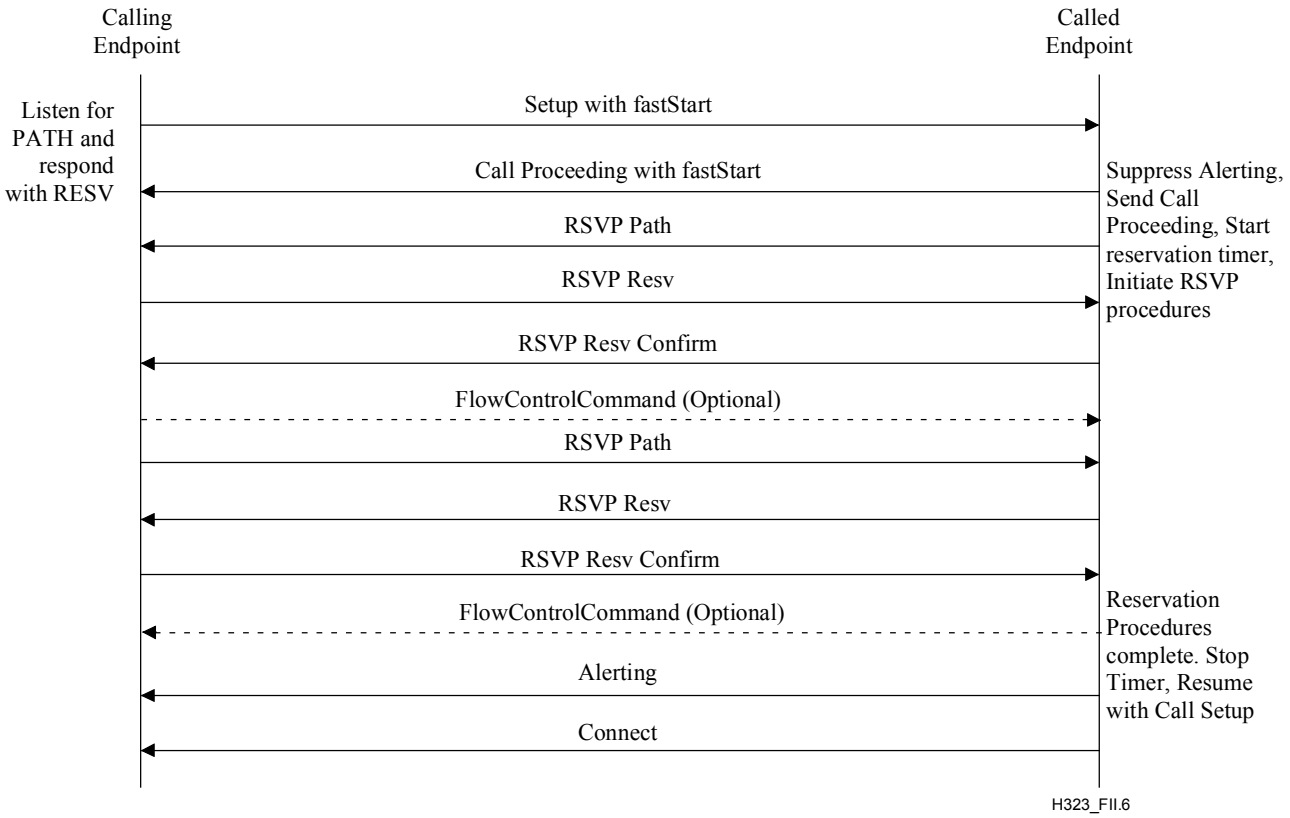


Figure II.6/H.323 – Synchronizing RSVP when using Fast Connect

In the event of RSVP failure, the called endpoint will take action according to the derived **QOSMode** set, as described in II.8.

Appendix III

Gatekeeper-based user location

III.1 Introduction

This appendix gives examples of how a Gatekeeper/proxy can implement user location services. These services depend on the Gatekeeper using the Gatekeeper routed call signalling model.

III.2 Signalling

In the scenario shown in Figure III.1, the Gatekeeper implements a "divert on no reply" service. Endpoint 1 calls Endpoint 2 with the Call Signalling Channel routed through the Gatekeeper. If there is no answer after some timeout, the Gatekeeper diverts the call to an alternate endpoint. Messages (1) to (5) show the Gatekeeper attempting to establish a call between Endpoint 1 and Endpoint 2. In this example, Endpoint 2 does not answer and so the Gatekeeper clears the call to Endpoint 2 by sending Release Complete (6). The Gatekeeper then tries Endpoint 3 by sending Setup (7). When Endpoint 3 answers the call using Connect (9), the Gatekeeper forwards the Connect (10) back to Endpoint 1.

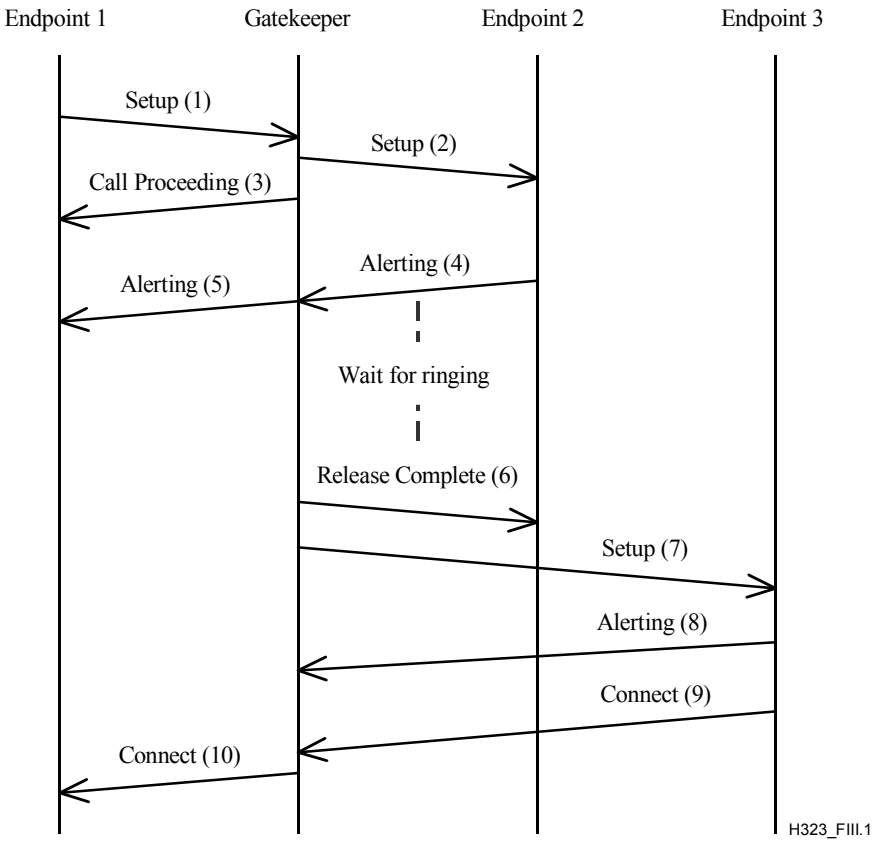


Figure III.1/H.323 – Example of user location using H.225.0 call signalling (RAS signalling not shown for clarity)

A similar approach can be used to provide "divert on busy" service. In this case, Endpoint 2 would return a Release Complete indicating that is busy. The Gatekeeper would then attempt to establish a call to Endpoint 3.

In the scenario shown in Figure III.2, the Gatekeeper attempts to establish contact with Endpoints 2 and 3 simultaneously by sending Setups (2) and (3). In this example the user at Endpoint 3 answers by sending Connect (7). The Gatekeeper forwards the Connect (8) back to Endpoint 1 and clears the call attempt to Endpoint 2 using Release Complete (9). The Gatekeeper should ignore any Connect message received from Endpoint 2 which arrives after the Connect (8) message from Endpoint 3 so that only one call is completed.

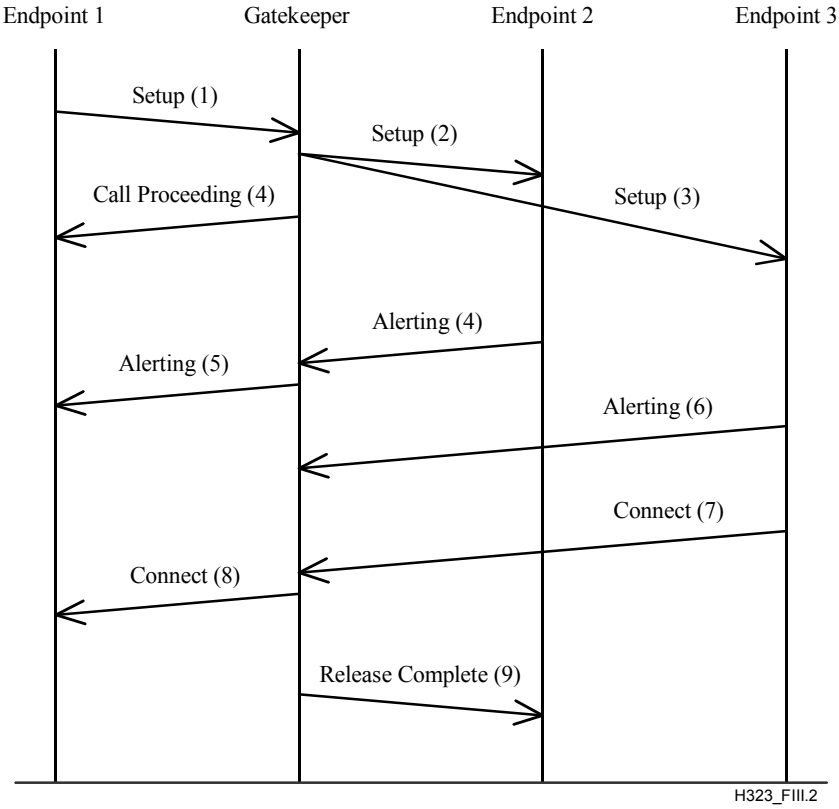


Figure III.2/H.323 – Example of user location using H.225.0 call signalling (RAS signalling not shown for clarity)

Note that if the Gatekeeper is performing this type of user location algorithm, it should not pass the **h245Address** field in any of the Setup Acknowledge, Call Proceeding, and Alerting messages from Endpoint 2 or Endpoint 3 to Endpoint 1 as this may give the wrong result.

Appendix IV

Signalling prioritized alternative logical channels in H.245

IV.1 Introduction

This appendix describes a simple method by which alternative logical channels may be signalled. No coding or semantic changes are required.

This method depends upon the guaranteed ordered delivery that is provided by TCP and is consequently equally applicable to both tunnelled and non-tunnelled H.245 signalling. Tunnelled signalling further depends upon guaranteed processing order where multiple H.245 messages are tunnelled in a single H.225.0 call signalling message.

IV.2 Signalling

All alternative logical channels are identified by the use of a common **forwardLogicalChannelNumber** in **openLogicalChannel** messages, one alternative per message. Messages may be sent either via the H.245 tunnel (one or more OLC messages per call signalling message) or via separate H.245 connection. Alternative logical channels are signalled in order of decreasing desirability, i.e., the first OLC message specifies the **dataType** that the sender of the OLC would prefer to use on the logical channel.

The receiver of these OLC messages is not required to be aware that this method of alternative propositions is being used. Prior to reception of an acceptable OLC request, it will reject unacceptable OLC requests, typically with a cause code of **dataTypeNotSupported**, **dataTypeNotAvailable**, or **unknownDataType**. When an acceptable OLC request is received, the endpoint will respond with an **openLogicalChannelAck** message. Any subsequently received alternative OLCs are rejected by the receiver with a cause code of **unspecified**, as the requested logical channel number will map to a currently open channel.

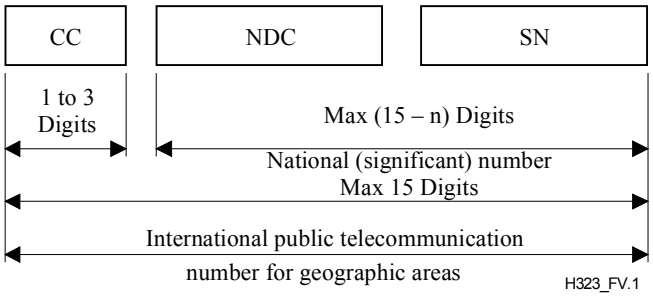
The sender of such a prioritized sequence of **openLogicalChannel** messages must keep track of the number of OLC reject messages received prior to reception of an **openLogicalChannelAck** message in order to determine which proposed alternative was accepted by the peer.

Appendix V

Use of E.164 and ISO/IEC 11571 numbering plans

V.1 E.164 numbering plan

ITU-T defines E.164 numbers the following way for geographic areas (see Figure V.1):



CC Country Code for geographic areas
n Number of digits in the country code
NDC National Destination Code (optional)
SN Subscriber Number

NOTE – National and international prefixes are not part of the international public telecommunication number for geographic areas.

Figure V.1/H.323 – International public telecommunication number structure for geographic areas

Similar descriptions are also defined for non-geographic areas. ITU-T Rec. E.164 further defines country codes (CC) for all the countries and regions of the world.

An international E.164 number always starts with a country code and its total length is always 15 digits or less. More importantly, it does not include any prefixes that are part of a dialling plan (for example, "011" for an international call placed in North America, or "1" for a long-distance

call), nor does it include "#" or "*". The number "49 30 345 67 00" is an E.164 number with CC = 49 for Germany. A national number is the international number stripped of the country code, "30 345 67 00" in this case. The subscriber number is the national number stripped of the national destination code, "345 67 00" in this case.

An E.164 number has global significance: any E.164 number can be reached from any location in the world. A "dialled digit sequence", however, only has significance within a specific domain. Within a typical private numbering plan in an enterprise, for example, a prefix, such as "9", may indicate that a call goes "outside", at which point the local telephone company's dialling plan takes over. Each telephone company or private network is free to choose its own dialling plan. It is also free to change it as it pleases – and frequently does so (adding new area codes, for example).

In a typical geographically determined network where users input telephone numbers manually and where users do not travel too much, having different dialling plans everywhere is usually a problem. However, when a user travels, the user must determine the other network's numbering plan in order to place calls. When computer systems perform the dialling automatically, the user is usually required to customize the dialling software for every region or network.

Because of these issues with varying dialling plans and automated dialling, it is essential to be able to refer to an absolute "telephone number" instead of "what you have to dial to reach it from a specific location." Proper usage of E.164 numbers can resolve these issues. Many systems use E.164 numbers instead of dialled digits: for example, a PBX may gather the dialled digits from a user on a telephone and then initiate a call to the local phone company using an E.164 number in the Called Party Number information element in Q.931. When completing the Called Party Number IE, specifying the numbering plan as "ISDN/telephony numbering plan (ITU-T Rec. E.164)" indicates an E.164 number. Specifying the type of number as "unknown" and specifying the numbering plan as "unknown" indicates dialled digits.

The following are a set of definitions from ITU-T Rec. E.164:

V.1.1 number: A string of decimal digits that uniquely indicates the public network termination point. The number contains the information necessary to route the call to this termination point.

A number can be in a format determined nationally or in an international format. The international format is known as the International Public Telecommunication Number which includes the country code and subsequent digits, but not the international prefix.

V.1.2 numbering plan: A numbering plan specifies the format and structure of the numbers used within that plan. It typically consists of decimal digits segmented into groups in order to identify specific elements used for identification, routing and charging capabilities, e.g., within E.164 to identify countries, national destinations and subscribers.

A numbering plan does not include prefixes, suffixes, and additional information required to complete a call.

The national numbering plan is the national implementation of the E.164 numbering plan.

V.1.3 dialling plan: A string or combination of decimal digits, symbols, and additional information that define the method by which the numbering plan is used. A dialling plan includes the use of prefixes, suffixes, and additional information, supplemental to the numbering plan, required to complete the call.

V.1.4 address: A string or combination of decimal digits, symbols, and additional information which identifies the specific termination point(s) of a connection in a public network(s) or, where applicable, in interconnected private network(s).

V.1.5 prefix: A prefix is an indicator consisting of one or more digits, that allows the selection of different types of number formats, networks and/or service.

V.1.6 international prefix: A digit or combination of digits used to indicate that the number following is an International Public Telecommunication Number.

V.1.7 country code (CC) for geographic areas: The combination of one, two or three digits identifying a specific country, countries in an integrated numbering plan, or a specific geographic area.

V.1.8 national (significant) number [N(S)N]: That portion of the number that follows the country code for geographic areas. The national (significant) number consists of the National Destination Code (NDC) followed by the Subscriber Number (SN). The function and format of the N(S)N is nationally determined.

V.1.9 national destination code (NDC): A nationally optional code field, within the E.164 number plan, which combined with the Subscriber's Number (SN) will constitute the national (significant) number of the international public telecommunication number for geographic areas. The NDC will have a network and/or trunk code selection function.

The NDC can be a decimal digit or a combination of decimal digits (not including any prefix) identifying a numbering area within a country (or group of countries included in one integrated numbering plan or a specific geographic area) and/or network/services.

V.1.10 national (trunk) prefix: A digit or combination of digits used by a calling subscriber, making a call to a subscriber in his own country but outside his own numbering area. It provides access to the automatic outgoing trunk equipment.

V.1.11 subscriber number (SN): The number identifying a subscriber in a network or numbering area.

V.2 Private network number

Private Network Numbers are used in private or virtual private telephony networks, e.g., a corporate network of PBXs and virtual private lines.

ISO/IEC 11571 defines Private Numbering Plan (PNP) number as having up to three regional levels.

A PNP Number shall comprise a sequence of x decimal digits (0,1,2,3,4,5,6,7,8,9) with the possibility that different PNP Numbers within the same PNP can have different values of x. The maximum value of x shall be the same as for the public ISDN numbering plan; see ITU-T Rec. E.164 and Figure V.2.

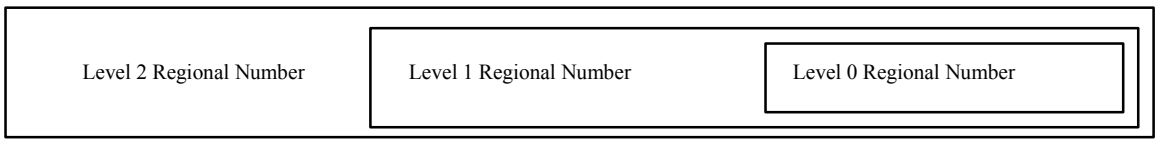


Figure V.2/H.323 – Structure of a PNP Number with three levels of regions

A level n Regional Number (RN) shall have significance only within the level n region to which it applies. When that number is used outside that level n region, it shall be in the form of an RN of level greater than n. Only a Complete Number shall have significance throughout the entire PNP.

A typical example in North America would be a 4-digit "extension" as the Level 0 Regional Number: a 3-digit "location code" combined with the 4-digit "extension" would form the Level 1 Regional Number. The Level 2 Regional Number would be nil.

A prefix could also be used to signal which regional number is used, and would not be part of the regional number per se, but only part of the dialling plan. Again, a typical example would be the use of digit "6" to access a Level 1 Regional Number, and no digit for a Level 0 Regional Number.

The following are a set of definitions from ISO/IEC 11571:

V.2.1 private numbering plan (PNP): The numbering plan explicitly relating to a particular private numbering domain, defined by the PISN Administrator of that domain.

V.2.2 PNP number: A number belonging to a PNP.

V.2.3 region: The entire domain or a sub-domain of a PNP. A region does not necessarily correspond to a geographical area of a PISN.

V.2.4 region code (RC): The leading digits of a PNP Number which identify a region. The RC may be omitted to yield a shortened form of a PNP Number for use internally to that region.

V.2.5 regional number (RN): A particular form of a PNP Number which is unambiguous in the region concerned.

V.2.6 complete number: A number which is unambiguous in the entire PNP, i.e., which corresponds to the highest regional level employed in that PISN.

V.3 H.323 versions 1, 2 and 3 usage

H.323 versions 1, 2 and 3 systems had a terminology problem with respect to dialled digits and real E.164 numbers. References to E.164 addresses in those versions actually referred to dialled digits and not E.164 digits, as the names of the fields implied. In H.323 versions 2 and 3 systems, a real E.164 number was placed in the **publicNumber** field and not in the **e164** field. The **e164** field thus corresponded to a dialled digits sequence.

Beginning with H.323 Version 4 systems, the field **e164** was renamed to **dialledDigits** and the field **publicNumber** was renamed to **e164Number**. The name change was intended to more explicitly convey that dialled digits shall be stored in the **dialledDigits** field and that E.164 numbers shall be stored in the **e164Number** field.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	**Audiovisual and multimedia systems**
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure and Internet protocol aspects
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2004

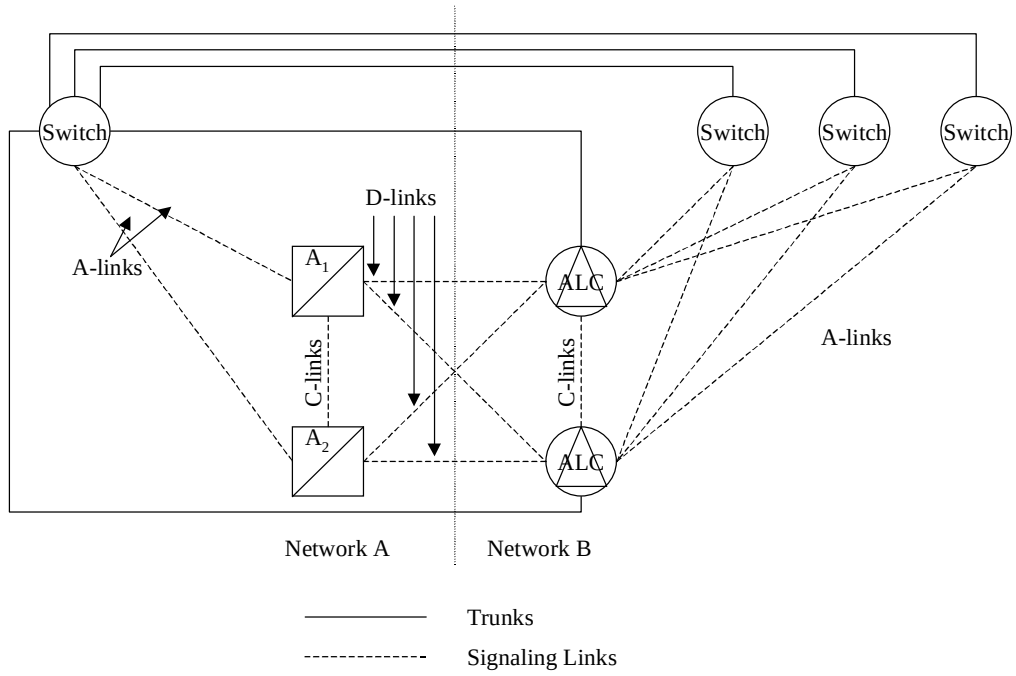


Figure 6-85. Interconnection of an ALC

6.26.3 CCS Network to Voice Over Packet (VOP) Network Interconnection

A conceptual view of the VOP Network and its interconnection to the Public Switch Telephone Network (PSTN) is shown in Figure 6-86. There are various VOP network architectures being developed, but one constant in these architectures is that signaling interconnection to the PSTN shall be based on the SS7 protocol. The architecture discussed in this section is general and is not intended to address all specific VOP network implementations.

The VOP architecture can be broken down by the generic Functional Elements (FEs) contained with the VOP network. Note that the relevant interfaces to the PSTN for these FEs are still under development within the industry. The discussion of these FEs does not imply any specific vendor implementation, but is used to convey the functional composition of the VOP network and how it may interact with the PSTN. The FEs of the VOP as shown in Figure 6-86 are as follows:

- Access Gateway
- Trunk Gateway
- Signaling Gateway
- Call Connection Agent
- Service Agent
- Core Network.

Each of these FEs is discussed in Section 6.26.3.1.

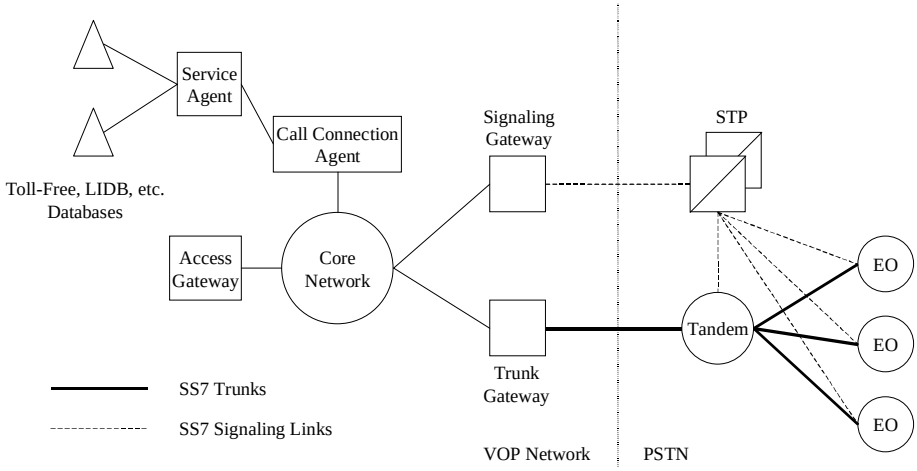


Figure 6-86. CCS Network to VOP Network Interconnection

6.26.3.1 Functional Elements of the VOP Network

6.26.3.1.1 Access Gateway (AG)

This FE supports the line side interface to the VOP network. Traditional phone lines and PBXs currently used for the PSTN can access the VOP network through this FE. As such, this FE provides functions such as packetization, echo control, etc. It may be associated with a specific Call Connection Agent (CCA) that provides it with the necessary call control instructions. On receiving the appropriate commands from the CCA, this FE also provides functions such as audible ringing, power ringing, miscellaneous tones, etc. It is assumed that the AG has the functionality to set up a transport connection through the core network when instructed by the CCA. Thus, when a VOP network offers local phone services, it will have this FE. An end-to-end call initiated by a VOP local phone customer could originate at this FE and terminate at an end office in the CCS/SS7 network.

6.26.3.1.2 Trunk Gateway (TG)

This FE supports a trunk side interface to the PSTN. It terminates circuit-switched trunks in the PSTN and virtual circuits in the packet network (core network) and, as such, provides functions such as packetization of voice. Even though it terminates trunks in the PSTN, it is assumed that this FE does not provide the resource management functions for trunks that it terminates. However, it is assumed that the TG has the capability to set up and manage transport connections

through the core network when instructed by the CCA. It may be associated with a specific CCA or multiple CCAs which provide it with the necessary call control instructions.

6.26.3.1.3 Signaling Gateway (SG)

This FE is used to interconnect the VOP network to the PSTN signaling network. It terminates SS7 links from the PSTN CCS/SS7 networks and thus provides, at a minimum, MTP Level 1 and 2 functionality. It subsequently communicates with one or more CCAs to support the end-to-end signaling for calls and services within the PSTN.

6.26.3.1.4 Call Connection Agent (CCA)

This FE provides much of the necessary call processing functionality to support voice on the packet network. It processes messages received from various other FEs to manage call states. It communicates with other CCAs to set up and manage end-to-end calls. CCAs interact with AGs and TGs using call control commands. For an end-to-end call between the VOP network and the PSTN, the call processing application in the CCA will interact with the PSTN end offices using ISUP that is transported through the SG and the PSTN STPs.

6.26.3.1.5 Service Agent (SA)

This FE generates Transaction Capabilities Application Part (TCAP) messages to interact with SCPs for vertical services (Intelligent Network services) such as Toll-Free and Local Number Portability (LNP). The SA also has the capability to launch TCAP queries to SCPs in the traditional PSTN via the SG.

6.26.3.1.6 Core Network

The core network provides transport for the VOP network. The core network could utilize various technical alternatives such as ATM, IP over ATM, or even pure IP without ATM. The concept behind the signaling and control in the core network is that the signaling for the call control is bearer technology independent and that a separate bearer signaling (e.g., Private Network Node Interface [PNNI] for Asynchronous Transfer Mode [ATM]) is used for establishing a bearer connection across the core network.

6.26.3.2 Interconnection of the SG and TG to the PSTN

As stated in Section 6.26.3, Figure 6-86 shows the interconnection of a VOP network to the PSTN. Interconnection between the VOP network and the PSTN uses the SG and TG of the VOP network and the STP and tandem switch of the PSTN. Specifically, the TG in the VOP network has SS7 trunking to the tandem switch of the PSTN. The SG has SS7 signaling links to the mated STP pair of the PSTN.

The SS7 signaling links between the SG and mated STP pair may be A-links, D-links, or Bridge Links (B-links). The type of link set chosen is influenced by the method used to identify the VOP network from the view of the PSTN. For example, if A-links are used as the method of interconnection, the entire VOP network including all network elements can be viewed as one SEP by the PSTN (i.e., the VOP network is identified by one Point Code [PC]). However, if the mated STP pair in the PSTN can be interconnected to the VOP network by a pair of SGs, B/D-links may be used. In the B/D-link interconnection scenario, the PCs of the SG pair will be different from the PC of the TG. This is the same interconnection architecture used when interconnecting two mated STP pairs in the PSTN. There are also other considerations (e.g., capacity, reliability, SS7 message routing, network management) that must be examined when choosing an interconnection architecture.

Telcordia™ Technologies

Performance from Experience

Telcordia Notes on the Networks

Telcordia Technologies Special Report
SR-2275
Issue 4
October 2000

An SAIC Company

Telcordia Notes on the Networks

SR-2275 replaces SR-2275, *Bellcore Notes on the Networks*, Issue 3, December 1997.

Related documents:

SR-NOTES-SERIES-01, *Telcordia Notes on the Synchronous Optical Network (SONET)*
SR-NOTES-SERIES-02, *Telcordia Notes on Dense Wavelength-Division Multiplexing (DWDM) and Optical Networking*
SR-NOTES-SERIES-03, *Telcordia Notes on Number Portability and Number Pooling*
SR-NOTES-SERIES-04, *Telcordia Notes on the Evolution of Enhanced Emergency Services.*

To obtain copies of this document, contact your company's document coordinator or your Telcordia account manager, or call +1 800.521.2673 (from the USA and Canada) or +1 732.699.5800 (all others), or visit our Web site at www.telcordia.com. Telcordia employees should call +1 732.699.5802.

Trademark Acknowledgments

Telcordia is a trademark of Telcordia Technologies, Inc.
CLCI, CLEI, CLFI, CLLI, ISCP, NMA, and SEAS are trademarks of Telcordia Technologies, Inc.
COMMON LANGUAGE, SPACE, TELEGATE, AIRBOSS, and TIRKS are registered trademarks of Telcordia Technologies, Inc.
CLASS is a service mark of Telcordia Technologies, Inc.
Appletalk is a registered trademark of Apple Computer, Inc.
DECNet is a trademark of Digital Equipment Corporation.
1/1AESS, 4ESS, 5ESS, Dataphone, and SLC are registered trademarks of Lucent Technologies, Inc.
DMS-10, DMS-100F, DATAPATH, and TOPS are trademarks of Nortel.
DMS-100 is a registered trademark of Nortel.
NEAX-61E is a trademark of NEC America, Inc.
EWSD is a registered trademark of Siemens AG.

Any other companies and products not specifically mentioned herein are trademarks or service marks of their respective trademark and service mark owners.

prevents repetitive call attempts to reach distant busy lines, reducing inefficient use of circuits.

- TCAP supports cellular switching functions such as automated user registration.
- There is an ability to protect the transfer of TCAP information against a variety of security threats, as needed.

14.2.2.5 Operations, Maintenance, and Administration Part (OMAP)

OMAP is the layer of the SS7 protocol that is specified for managing the CCS network by using SS7 to transport operation and maintenance information between SPs. Architecturally, OMAP lies above TCAP in the SS7 protocol stack and uses the remote operations service of TCAP to communicate between OMAP applications. OMAP functions include network monitoring, routing updates, signaling network management, automatic call gapping, and consolidation of Operations, Administration, and Maintenance (OA&M) information. OMAP currently performs these functions through the following procedures:

- MTP Routing Verification Test (MRVT) — verifies MTP routing data for a Destination Point Code (DPC).
- SCCP Routing Verification Test (SRVT) — verifies SCCP routing data for a global title address.
- Link Equipment Failure (LEF) — notifies an SP of a signaling terminal or interface equipment failure at the far end of a signaling link.
- Link Fault Sectionalization (LFS) — identifies the failed component on a signaling link.
- Circuit Validation Test (CVT) — ensures that two exchanges have sufficient and consistent translation data for placing a call on a specific circuit of an interexchange circuit group.

More details on the SS7 protocol can be found in Section 6 of this document and in GR-246-CORE.

14.2.3 CCS Call Setup

This section describes an example of basic intraLATA Plain Old Telephone Service (POTS) call setup using CCS and gives additional information on interLATA and ISDN calls.

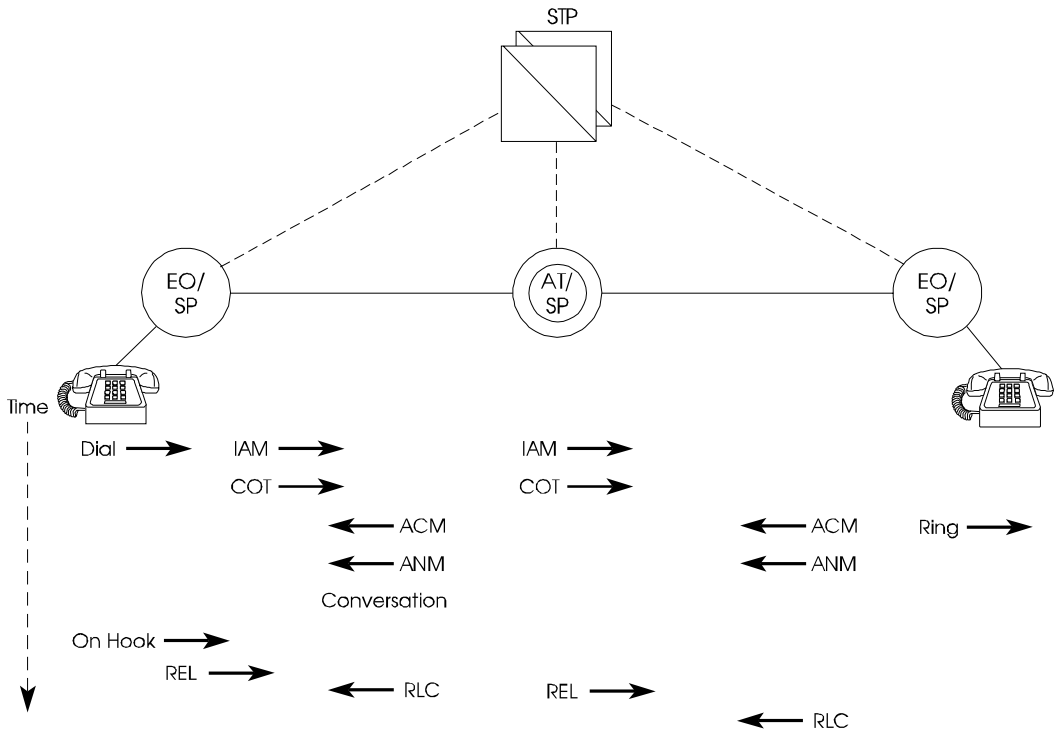
The ISDNUP portion of the SS7 protocol is used to support call setup. The Initial Address Message (IAM) is a mandatory message sent in the forward direction to initiate seizure of an outgoing circuit and to transmit address and other information relating to the routing and handling of a call. The Address Complete Message (ACM) is a message sent in the backward direction indicating that all the address signals

required for routing the call to the called party have been received. The Answer Message (ANM) is a message sent in the backward direction indicating that the call has been answered. The Release Message (REL) is a message sent in either direction indicating that the circuit identified in the message is being released due to the reason (cause) supplied and is ready to be put in the idle state on receipt of the Release Complete Message (RLC). The RLC is a message sent in either direction in response to the receipt of an REL.

The following describes how these messages are used for setup of an intraLATA interoffice call. This call scenario is for an intraLATA call switched through an Access Tandem (AT) where a Continuity Check Message (COT) is required. Refer to Figure 14-4 for a diagram of the scenario.

When the customer dials an intraLATA interoffice call, the originating office sends an IAM over the SS7 signaling link to the AT via the STP pair. The AT then sends an IAM to the terminating office indicating the circuit to be used for the call between the AT and the terminating office. When the terminating office receives the IAM and the COT, it sends an ACM to the AT and applies power ringing to the called party's line. When the AT receives the ACM, it sends an ACM to the originating end office. When the called party goes off-hook, an ANM is sent from the terminating end office to the AT. When the AT receives the ANM, it sends an ANM to the originating end office. After the calling and called party finish their conversation, one party will go on-hook. If the calling party goes on-hook, the originating end office sends an REL to the AT. When the access tandem receives the REL, it sends an RLC to the originating office, and sends an REL to the terminating office. When the terminating office receives the REL, it sends an RLC to the AT.

CCS-based POTS call setup to ICs uses the same switch-to-switch message flow as described above for an intraLATA call. For calls routed to ICs, however, additional optional parameters may be included in the IAM depending on which IC has been selected. Examples of these additional optional parameters include Charge Number (containing Automatic Number Identification [ANI]), Carrier Identification Parameter (containing the 3- or 4-digit Carrier Identification Code [CIC] for the call), and Calling Party Number (CPN).



Legend:

ACM	=	Address Complete Message
ANM	=	Answer Message
AT	=	Access Tandem
COT	=	Continuity Check Message
EO	=	End Office
IAM	=	Initial Address Message
REL	=	Release Message
RLC	=	Release Complete Message
SP	=	Signaling Point
STP	=	Signaling Transfer Point

Figure 14-4. CCS IntraLATA Call Setup

From a message flow standpoint, ISDN call setup is basically the same as POTS call setup. The key differences are in the bearer capabilities that can be requested (such as 64 clear channel) and additional information that can be sent as part of call setup (such as high-layer and low-layer compatibility).

More information on CCS call setup can be found in GR-317-CORE, *Switching System Generic Requirements for Call Control Using the Integrated Services Digital Network User Part (ISDNUP)*; GR-394-CORE, *Switching System Generic Requirements for Interexchange Carrier Interconnection Using the Integrated Services Digital Network User Part (ISDNUP)*; TR-NWT-000444, *Switching System Generic Requirements Supporting ISDN Access Using the ISDN User Part*; and GR-905-CORE, *Common Channel Signaling (CCS) Network Interface Specification (CCSNIS) Supporting Network Interconnection, Message Transfer Part (MTP), and Integrated Services Digital Network User Part (ISDNUP)*. Sections 14.3 through 14.9 describe CCS-based services such as CLASS features, ISDN, ABS, 800 Data Base Service, AIN, and ISCP, and give examples of call processing for those services.

14.3 CLASS Features

CLASS features provide capabilities beyond existing call-management services, such as Custom Calling services, and are generally based on the transport of the Calling Party Number (CPN). Most of the CLASS features do not require the use of specialized Customer Premises Equipment (CPE). The CLASS Calling Identity Delivery (CID) services that do require some form of specialized CPE are Calling Number Delivery (CND), Calling Name Delivery (CNAM), Calling Identity Delivery on Call Waiting (CIDCW), Call Waiting Deluxe (CWD), and Bulk Calling Line Identification (BCLID). The interface between the switch and the customer's CPE display device is described in GR-30-CORE, *Voiceband Data Transmission Interface Generic Requirements.*

CLASS features have been tariffed for use by residence and small-business customers since 1987. The availability of the features varies among LECs, individual regulatory jurisdictions, and exchange serving areas. Some of the features, such as Automatic Callback (AC), require the customer to dial a vertical service code (for AC, the code is *66). Other CLASS features, such as the CID features identified above, do not require any special dialing by the customer. Vertical service codes are administered by the North American Numbering Plan (NANP). Increasingly, LECs are offering selected CLASS features and Custom Calling services on a pay-per-use as well as a subscription basis.

CLASS features can be used by customers with dial pulse (rotary) and Dual-Tone Multifrequency (DTMF) telephone sets. Rotary customers typically dial "11" instead of the "*," which is not available on rotary dials. With the convenience and screening capabilities the CLASS features provide, customers are afforded greater control over their calls. The provision of these services depends on the installation of the CLASS feature hardware and software in the end offices and CCS in the end offices

I E T F®
Search
Chat Live with the IETF Community

Home
About the IETF
- Mission
- Standards Process
- Note Well
- NomCom
- Blog
- Info for Newcomers

Internet-Drafts
- Datatracker
- Search
- Submit

RFC Pages
- Search RFC Ed Index
- RFC Editor Queue

IANA Pages
- Protocol Parameters
- IANA Transition

Working Groups
- WG Charters
- Email Lists
- WG Chairs' Page

Resources
- Community Tools
- Tools Team Pages
- Edu Team Pages
- Mentoring Program
- Tutorials
- Wikis

Meetings
- Upcoming Meetings
- Past Meetings
- Interim Meetings
- Important Dates
- Proceedings

Mailing Lists
- Announcement Lists
- Discussion Lists
- Non-WG Lists

IESG
- Announcements
- Statements
- Members
- Minutes

IPR
- IPR Policy
- File a Disclosure
- Disclosure Search

Liaisons
- Liaison Statements
- Liaison Managers

Contact
- IETF Secretariat
- Ombudsteam
- Media
- Subpoenas
- Report Web Site Errors

Customize View
- Brief
- Normal
- Extended
- Advanced

(Requires Javascript)

The Internet Engineering Task Force (IETF®)

The goal of the IETF is to make the Internet work better.

The mission of the IETF is to make the Internet work better by producing high quality, relevant technical documents that influence the way people design, use, and manage the Internet. Newcomers to the IETF should start here.

News

- IETF 104 in Prague!
- Chair's Blog
- IETF Daily Dose

Next Meeting: IETF 99 Prague, Czech Republic

IETF 99, Prague, Czech Republic (UTC +2)
July 16-21, 2017

- Register
- Important Dates
- Wiki
- Meeting Materials
- Remote Participation
- Hackathon (open to public)

Email Archives

A new mail archive tool realizing the requirements developed in RFC 6778 is now in use:

- Search all IETF email archives

If you choose to log in, use your datatracker credentials.

(Read full announcement in the archives here.)

Recent Meeting: IETF 98 - Chicago, IL

- IETF 98 Information
- IETF 98 Proceedings

Internet-Drafts and RFCs Quick Search

Search

Related Web Pages

IASA and IAOC | IAB | RFC Editor | IANA | IRTF | IETF Trust | ISOC
ISOC Fellowship to the IETF Program

The Internet Engineering Task Force (IETF) is an organized activity of the Internet Society (ISOC).

Internet Society ISOC is a non-profit organization founded in 1992 to provide leadership in Internet-related standards, education, and policy. It is dedicated to ensuring the open development, evolution and use of the Internet for the benefit of people throughout the world. See: www.internetsociety.org

APPENDIX BB

Network Working Group
Request for Comments: 3261
Obsoletes: 2543
Category: Standards Track

J. Rosenberg
dynamicsoft
H. Schulzrinne
Columbia U.
G. Camarillo
Ericsson
A. Johnston
WorldCom
J. Peterson
Neustar
R. Sparks
dynamicsoft
M. Handley
ICIR
E. Schooler
AT&T

June 2002

SIP: Session Initiation Protocol

Status of this Memo

Copyright Notice

Abstract

This document describes Session Initiation Protocol (SIP), an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences.

SIP invitations used to create sessions carry session descriptions that allow participants to agree on a set of compatible media types. SIP makes use of elements called proxy servers to help route requests to the user's current location, authenticate and authorize users for services, implement provider call-routing policies, and provide features to users. SIP also provides a registration function that allows users to upload their current locations for use by proxy servers. SIP runs on top of several different transport protocols.

Table of Contents

1 Introduction

There are many applications of the Internet that require the creation and management of a session, where a session is considered an exchange of data between an association of participants. The implementation of these applications is complicated by the practices of participants: users may move between endpoints, they may be addressable by multiple names, and they may communicate in several different media - sometimes simultaneously. Numerous protocols have been authored that carry various forms of real-time multimedia session data such as voice, video, or text messages. The Session Initiation Protocol (SIP) works in concert with these protocols by

enabling Internet endpoints (called user agents) to discover one another and to agree on a characterization of a session they would like to share. For locating prospective session participants, and for other functions, SIP enables the creation of an infrastructure of network hosts (called proxy servers) to which user agents can send registrations, invitations to sessions, and other requests. SIP is an agile, general-purpose tool for creating, modifying, and terminating sessions that works independently of underlying transport protocols and without dependency on the type of session that is being established.

2 Overview of SIP Functionality

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions (conferences) such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility [27] - users can maintain a single externally visible identifier regardless of their network location.

SIP supports five facets of establishing and terminating multimedia communications:

- User location: determination of the end system to be used for communication;
- User availability: determination of the willingness of the called party to engage in communications;
- User capabilities: determination of the media and media parameters to be used;
- Session setup: "ringing", establishment of session parameters at both called and calling party;
- Session management: including transfer and termination of sessions, modifying session parameters, and invoking services.

SIP is not a vertically integrated communications system. SIP is rather a component that can be used with other IETF protocols to build a complete multimedia architecture. Typically, these architectures will include protocols such as the Real-time Transport Protocol (RTP) (RFC 1889 [28]) for transporting real-time data and providing QoS feedback, the Real-Time streaming protocol (RTSP) (RFC 2326 [29]) for controlling delivery of streaming media, the Media

Gateway Control Protocol (MEGACO) (RFC 3015 [30]) for controlling gateways to the Public Switched Telephone Network (PSTN), and the Session Description Protocol (SDP) (RFC 2327 [1]) for describing multimedia sessions. Therefore, SIP should be used in conjunction with other protocols in order to provide complete services to the users. However, the basic functionality and operation of SIP does not depend on any of these protocols.

SIP does not provide services. Rather, SIP provides primitives that can be used to implement different services. For example, SIP can locate a user and deliver an opaque object to his current location. If this primitive is used to deliver a session description written in SDP, for instance, the endpoints can agree on the parameters of a session. If the same primitive is used to deliver a photo of the caller as well as the session description, a "caller ID" service can be easily implemented. As this example shows, a single primitive is typically used to provide several different services.

SIP does not offer conference control services such as floor control or voting and does not prescribe how a conference is to be managed. SIP can be used to initiate a session that uses some other conference control protocol. Since SIP messages and the sessions they establish can pass through entirely different networks, SIP cannot, and does not, provide any kind of network resource reservation capabilities.

The nature of the services provided make security particularly important. To that end, SIP provides a suite of security services, which include denial-of-service prevention, authentication (both user to user and proxy to user), integrity protection, and encryption and privacy services.

SIP works with both IPv4 and IPv6.

3 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [2] and indicate requirement levels for compliant SIP implementations.

4 Overview of Operation

This section introduces the basic operations of SIP using simple examples. This section is tutorial in nature and does not contain any normative statements.

The first example shows the basic functions of SIP: location of an end point, signal of a desire to communicate, negotiation of session parameters to establish the session, and teardown of the session once established.

Figure 1 shows a typical example of a SIP message exchange between two users, Alice and Bob. (Each message is labeled with the letter "F" and a number for reference by the text.) In this example, Alice uses a SIP application on her PC (referred to as a softphone) to call Bob on his SIP phone over the Internet. Also shown are two SIP proxy servers that act on behalf of Alice and Bob to facilitate the session establishment. This typical arrangement is often referred to as the "SIP trapezoid" as shown by the geometric shape of the dotted lines in Figure 1.

Alice "calls" Bob using his SIP identity, a type of Uniform Resource Identifier (URI) called a SIP URI. SIP URIs are defined in Section 19.1. It has a similar form to an email address, typically containing a username and a host name. In this case, it is sip:bob@biloxi.com, where biloxi.com is the domain of Bob's SIP service provider. Alice has a SIP URI of sip:alice@atlanta.com. Alice might have typed in Bob's URI or perhaps clicked on a hyperlink or an entry in an address book. SIP also provides a secure URI, called a SIPS URI. An example would be sips:bob@biloxi.com. A call made to a SIPS URI guarantees that secure, encrypted transport (namely TLS) is used to carry all SIP messages from the caller to the domain of the callee. From there, the request is sent securely to the callee, but with security mechanisms that depend on the policy of the domain of the callee.

SIP is based on an HTTP-like request/response transaction model. Each transaction consists of a request that invokes a particular method, or function, on the server and at least one response. In this example, the transaction begins with Alice's softphone sending an INVITE request addressed to Bob's SIP URI. INVITE is an example of a SIP method that specifies the action that the requestor (Alice) wants the server (Bob) to take. The INVITE request contains a number of header fields. Header fields are named attributes that provide additional information about a message. The ones present in an INVITE include a unique identifier for the call, the destination address, Alice's address, and information about the type of session that Alice wishes to establish with Bob. The INVITE (message F1 in Figure 1) might look like this:

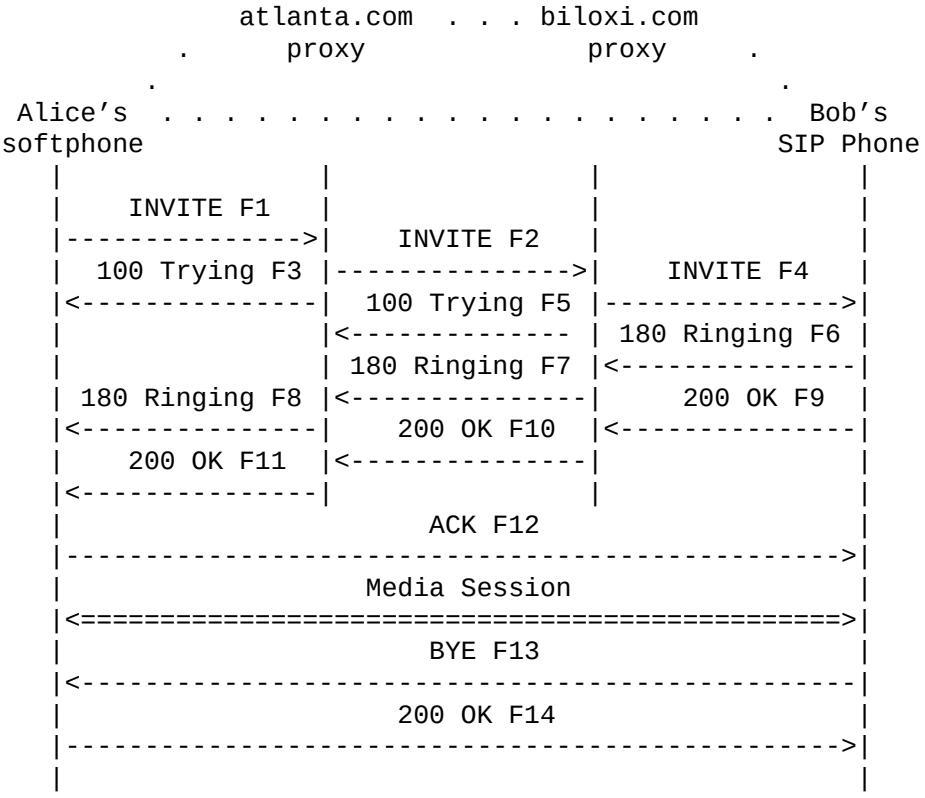


Figure 1: SIP session setup example with SIP trapezoid

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)
```

The first line of the text-encoded message contains the method name (INVITE). The lines that follow are a list of header fields. This example contains a minimum required set. The header fields are briefly described below:

Via contains the address (pc33.atlanta.com) at which Alice is expecting to receive responses to this request. It also contains a branch parameter that identifies this transaction.

To contains a display name (Bob) and a SIP or SIPS URI (sip:bob@biloxi.com) towards which the request was originally directed. Display names are described in RFC 2822 [3].

From also contains a display name (Alice) and a SIP or SIPS URI (sip:alice@atlanta.com) that indicate the originator of the request. This header field also has a tag parameter containing a random string (1928301774) that was added to the URI by the softphone. It is used for identification purposes.

Call-ID contains a globally unique identifier for this call, generated by the combination of a random string and the softphone's host name or IP address. The combination of the To tag, From tag, and Call-ID completely defines a peer-to-peer SIP relationship between Alice and Bob and is referred to as a dialog.

CSeq or Command Sequence contains an integer and a method name. The CSeq number is incremented for each new request within a dialog and is a traditional sequence number.

Contact contains a SIP or SIPS URI that represents a direct route to contact Alice, usually composed of a username at a fully qualified domain name (FQDN). While an FQDN is preferred, many end systems do not have registered domain names, so IP addresses are permitted. While the Via header field tells other elements where to send the response, the Contact header field tells other elements where to send future requests.

Max-Forwards serves to limit the number of hops a request can make on the way to its destination. It consists of an integer that is decremented by one at each hop.

Content-Type contains a description of the message body (not shown).

Content-Length contains an octet (byte) count of the message body.

The complete set of SIP header fields is defined in Section 20.

The details of the session, such as the type of media, codec, or sampling rate, are not described using SIP. Rather, the body of a SIP message contains a description of the session, encoded in some other protocol format. One such format is the Session Description Protocol (SDP) (RFC 2327 [1]). This SDP message (not shown in the

example) is carried by the SIP message in a way that is analogous to a document attachment being carried by an email message, or a web page being carried in an HTTP message.

Since the softphone does not know the location of Bob or the SIP server in the biloxi.com domain, the softphone sends the INVITE to the SIP server that serves Alice's domain, atlanta.com. The address of the atlanta.com SIP server could have been configured in Alice's softphone, or it could have been discovered by DHCP, for example.

The atlanta.com SIP server is a type of SIP server known as a proxy server. A proxy server receives SIP requests and forwards them on behalf of the requestor. In this example, the proxy server receives the INVITE request and sends a 100 (Trying) response back to Alice's softphone. The 100 (Trying) response indicates that the INVITE has been received and that the proxy is working on her behalf to route the INVITE to the destination. Responses in SIP use a three-digit code followed by a descriptive phrase. This response contains the same To, From, Call-ID, CSeq and branch parameter in the Via as the INVITE, which allows Alice's softphone to correlate this response to the sent INVITE. The atlanta.com proxy server locates the proxy server at biloxi.com, possibly by performing a particular type of DNS (Domain Name Service) lookup to find the SIP server that serves the biloxi.com domain. This is described in [4]. As a result, it obtains the IP address of the biloxi.com proxy server and forwards, or proxies, the INVITE request there. Before forwarding the request, the atlanta.com proxy server adds an additional Via header field value that contains its own address (the INVITE already contains Alice's address in the first Via). The biloxi.com proxy server receives the INVITE and responds with a 100 (Trying) response back to the atlanta.com proxy server to indicate that it has received the INVITE and is processing the request. The proxy server consults a database, generically called a location service, that contains the current IP address of Bob. (We shall see in the next section how this database can be populated.) The biloxi.com proxy server adds another Via header field value with its own address to the INVITE and proxies it to Bob's SIP phone.

Bob's SIP phone receives the INVITE and alerts Bob to the incoming call from Alice so that Bob can decide whether to answer the call, that is, Bob's phone rings. Bob's SIP phone indicates this in a 180 (Ringing) response, which is routed back through the two proxies in the reverse direction. Each proxy uses the Via header field to determine where to send the response and removes its own address from the top. As a result, although DNS and location service lookups were required to route the initial INVITE, the 180 (Ringing) response can be returned to the caller without lookups or without state being

maintained in the proxies. This also has the desirable property that each proxy that sees the INVITE will also see all responses to the INVITE.

When Alice's softphone receives the 180 (Ringing) response, it passes this information to Alice, perhaps using an audio ringback tone or by displaying a message on Alice's screen.

In this example, Bob decides to answer the call. When he picks up the handset, his SIP phone sends a 200 (OK) response to indicate that the call has been answered. The 200 (OK) contains a message body with the SDP media description of the type of session that Bob is willing to establish with Alice. As a result, there is a two-phase exchange of SDP messages: Alice sent one to Bob, and Bob sent one back to Alice. This two-phase exchange provides basic negotiation capabilities and is based on a simple offer/answer model of SDP exchange. If Bob did not wish to answer the call or was busy on another call, an error response would have been sent instead of the 200 (OK), which would have resulted in no media session being established. The complete list of SIP response codes is in Section 21. The 200 (OK) (message F9 in Figure 1) might look like this as Bob sends it out:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com
   ;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com
   ;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com
   ;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

The first line of the response contains the response code (200) and the reason phrase (OK). The remaining lines contain header fields. The Via, To, From, Call-ID, and CSeq header fields are copied from the INVITE request. (There are three Via header field values - one added by Alice's SIP phone, one added by the atlanta.com proxy, and one added by the biloxi.com proxy.) Bob's SIP phone has added a tag parameter to the To header field. This tag will be incorporated by both endpoints into the dialog and will be included in all future

requests and responses in this call. The Contact header field contains a URI at which Bob can be directly reached at his SIP phone. The Content-Type and Content-Length refer to the message body (not shown) that contains Bob's SDP media information.

In addition to DNS and location service lookups shown in this example, proxy servers can make flexible "routing decisions" to decide where to send a request. For example, if Bob's SIP phone returned a 486 (Busy Here) response, the biloxi.com proxy server could proxy the INVITE to Bob's voicemail server. A proxy server can also send an INVITE to a number of locations at the same time. This type of parallel search is known as forking.

In this case, the 200 (OK) is routed back through the two proxies and is received by Alice's softphone, which then stops the ringback tone and indicates that the call has been answered. Finally, Alice's softphone sends an acknowledgement message, ACK, to Bob's SIP phone to confirm the reception of the final response (200 (OK)). In this example, the ACK is sent directly from Alice's softphone to Bob's SIP phone, bypassing the two proxies. This occurs because the endpoints have learned each other's address from the Contact header fields through the INVITE/200 (OK) exchange, which was not known when the initial INVITE was sent. The lookups performed by the two proxies are no longer needed, so the proxies drop out of the call flow. This completes the INVITE/200/ACK three-way handshake used to establish SIP sessions. Full details on session setup are in Section 13.

Alice and Bob's media session has now begun, and they send media packets using the format to which they agreed in the exchange of SDP. In general, the end-to-end media packets take a different path from the SIP signaling messages.

During the session, either Alice or Bob may decide to change the characteristics of the media session. This is accomplished by sending a re-INVITE containing a new media description. This re-INVITE references the existing dialog so that the other party knows that it is to modify an existing session instead of establishing a new session. The other party sends a 200 (OK) to accept the change. The requestor responds to the 200 (OK) with an ACK. If the other party does not accept the change, he sends an error response such as 488 (Not Acceptable Here), which also receives an ACK. However, the failure of the re-INVITE does not cause the existing call to fail - the session continues using the previously negotiated characteristics. Full details on session modification are in Section 14.

At the end of the call, Bob disconnects (hangs up) first and generates a BYE message. This BYE is routed directly to Alice's softphone, again bypassing the proxies. Alice confirms receipt of the BYE with a 200 (OK) response, which terminates the session and the BYE transaction. No ACK is sent - an ACK is only sent in response to a response to an INVITE request. The reasons for this special handling for INVITE will be discussed later, but relate to the reliability mechanisms in SIP, the length of time it can take for a ringing phone to be answered, and forking. For this reason, request handling in SIP is often classified as either INVITE or non-INVITE, referring to all other methods besides INVITE. Full details on session termination are in Section 15.

Section 24.2 describes the messages shown in Figure 1 in full.

In some cases, it may be useful for proxies in the SIP signaling path to see all the messaging between the endpoints for the duration of the session. For example, if the biloxi.com proxy server wished to remain in the SIP messaging path beyond the initial INVITE, it would add to the INVITE a required routing header field known as Record-Route that contained a URI resolving to the hostname or IP address of the proxy. This information would be received by both Bob's SIP phone and (due to the Record-Route header field being passed back in the 200 (OK)) Alice's softphone and stored for the duration of the dialog. The biloxi.com proxy server would then receive and proxy the ACK, BYE, and 200 (OK) to the BYE. Each proxy can independently decide to receive subsequent messages, and those messages will pass through all proxies that elect to receive it. This capability is frequently used for proxies that are providing mid-call features.

Registration is another common operation in SIP. Registration is one way that the biloxi.com server can learn the current location of Bob. Upon initialization, and at periodic intervals, Bob's SIP phone sends REGISTER messages to a server in the biloxi.com domain known as a SIP registrar. The REGISTER messages associate Bob's SIP or SIPS URI (sip:bob@biloxi.com) with the machine into which he is currently logged (conveyed as a SIP or SIPS URI in the Contact header field). The registrar writes this association, also called a binding, to a database, called the location service, where it can be used by the proxy in the biloxi.com domain. Often, a registrar server for a domain is co-located with the proxy for that domain. It is an important concept that the distinction between types of SIP servers is logical, not physical.

Bob is not limited to registering from a single device. For example, both his SIP phone at home and the one in the office could send registrations. This information is stored together in the location

service and allows a proxy to perform various types of searches to locate Bob. Similarly, more than one user can be registered on a single device at the same time.

The location service is just an abstract concept. It generally contains information that allows a proxy to input a URI and receive a set of zero or more URIs that tell the proxy where to send the request. Registrations are one way to create this information, but not the only way. Arbitrary mapping functions can be configured at the discretion of the administrator.

Finally, it is important to note that in SIP, registration is used for routing incoming SIP requests and has no role in authorizing outgoing requests. Authorization and authentication are handled in SIP either on a request-by-request basis with a challenge/response mechanism, or by using a lower layer scheme as discussed in Section 26.

The complete set of SIP message details for this registration example is in Section 24.1.

Additional operations in SIP, such as querying for the capabilities of a SIP server or client using OPTIONS, or canceling a pending request using CANCEL, will be introduced in later sections.

5 Structure of the Protocol

SIP is structured as a layered protocol, which means that its behavior is described in terms of a set of fairly independent processing stages with only a loose coupling between each stage. The protocol behavior is described as layers for the purpose of presentation, allowing the description of functions common across elements in a single section. It does not dictate an implementation in any way. When we say that an element "contains" a layer, we mean it is compliant to the set of rules defined by that layer.

Not every element specified by the protocol contains every layer. Furthermore, the elements specified by SIP are logical elements, not physical ones. A physical realization can choose to act as different logical elements, perhaps even on a transaction-by-transaction basis.

The lowest layer of SIP is its syntax and encoding. Its encoding is specified using an augmented Backus-Naur Form grammar (BNF). The complete BNF is specified in Section 25; an overview of a SIP message's structure can be found in Section 7.

The second layer is the transport layer. It defines how a client sends requests and receives responses and how a server receives requests and sends responses over the network. All SIP elements contain a transport layer. The transport layer is described in Section 18.

The third layer is the transaction layer. Transactions are a fundamental component of SIP. A transaction is a request sent by a client transaction (using the transport layer) to a server transaction, along with all responses to that request sent from the server transaction back to the client. The transaction layer handles application-layer retransmissions, matching of responses to requests, and application-layer timeouts. Any task that a user agent client (UAC) accomplishes takes place using a series of transactions. Discussion of transactions can be found in Section 17. User agents contain a transaction layer, as do stateful proxies. Stateless proxies do not contain a transaction layer. The transaction layer has a client component (referred to as a client transaction) and a server component (referred to as a server transaction), each of which are represented by a finite state machine that is constructed to process a particular request.

The layer above the transaction layer is called the transaction user (TU). Each of the SIP entities, except the stateless proxy, is a transaction user. When a TU wishes to send a request, it creates a client transaction instance and passes it the request along with the destination IP address, port, and transport to which to send the request. A TU that creates a client transaction can also cancel it. When a client cancels a transaction, it requests that the server stop further processing, revert to the state that existed before the transaction was initiated, and generate a specific error response to that transaction. This is done with a CANCEL request, which constitutes its own transaction, but references the transaction to be cancelled (Section 9).

The SIP elements, that is, user agent clients and servers, stateless and stateful proxies and registrars, contain a core that distinguishes them from each other. Cores, except for the stateless proxy, are transaction users. While the behavior of the UAC and UAS cores depends on the method, there are some common rules for all methods (Section 8). For a UAC, these rules govern the construction of a request; for a UAS, they govern the processing of a request and generating a response. Since registrations play an important role in SIP, a UAS that handles a REGISTER is given the special name registrar. Section 10 describes UAC and UAS core behavior for the REGISTER method. Section 11 describes UAC and UAS core behavior for the OPTIONS method, used for determining the capabilities of a UA.

Certain other requests are sent within a dialog. A dialog is a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages and proper routing of requests between the user agents. The INVITE method is the only way defined in this specification to establish a dialog. When a UAC sends a request that is within the context of a dialog, it follows the common UAC rules as discussed in Section 8 but also the rules for mid-dialog requests. Section 12 discusses dialogs and presents the procedures for their construction and maintenance, in addition to construction of requests within a dialog.

The most important method in SIP is the INVITE method, which is used to establish a session between participants. A session is a collection of participants, and streams of media between them, for the purposes of communication. Section 13 discusses how sessions are initiated, resulting in one or more SIP dialogs. Section 14 discusses how characteristics of that session are modified through the use of an INVITE request within a dialog. Finally, section 15 discusses how a session is terminated.

The procedures of Sections 8, 10, 11, 12, 13, 14, and 15 deal entirely with the UA core (Section 9 describes cancellation, which applies to both UA core and proxy core). Section 16 discusses the proxy element, which facilitates routing of messages between user agents.

6 Definitions

The following terms have special significance for SIP.

Address-of-Record: An address-of-record (AOR) is a SIP or SIPS URI that points to a domain with a location service that can map the URI to another URI where the user might be available. Typically, the location service is populated through registrations. An AOR is frequently thought of as the "public address" of the user.

Back-to-Back User Agent: A back-to-back user agent (B2BUA) is a logical entity that receives a request and processes it as a user agent server (UAS). In order to determine how the request should be answered, it acts as a user agent client (UAC) and generates requests. Unlike a proxy server, it maintains dialog state and must participate in all requests sent on the dialogs it has established. Since it is a concatenation of a UAC and UAS, no explicit definitions are needed for its behavior.

Call: A call is an informal term that refers to some communication between peers, generally set up for the purposes of a multimedia conversation.

Call Leg: Another name for a dialog [31]; no longer used in this specification.

Call Stateful: A proxy is call stateful if it retains state for a dialog from the initiating INVITE to the terminating BYE request. A call stateful proxy is always transaction stateful, but the converse is not necessarily true.

Client: A client is any network element that sends SIP requests and receives SIP responses. Clients may or may not interact directly with a human user. User agent clients and proxies are clients.

Conference: A multimedia session (see below) that contains multiple participants.

Core: Core designates the functions specific to a particular type of SIP entity, i.e., specific to either a stateful or stateless proxy, a user agent or registrar. All cores, except those for the stateless proxy, are transaction users.

Dialog: A dialog is a peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is established by SIP messages, such as a 2xx response to an INVITE request. A dialog is identified by a call identifier, local tag, and a remote tag. A dialog was formerly known as a call leg in RFC 2543.

Downstream: A direction of message forwarding within a transaction that refers to the direction that requests flow from the user agent client to user agent server.

Final Response: A response that terminates a SIP transaction, as opposed to a provisional response that does not. All 2xx, 3xx, 4xx, 5xx and 6xx responses are final.

Header: A header is a component of a SIP message that conveys information about the message. It is structured as a sequence of header fields.

Header Field: A header field is a component of the SIP message header. A header field can appear as one or more header field rows. Header field rows consist of a header field name and zero or more header field values. Multiple header field values on a

given header field row are separated by commas. Some header fields can only have a single header field value, and as a result, always appear as a single header field row.

Header Field Value: A header field value is a single value; a header field consists of zero or more header field values.

Home Domain: The domain providing service to a SIP user. Typically, this is the domain present in the URI in the address-of-record of a registration.

Informational Response: Same as a provisional response.

Initiator, Calling Party, Caller: The party initiating a session (and dialog) with an INVITE request. A caller retains this role from the time it sends the initial INVITE that established a dialog until the termination of that dialog.

Invitation: An INVITE request.

Invitee, Invited User, Called Party, Callee: The party that receives an INVITE request for the purpose of establishing a new session. A callee retains this role from the time it receives the INVITE until the termination of the dialog established by that INVITE.

Location Service: A location service is used by a SIP redirect or proxy server to obtain information about a callee's possible location(s). It contains a list of bindings of address-of-record keys to zero or more contact addresses. The bindings can be created and removed in many ways; this specification defines a REGISTER method that updates the bindings.

Loop: A request that arrives at a proxy, is forwarded, and later arrives back at the same proxy. When it arrives the second time, its Request-URI is identical to the first time, and other header fields that affect proxy operation are unchanged, so that the proxy would make the same processing decision on the request it made the first time. Looped requests are errors, and the procedures for detecting them and handling them are described by the protocol.

Loose Routing: A proxy is said to be loose routing if it follows the procedures defined in this specification for processing of the Route header field. These procedures separate the destination of the request (present in the Request-URI) from

the set of proxies that need to be visited along the way (present in the Route header field). A proxy compliant to these mechanisms is also known as a loose router.

Message: Data sent between SIP elements as part of the protocol. SIP messages are either requests or responses.

Method: The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are INVITE and BYE.

Outbound Proxy: A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.

Parallel Search: In a parallel search, a proxy issues several requests to possible user locations upon receiving an incoming request. Rather than issuing one request and then waiting for the final response before issuing the next request as in a sequential search, a parallel search issues requests without waiting for the result of previous requests.

Provisional Response: A response used by the server to indicate progress, but that does not terminate a SIP transaction. 1xx responses are provisional, other responses are considered final.

Proxy, Proxy Server: An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity "closer" to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.

Recursion: A client recurses on a 3xx response when it generates a new request to one or more of the URIs in the Contact header field in the response.

Redirect Server: A redirect server is a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.

Registrar: A registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

Regular Transaction: A regular transaction is any transaction with a method other than INVITE, ACK, or CANCEL.

Request: A SIP message sent from a client to a server, for the purpose of invoking a particular operation.

Response: A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server.

Ringback: Ringback is the signaling tone produced by the calling party's application indicating that a called party is being alerted (ringing).

Route Set: A route set is a collection of ordered SIP or SIPS URI which represent a list of proxies that must be traversed when sending a particular request. A route set can be learned, through headers like Record-Route, or it can be configured.

Server: A server is a network element that receives requests in order to service them and sends back responses to those requests. Examples of servers are proxies, user agent servers, redirect servers, and registrars.

Sequential Search: In a sequential search, a proxy server attempts each contact address in sequence, proceeding to the next one only after the previous has generated a final response. A 2xx or 6xx class final response always terminates a sequential search.

Session: From the SDP specification: "A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session." (RFC 2327 [1]) (A session as defined for SDP can comprise one or more RTP sessions.) As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the SDP user name, session id, network type, address type, and address elements in the origin field.

SIP Transaction: A SIP transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response

sent from the server to the client. If the request is INVITE and the final response is a non-2xx, the transaction also includes an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate transaction.

Spiral: A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but this time differs in a way that will result in a different processing decision than the original request. Typically, this means that the request's Request-URI differs from its previous arrival. A spiral is not an error condition, unlike a loop. A typical cause for this is call forwarding. A user calls joe@example.com. The example.com proxy forwards it to Joe's PC, which in turn, forwards it to bob@example.com. This request is proxied back to the example.com proxy. However, this is not a loop. Since the request is targeted at a different user, it is considered a spiral, and is a valid condition.

Stateful Proxy: A logical entity that maintains the client and server transaction state machines defined by this specification during the processing of a request, also known as a transaction stateful proxy. The behavior of a stateful proxy is further defined in Section 16. A (transaction) stateful proxy is not the same as a call stateful proxy.

Stateless Proxy: A logical entity that does not maintain the client or server transaction state machines defined in this specification when it processes requests. A stateless proxy forwards every request it receives downstream and every response it receives upstream.

Strict Routing: A proxy is said to be strict routing if it follows the Route processing rules of RFC 2543 and many prior work in progress versions of this RFC. That rule caused proxies to destroy the contents of the Request-URI when a Route header field was present. Strict routing behavior is not used in this specification, in favor of a loose routing behavior. Proxies that perform strict routing are also known as strict routers.

Target Refresh Request: A target refresh request sent within a dialog is defined as a request that can modify the remote target of the dialog.

Transaction User (TU): The layer of protocol processing that resides above the transaction layer. Transaction users include the UAC core, UAS core, and proxy core.

Upstream: A direction of message forwarding within a transaction that refers to the direction that responses flow from the user agent server back to the user agent client.

URL-encoded: A character string encoded according to RFC 2396, Section 2.4 [5].

User Agent Client (UAC): A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a user agent server for the processing of that transaction.

UAC Core: The set of processing functions required of a UAC that reside above the transaction and transport layers.

User Agent Server (UAS): A user agent server is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later, it assumes the role of a user agent client for the processing of that transaction.

UAS Core: The set of processing functions required at a UAS that resides above the transaction and transport layers.

User Agent (UA): A logical entity that can act as both a user agent client and user agent server.

The role of UAC and UAS, as well as proxy and redirect servers, are defined on a transaction-by-transaction basis. For example, the user agent initiating a call acts as a UAC when sending the initial INVITE request and as a UAS when receiving a BYE request from the callee. Similarly, the same software can act as a proxy server for one request and as a redirect server for the next request.

Proxy, location, and registrar servers defined above are logical entities; implementations MAY combine them into a single application.

7 SIP Messages

SIP is a text-based protocol and uses the UTF-8 charset (RFC 2279 [7]).

A SIP message is either a request from a client to a server, or a response from a server to a client.

Both Request (section 7.1) and Response (section 7.2) messages use the basic format of RFC 2822 [3], even though the syntax differs in character set and syntax specifics. (SIP allows header fields that would not be valid RFC 2822 header fields, for example.) Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body.

```
generic-message  =  start-line
                    *message-header
                    CRLF
                    [ message-body ]
start-line       =  Request-Line / Status-Line
```

The start-line, each message-header line, and the empty line MUST be terminated by a carriage-return line-feed sequence (CRLF). Note that the empty line MUST be present even if the message-body is not.

Except for the above difference in character sets, much of SIP's message and header field syntax is identical to HTTP/1.1. Rather than repeating the syntax and semantics here, we use [HX.Y] to refer to Section X.Y of the current HTTP/1.1 specification (RFC 2616 [8]).

However, SIP is not an extension of HTTP.

7.1 Requests

SIP requests are distinguished by having a Request-Line for a start-line. A Request-Line contains a method name, a Request-URI, and the protocol version separated by a single space (SP) character.

The Request-Line ends with CRLF. No CR or LF are allowed except in the end-of-line CRLF sequence. No linear whitespace (LWS) is allowed in any of the elements.

```
Request-Line  =  Method SP Request-URI SP SIP-Version CRLF
```

Method: This specification defines six methods: REGISTER for registering contact information, INVITE, ACK, and CANCEL for setting up sessions, BYE for terminating sessions, and OPTIONS for querying servers about their capabilities. SIP extensions, documented in standards track RFCs, may define additional methods.

Request-URI: The Request-URI is a SIP or SIPS URI as described in Section 19.1 or a general URI (RFC 2396 [5]). It indicates the user or service to which this request is being addressed. The Request-URI MUST NOT contain unescaped spaces or control characters and MUST NOT be enclosed in "<>".

SIP elements MAY support Request-URIs with schemes other than "sip" and "sips", for example the "tel" URI scheme of RFC 2806 [9]. SIP elements MAY translate non-SIP URIs using any mechanism at their disposal, resulting in SIP URI, SIPS URI, or some other scheme.

SIP-Version: Both request and response messages include the version of SIP in use, and follow [H3.1] (with HTTP replaced by SIP, and HTTP/1.1 replaced by SIP/2.0) regarding version ordering, compliance requirements, and upgrading of version numbers. To be compliant with this specification, applications sending SIP messages MUST include a SIP-Version of "SIP/2.0". The SIP-Version string is case-insensitive, but implementations MUST send upper-case.

Unlike HTTP/1.1, SIP treats the version number as a literal string. In practice, this should make no difference.

7.2 Responses

SIP responses are distinguished from requests by having a Status-Line as their start-line. A Status-Line consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single SP character.

No CR or LF is allowed except in the final CRLF sequence.

```
Status-Line  =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
```

The Status-Code is a 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata, whereas the Reason-Phrase is intended for the human user. A client is not required to examine or display the Reason-Phrase.

While this specification suggests specific wording for the reason phrase, implementations MAY choose other text, for example, in the language indicated in the Accept-Language header field of the request.

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. For this reason, any response with a status code between 100 and 199 is referred to as a "1xx response", any response with a status code between 200 and 299 as a "2xx response", and so on. SIP/2.0 allows six values for the first digit:

1xx: Provisional -- request received, continuing to process the request;

2xx: Success -- the action was successfully received, understood, and accepted;

3xx: Redirection -- further action needs to be taken in order to complete the request;

4xx: Client Error -- the request contains bad syntax or cannot be fulfilled at this server;

5xx: Server Error -- the server failed to fulfill an apparently valid request;

6xx: Global Failure -- the request cannot be fulfilled at any server.

Section 21 defines these classes and describes the individual codes.

7.3 Header Fields

SIP header fields are similar to HTTP header fields in both syntax and semantics. In particular, SIP header fields follow the [H4.2] definitions of syntax for the message-header and the rules for extending header fields over multiple lines. However, the latter is specified in HTTP with implicit whitespace and folding. This specification conforms to RFC 2234 [10] and uses only explicit whitespace and folding as an integral part of the grammar.

[H4.2] also specifies that multiple header fields of the same field name whose value is a comma-separated list can be combined into one header field. That applies to SIP as well, but the specific rule is different because of the different grammars. Specifically, any SIP header whose grammar is of the form

```
header  =  "header-name" HCOLON header-value *(COMMA header-value)
```

allows for combining header fields of the same name into a comma-separated list. The Contact header field allows a comma-separated list unless the header field value is "*".

7.3.1 Header Field Format

Header fields follow the same generic header format as that given in Section 2.2 of RFC 2822 [3]. Each header field consists of a field name followed by a colon (":") and the field value.

```
field-name: field-value
```

The formal grammar for a message-header specified in Section 25 allows for an arbitrary amount of whitespace on either side of the colon; however, implementations should avoid spaces between the field name and the colon and use a single space (SP) between the colon and the field-value.

```
Subject:            lunch
Subject      :      lunch
Subject            :lunch
Subject: lunch
```

Thus, the above are all valid and equivalent, but the last is the preferred form.

Header fields can be extended over multiple lines by preceding each extra line with at least one SP or horizontal tab (HT). The line break and the whitespace at the beginning of the next line are treated as a single SP character. Thus, the following are equivalent:

```
Subject: I know you're there, pick up the phone and talk to me!
Subject: I know you're there,
         pick up the phone
         and talk to me!
```

The relative order of header fields with different field names is not significant. However, it is RECOMMENDED that header fields which are needed for proxy processing (Via, Route, Record-Route, Proxy-Require, Max-Forwards, and Proxy-Authorization, for example) appear towards the top of the message to facilitate rapid parsing. The relative order of header field rows with the same field name is important. Multiple header field rows with the same field-name MAY be present in a message if and only if the entire field-value for that header field is defined as a comma-separated list (that is, if follows the grammar defined in Section 7.3). It MUST be possible to combine the multiple header field rows into one "field-name: field-value" pair, without changing the semantics of the message, by appending each subsequent field-value to the first, each separated by a comma. The exceptions to this rule are the WWW-Authenticate, Authorization, Proxy-Authenticate, and Proxy-Authorization header fields. Multiple header

field rows with these names MAY be present in a message, but since their grammar does not follow the general form listed in Section 7.3, they MUST NOT be combined into a single header field row.

Implementations MUST be able to process multiple header field rows with the same name in any combination of the single-value-per-line or comma-separated value forms.

The following groups of header field rows are valid and equivalent:

```
Route: <sip:alice@atlanta.com>
Subject: Lunch
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>
Subject: Lunch

Subject: Lunch
Route: <sip:alice@atlanta.com>, <sip:bob@biloxi.com>,
       <sip:carol@chicago.com>
```

Each of the following blocks is valid but not equivalent to the others:

```
Route: <sip:alice@atlanta.com>
Route: <sip:bob@biloxi.com>
Route: <sip:carol@chicago.com>

Route: <sip:bob@biloxi.com>
Route: <sip:alice@atlanta.com>
Route: <sip:carol@chicago.com>

Route: <sip:alice@atlanta.com>,<sip:carol@chicago.com>,
       <sip:bob@biloxi.com>
```

The format of a header field-value is defined per header-name. It will always be either an opaque sequence of TEXT-UTF8 octets, or a combination of whitespace, tokens, separators, and quoted strings. Many existing header fields will adhere to the general form of a value followed by a semi-colon separated sequence of parameter-name, parameter-value pairs:

```
      field-name: field-value *(;parameter-name=parameter-value)
```

Even though an arbitrary number of parameter pairs may be attached to a header field value, any given parameter-name MUST NOT appear more than once.

When comparing header fields, field names are always case-insensitive. Unless otherwise stated in the definition of a particular header field, field values, parameter names, and parameter values are case-insensitive. Tokens are always case-insensitive. Unless specified otherwise, values expressed as quoted strings are case-sensitive. For example,

```
Contact: <sip:alice@atlanta.com>;expires=3600
```

is equivalent to

```
CONTACT: <sip:alice@atlanta.com>;ExPiReS=3600
```

and

```
Content-Disposition: session;handling=optional
```

is equivalent to

```
content-disposition: Session;HANDLING=OPTIONAL
```

The following two header fields are not equivalent:

```
Warning: 370 devnull "Choose a bigger pipe"
Warning: 370 devnull "CHOOSE A BIGGER PIPE"
```

7.3.2 Header Field Classification

Some header fields only make sense in requests or responses. These are called request header fields and response header fields, respectively. If a header field appears in a message not matching its category (such as a request header field in a response), it MUST be ignored. Section 20 defines the classification of each header field.

7.3.3 Compact Form

SIP provides a mechanism to represent common header field names in an abbreviated form. This may be useful when messages would otherwise become too large to be carried on the transport available to it (exceeding the maximum transmission unit (MTU) when using UDP, for example). These compact forms are defined in Section 20. A compact form MAY be substituted for the longer form of a header field name at any time without changing the semantics of the message. A header

field name MAY appear in both long and short forms within the same message. Implementations MUST accept both the long and short forms of each header name.

7.4 Bodies

Requests, including new requests defined in extensions to this specification, MAY contain message bodies unless otherwise noted. The interpretation of the body depends on the request method.

For response messages, the request method and the response status code determine the type and interpretation of any message body. All responses MAY include a body.

7.4.1 Message Body Type

The Internet media type of the message body MUST be given by the Content-Type header field. If the body has undergone any encoding such as compression, then this MUST be indicated by the Content-Encoding header field; otherwise, Content-Encoding MUST be omitted. If applicable, the character set of the message body is indicated as part of the Content-Type header-field value.

The "multipart" MIME type defined in RFC 2046 [11] MAY be used within the body of the message. Implementations that send requests containing multipart message bodies MUST send a session description as a non-multipart message body if the remote implementation requests this through an Accept header field that does not contain multipart.

SIP messages MAY contain binary bodies or body parts. When no explicit charset parameter is provided by the sender, media subtypes of the "text" type are defined to have a default charset value of "UTF-8".

7.4.2 Message Body Length

The body length in bytes is provided by the Content-Length header field. Section 20.14 describes the necessary contents of this header field in detail.

The "chunked" transfer encoding of HTTP/1.1 MUST NOT be used for SIP. (Note: The chunked encoding modifies the body of a message in order to transfer it as a series of chunks, each with its own size indicator.)

7.5 Framing SIP Messages

Unlike HTTP, SIP implementations can use UDP or other unreliable datagram protocols. Each such datagram carries one request or response. See Section 18 on constraints on usage of unreliable transports.

Implementations processing SIP messages over stream-oriented transports MUST ignore any CRLF appearing before the start-line [H4.1].

> The Content-Length header field value is used to locate the end of each SIP message in a stream. It will always be present when SIP messages are sent over stream-oriented transports.

8 General User Agent Behavior

A user agent represents an end system. It contains a user agent client (UAC), which generates requests, and a user agent server (UAS), which responds to them. A UAC is capable of generating a request based on some external stimulus (the user clicking a button, or a signal on a PSTN line) and processing a response. A UAS is capable of receiving a request and generating a response based on user input, external stimulus, the result of a program execution, or some other mechanism.

When a UAC sends a request, the request passes through some number of proxy servers, which forward the request towards the UAS. When the UAS generates a response, the response is forwarded towards the UAC.

UAC and UAS procedures depend strongly on two factors. First, based on whether the request or response is inside or outside of a dialog, and second, based on the method of a request. Dialogs are discussed thoroughly in Section 12; they represent a peer-to-peer relationship between user agents and are established by specific SIP methods, such as INVITE.

In this section, we discuss the method-independent rules for UAC and UAS behavior when processing requests that are outside of a dialog. This includes, of course, the requests which themselves establish a dialog.

Security procedures for requests and responses outside of a dialog are described in Section 26. Specifically, mechanisms exist for the UAS and UAC to mutually authenticate. A limited set of privacy features are also supported through encryption of bodies using S/MIME.

8.1 UAC Behavior

This section covers UAC behavior outside of a dialog.

8.1.1 Generating the Request

A valid SIP request formulated by a UAC MUST, at a minimum, contain the following header fields: To, From, CSeq, Call-ID, Max-Forwards, and Via; all of these header fields are mandatory in all SIP requests. These six header fields are the fundamental building blocks of a SIP message, as they jointly provide for most of the critical message routing services including the addressing of messages, the routing of responses, limiting message propagation, ordering of messages, and the unique identification of transactions. These header fields are in addition to the mandatory request line, which contains the method, Request-URI, and SIP version.

Examples of requests sent outside of a dialog include an INVITE to establish a session (Section 13) and an OPTIONS to query for capabilities (Section 11).

8.1.1.1 Request-URI

The initial Request-URI of the message SHOULD be set to the value of the URI in the To field. One notable exception is the REGISTER method; behavior for setting the Request-URI of REGISTER is given in Section 10. It may also be undesirable for privacy reasons or convenience to set these fields to the same value (especially if the originating UA expects that the Request-URI will be changed during transit).

In some special circumstances, the presence of a pre-existing route set can affect the Request-URI of the message. A pre-existing route set is an ordered set of URIs that identify a chain of servers, to which a UAC will send outgoing requests that are outside of a dialog. Commonly, they are configured on the UA by a user or service provider manually, or through some other non-SIP mechanism. When a provider wishes to configure a UA with an outbound proxy, it is RECOMMENDED that this be done by providing it with a pre-existing route set with a single URI, that of the outbound proxy.

When a pre-existing route set is present, the procedures for populating the Request-URI and Route header field detailed in Section 12.2.1.1 MUST be followed (even though there is no dialog), using the desired Request-URI as the remote target URI.

8.1.1.2 To

The To header field first and foremost specifies the desired "logical" recipient of the request, or the address-of-record of the user or resource that is the target of this request. This may or may not be the ultimate recipient of the request. The To header field MAY contain a SIP or SIPS URI, but it may also make use of other URI schemes (the tel URL (RFC 2806 [9]), for example) when appropriate. All SIP implementations MUST support the SIP URI scheme. Any implementation that supports TLS MUST support the SIPS URI scheme. The To header field allows for a display name.

A UAC may learn how to populate the To header field for a particular request in a number of ways. Usually the user will suggest the To header field through a human interface, perhaps inputting the URI manually or selecting it from some sort of address book. Frequently, the user will not enter a complete URI, but rather a string of digits or letters (for example, "bob"). It is at the discretion of the UA to choose how to interpret this input. Using the string to form the user part of a SIP URI implies that the UA wishes the name to be resolved in the domain to the right-hand side (RHS) of the at-sign in the SIP URI (for instance, sip:bob@example.com). Using the string to form the user part of a SIPS URI implies that the UA wishes to communicate securely, and that the name is to be resolved in the domain to the RHS of the at-sign. The RHS will frequently be the home domain of the requestor, which allows for the home domain to process the outgoing request. This is useful for features like "speed dial" that require interpretation of the user part in the home domain. The tel URL may be used when the UA does not wish to specify the domain that should interpret a telephone number that has been input by the user. Rather, each domain through which the request passes would be given that opportunity. As an example, a user in an airport might log in and send requests through an outbound proxy in the airport. If they enter "411" (this is the phone number for local directory assistance in the United States), that needs to be interpreted and processed by the outbound proxy in the airport, not the user's home domain. In this case, tel:411 would be the right choice.

A request outside of a dialog MUST NOT contain a To tag; the tag in the To field of a request identifies the peer of the dialog. Since no dialog is established, no tag is present.

For further information on the To header field, see Section 20.39. The following is an example of a valid To header field:

```
To: Carol <sip:carol@chicago.com>
```

8.1.1.3 From

The From header field indicates the logical identity of the initiator of the request, possibly the user's address-of-record. Like the To header field, it contains a URI and optionally a display name. It is used by SIP elements to determine which processing rules to apply to a request (for example, automatic call rejection). As such, it is very important that the From URI not contain IP addresses or the FQDN of the host on which the UA is running, since these are not logical names.

The From header field allows for a display name. A UAC SHOULD use the display name "Anonymous", along with a syntactically correct, but otherwise meaningless URI (like sip:thisis@anonymous.invalid), if the identity of the client is to remain hidden.

Usually, the value that populates the From header field in requests generated by a particular UA is pre-provisioned by the user or by the administrators of the user's local domain. If a particular UA is used by multiple users, it might have switchable profiles that include a URI corresponding to the identity of the profiled user. Recipients of requests can authenticate the originator of a request in order to ascertain that they are who their From header field claims they are (see Section 22 for more on authentication).

The From field MUST contain a new "tag" parameter, chosen by the UAC. See Section 19.3 for details on choosing a tag.

For further information on the From header field, see Section 20.20. Examples:

```
From: "Bob" <sips:bob@biloxi.com> ;tag=a48s
From: sip:+12125551212@phone2net.com;tag=887s
From: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

8.1.1.4 Call-ID

The Call-ID header field acts as a unique identifier to group together a series of messages. It MUST be the same for all requests and responses sent by either UA in a dialog. It SHOULD be the same in each registration from a UA.

In a new request created by a UAC outside of any dialog, the Call-ID header field MUST be selected by the UAC as a globally unique identifier over space and time unless overridden by method-specific behavior. All SIP UAs must have a means to guarantee that the Call-ID header fields they produce will not be inadvertently generated by any other UA. Note that when requests are retried after certain

failure responses that solicit an amendment to a request (for example, a challenge for authentication), these retried requests are not considered new requests, and therefore do not need new Call-ID header fields; see Section 8.1.3.5.

Use of cryptographically random identifiers (RFC 1750 [12]) in the generation of Call-IDs is RECOMMENDED. Implementations MAY use the form "localid@host". Call-IDs are case-sensitive and are simply compared byte-by-byte.

> Using cryptographically random identifiers provides some protection against session hijacking and reduces the likelihood of unintentional Call-ID collisions.

No provisioning or human interface is required for the selection of the Call-ID header field value for a request.

For further information on the Call-ID header field, see Section 20.8.

Example:

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@foo.bar.com
```

8.1.1.5 CSeq

The CSeq header field serves as a way to identify and order transactions. It consists of a sequence number and a method. The method MUST match that of the request. For non-REGISTER requests outside of a dialog, the sequence number value is arbitrary. The sequence number value MUST be expressible as a 32-bit unsigned integer and MUST be less than 2**31. As long as it follows the above guidelines, a client may use any mechanism it would like to select CSeq header field values.

Section 12.2.1.1 discusses construction of the CSeq for requests within a dialog.

Example:

```
CSeq: 4711 INVITE
```

8.1.1.6 Max-Forwards

The Max-Forwards header field serves to limit the number of hops a request can transit on the way to its destination. It consists of an integer that is decremented by one at each hop. If the Max-Forwards value reaches 0 before the request reaches its destination, it will be rejected with a 483(Too Many Hops) error response.

A UAC MUST insert a Max-Forwards header field into each request it originates with a value that SHOULD be 70. This number was chosen to be sufficiently large to guarantee that a request would not be dropped in any SIP network when there were no loops, but not so large as to consume proxy resources when a loop does occur. Lower values should be used with caution and only in networks where topologies are known by the UA.

8.1.1.7 Via

The Via header field indicates the transport used for the transaction and identifies the location where the response is to be sent. A Via header field value is added only after the transport that will be used to reach the next hop has been selected (which may involve the usage of the procedures in [4]).

When the UAC creates a request, it MUST insert a Via into that request. The protocol name and protocol version in the header field MUST be SIP and 2.0, respectively. The Via header field value MUST contain a branch parameter. This parameter is used to identify the transaction created by that request. This parameter is used by both the client and the server.

The branch parameter value MUST be unique across space and time for all requests sent by the UA. The exceptions to this rule are CANCEL and ACK for non-2xx responses. As discussed below, a CANCEL request will have the same value of the branch parameter as the request it cancels. As discussed in Section 17.1.1.3, an ACK for a non-2xx response will also have the same branch ID as the INVITE whose response it acknowledges.

> The uniqueness property of the branch ID parameter, to facilitate its use as a transaction ID, was not part of RFC 2543.

The branch ID inserted by an element compliant with this specification MUST always begin with the characters "z9hG4bK". These 7 characters are used as a magic cookie (7 is deemed sufficient to ensure that an older RFC 2543 implementation would not pick such a value), so that servers receiving the request can determine that the branch ID was constructed in the fashion described by this

specification (that is, globally unique). Beyond this requirement, the precise format of the branch token is implementation-defined.

The Via header maddr, ttl, and sent-by components will be set when the request is processed by the transport layer (Section 18).

Via processing for proxies is described in Section 16.6 Item 8 and Section 16.7 Item 3.

8.1.1.8 Contact

The Contact header field provides a SIP or SIPS URI that can be used to contact that specific instance of the UA for subsequent requests. The Contact header field MUST be present and contain exactly one SIP or SIPS URI in any request that can result in the establishment of a dialog. For the methods defined in this specification, that includes only the INVITE request. For these requests, the scope of the Contact is global. That is, the Contact header field value contains the URI at which the UA would like to receive requests, and this URI MUST be valid even if used in subsequent requests outside of any dialogs.

If the Request-URI or top Route header field value contains a SIPS URI, the Contact header field MUST contain a SIPS URI as well.

For further information on the Contact header field, see Section 20.10.

8.1.1.9 Supported and Require

If the UAC supports extensions to SIP that can be applied by the server to the response, the UAC SHOULD include a Supported header field in the request listing the option tags (Section 19.2) for those extensions.

The option tags listed MUST only refer to extensions defined in standards-track RFCs. This is to prevent servers from insisting that clients implement non-standard, vendor-defined features in order to receive service. Extensions defined by experimental and informational RFCs are explicitly excluded from usage with the Supported header field in a request, since they too are often used to document vendor-defined extensions.

If the UAC wishes to insist that a UAS understand an extension that the UAC will apply to the request in order to process the request, it MUST insert a Require header field into the request listing the option tag for that extension. If the UAC wishes to apply an extension to the request and insist that any proxies that are

traversed understand that extension, it MUST insert a Proxy-Require header field into the request listing the option tag for that extension.

As with the Supported header field, the option tags in the Require and Proxy-Require header fields MUST only refer to extensions defined in standards-track RFCs.

8.1.1.10 Additional Message Components

After a new request has been created, and the header fields described above have been properly constructed, any additional optional header fields are added, as are any header fields specific to the method.

SIP requests MAY contain a MIME-encoded message-body. Regardless of the type of body that a request contains, certain header fields must be formulated to characterize the contents of the body. For further information on these header fields, see Sections 20.11 through 20.15.

8.1.2 Sending the Request

The destination for the request is then computed. Unless there is local policy specifying otherwise, the destination MUST be determined by applying the DNS procedures described in [4] as follows. If the first element in the route set indicated a strict router (resulting in forming the request as described in Section 12.2.1.1), the procedures MUST be applied to the Request-URI of the request. Otherwise, the procedures are applied to the first Route header field value in the request (if one exists), or to the request's Request-URI if there is no Route header field present. These procedures yield an ordered set of address, port, and transports to attempt. Independent of which URI is used as input to the procedures of [4], if the Request-URI specifies a SIPS resource, the UAC MUST follow the procedures of [4] as if the input URI were a SIPS URI.

Local policy MAY specify an alternate set of destinations to attempt. If the Request-URI contains a SIPS URI, any alternate destinations MUST be contacted with TLS. Beyond that, there are no restrictions on the alternate destinations if the request contains no Route header field. This provides a simple alternative to a pre-existing route set as a way to specify an outbound proxy. However, that approach for configuring an outbound proxy is NOT RECOMMENDED; a pre-existing route set with a single URI SHOULD be used instead. If the request contains a Route header field, the request SHOULD be sent to the locations derived from its topmost value, but MAY be sent to any server that the UA is certain will honor the Route and Request-URI policies specified in this document (as opposed to those in RFC 2543). In particular, a UAC configured with an outbound proxy SHOULD

attempt to send the request to the location indicated in the first Route header field value instead of adopting the policy of sending all messages to the outbound proxy.

This ensures that outbound proxies that do not add Record-Route header field values will drop out of the path of subsequent requests. It allows endpoints that cannot resolve the first Route URI to delegate that task to an outbound proxy.

The UAC SHOULD follow the procedures defined in [4] for stateful elements, trying each address until a server is contacted. Each try constitutes a new transaction, and therefore each carries a different topmost Via header field value with a new branch parameter. Furthermore, the transport value in the Via header field is set to whatever transport was determined for the target server.

8.1.3 Processing Responses

Responses are first processed by the transport layer and then passed up to the transaction layer. The transaction layer performs its processing and then passes the response up to the TU. The majority of response processing in the TU is method specific. However, there are some general behaviors independent of the method.

8.1.3.1 Transaction Layer Errors

In some cases, the response returned by the transaction layer will not be a SIP message, but rather a transaction layer error. When a timeout error is received from the transaction layer, it MUST be treated as if a 408 (Request Timeout) status code has been received. If a fatal transport error is reported by the transport layer (generally, due to fatal ICMP errors in UDP or connection failures in TCP), the condition MUST be treated as a 503 (Service Unavailable) status code.

8.1.3.2 Unrecognized Responses

A UAC MUST treat any final response it does not recognize as being equivalent to the x00 response code of that class, and MUST be able to process the x00 response code for all classes. For example, if a UAC receives an unrecognized response code of 431, it can safely assume that there was something wrong with its request and treat the response as if it had received a 400 (Bad Request) response code. A UAC MUST treat any provisional response different than 100 that it does not recognize as 183 (Session Progress). A UAC MUST be able to process 100 and 183 responses.

8.1.3.3 Vias

If more than one Via header field value is present in a response, the UAC SHOULD discard the message.

> The presence of additional Via header field values that precede the originator of the request suggests that the message was misrouted or possibly corrupted.

8.1.3.4 Processing 3xx Responses

Upon receipt of a redirection response (for example, a 301 response status code), clients SHOULD use the URI(s) in the Contact header field to formulate one or more new requests based on the redirected request. This process is similar to that of a proxy recursing on a 3xx class response as detailed in Sections 16.5 and 16.6. A client starts with an initial target set containing exactly one URI, the Request-URI of the original request. If a client wishes to formulate new requests based on a 3xx class response to that request, it places the URIs to try into the target set. Subject to the restrictions in this specification, a client can choose which Contact URIs it places into the target set. As with proxy recursion, a client processing 3xx class responses MUST NOT add any given URI to the target set more than once. If the original request had a SIPS URI in the Request-URI, the client MAY choose to recurse to a non-SIPS URI, but SHOULD inform the user of the redirection to an insecure URI.

> Any new request may receive 3xx responses themselves containing the original URI as a contact. Two locations can be configured to redirect to each other. Placing any given URI in the target set only once prevents infinite redirection loops.

As the target set grows, the client MAY generate new requests to the URIs in any order. A common mechanism is to order the set by the "q" parameter value from the Contact header field value. Requests to the URIs MAY be generated serially or in parallel. One approach is to process groups of decreasing q-values serially and process the URIs in each q-value group in parallel. Another is to perform only serial processing in decreasing q-value order, arbitrarily choosing between contacts of equal q-value.

If contacting an address in the list results in a failure, as defined in the next paragraph, the element moves to the next address in the list, until the list is exhausted. If the list is exhausted, then the request has failed.

Failures SHOULD be detected through failure response codes (codes greater than 399); for network errors the client transaction will report any transport layer failures to the transaction user. Note that some response codes (detailed in 8.1.3.5) indicate that the request can be retried; requests that are reattempted should not be considered failures.

When a failure for a particular contact address is received, the client SHOULD try the next contact address. This will involve creating a new client transaction to deliver a new request.

In order to create a request based on a contact address in a 3xx response, a UAC MUST copy the entire URI from the target set into the Request-URI, except for the "method-param" and "header" URI parameters (see Section 19.1.1 for a definition of these parameters). It uses the "header" parameters to create header field values for the new request, overwriting header field values associated with the redirected request in accordance with the guidelines in Section 19.1.5.

Note that in some instances, header fields that have been communicated in the contact address may instead append to existing request header fields in the original redirected request. As a general rule, if the header field can accept a comma-separated list of values, then the new header field value MAY be appended to any existing values in the original redirected request. If the header field does not accept multiple values, the value in the original redirected request MAY be overwritten by the header field value communicated in the contact address. For example, if a contact address is returned with the following value:

sip:user@host?Subject=foo&Call-Info=<http://www.foo.com>

Then any Subject header field in the original redirected request is overwritten, but the HTTP URL is merely appended to any existing Call-Info header field values.

It is RECOMMENDED that the UAC reuse the same To, From, and Call-ID used in the original redirected request, but the UAC MAY also choose to update the Call-ID header field value for new requests, for example.

Finally, once the new request has been constructed, it is sent using a new client transaction, and therefore MUST have a new branch ID in the top Via field as discussed in Section 8.1.1.7.

In all other respects, requests sent upon receipt of a redirect response SHOULD re-use the header fields and bodies of the original request.

In some instances, Contact header field values may be cached at UAC temporarily or permanently depending on the status code received and the presence of an expiration interval; see Sections 21.3.2 and 21.3.3.

8.1.3.5 Processing 4xx Responses

Certain 4xx response codes require specific UA processing, independent of the method.

If a 401 (Unauthorized) or 407 (Proxy Authentication Required) response is received, the UAC SHOULD follow the authorization procedures of Section 22.2 and Section 22.3 to retry the request with credentials.

If a 413 (Request Entity Too Large) response is received (Section 21.4.11), the request contained a body that was longer than the UAS was willing to accept. If possible, the UAC SHOULD retry the request, either omitting the body or using one of a smaller length.

If a 415 (Unsupported Media Type) response is received (Section 21.4.13), the request contained media types not supported by the UAS. The UAC SHOULD retry sending the request, this time only using content with types listed in the Accept header field in the response, with encodings listed in the Accept-Encoding header field in the response, and with languages listed in the Accept-Language in the response.

If a 416 (Unsupported URI Scheme) response is received (Section 21.4.14), the Request-URI used a URI scheme not supported by the server. The client SHOULD retry the request, this time, using a SIP URI.

If a 420 (Bad Extension) response is received (Section 21.4.15), the request contained a Require or Proxy-Require header field listing an option-tag for a feature not supported by a proxy or UAS. The UAC SHOULD retry the request, this time omitting any extensions listed in the Unsupported header field in the response.

In all of the above cases, the request is retried by creating a new request with the appropriate modifications. This new request constitutes a new transaction and SHOULD have the same value of the Call-ID, To, and From of the previous request, but the CSeq should contain a new sequence number that is one higher than the previous.

With other 4xx responses, including those yet to be defined, a retry may or may not be possible depending on the method and the use case.

8.2 UAS Behavior

When a request outside of a dialog is processed by a UAS, there is a set of processing rules that are followed, independent of the method. Section 12 gives guidance on how a UAS can tell whether a request is inside or outside of a dialog.

Note that request processing is atomic. If a request is accepted, all state changes associated with it MUST be performed. If it is rejected, all state changes MUST NOT be performed.

UASs SHOULD process the requests in the order of the steps that follow in this section (that is, starting with authentication, then inspecting the method, the header fields, and so on throughout the remainder of this section).

8.2.1 Method Inspection

Once a request is authenticated (or authentication is skipped), the UAS MUST inspect the method of the request. If the UAS recognizes but does not support the method of a request, it MUST generate a 405 (Method Not Allowed) response. Procedures for generating responses are described in Section 8.2.6. The UAS MUST also add an Allow header field to the 405 (Method Not Allowed) response. The Allow header field MUST list the set of methods supported by the UAS generating the message. The Allow header field is presented in Section 20.5.

If the method is one supported by the server, processing continues.

8.2.2 Header Inspection

If a UAS does not understand a header field in a request (that is, the header field is not defined in this specification or in any supported extension), the server MUST ignore that header field and continue processing the message. A UAS SHOULD ignore any malformed header fields that are not necessary for processing requests.

8.2.2.1 To and Request-URI

The To header field identifies the original recipient of the request designated by the user identified in the From field. The original recipient may or may not be the UAS processing the request, due to call forwarding or other proxy operations. A UAS MAY apply any policy it wishes to determine whether to accept requests when the To

header field is not the identity of the UAS. However, it is RECOMMENDED that a UAS accept requests even if they do not recognize the URI scheme (for example, a tel: URI) in the To header field, or if the To header field does not address a known or current user of this UAS. If, on the other hand, the UAS decides to reject the request, it SHOULD generate a response with a 403 (Forbidden) status code and pass it to the server transaction for transmission.

However, the Request-URI identifies the UAS that is to process the request. If the Request-URI uses a scheme not supported by the UAS, it SHOULD reject the request with a 416 (Unsupported URI Scheme) response. If the Request-URI does not identify an address that the UAS is willing to accept requests for, it SHOULD reject the request with a 404 (Not Found) response. Typically, a UA that uses the REGISTER method to bind its address-of-record to a specific contact address will see requests whose Request-URI equals that contact address. Other potential sources of received Request-URIs include the Contact header fields of requests and responses sent by the UA that establish or refresh dialogs.

8.2.2.2 Merged Requests

If the request has no tag in the To header field, the UAS core MUST check the request against ongoing transactions. If the From tag, Call-ID, and CSeq exactly match those associated with an ongoing transaction, but the request does not match that transaction (based on the matching rules in Section 17.2.3), the UAS core SHOULD generate a 482 (Loop Detected) response and pass it to the server transaction.

> The same request has arrived at the UAS more than once, following different paths, most likely due to forking. The UAS processes the first such request received and responds with a 482 (Loop Detected) to the rest of them.

8.2.2.3 Require

Assuming the UAS decides that it is the proper element to process the request, it examines the Require header field, if present.

The Require header field is used by a UAC to tell a UAS about SIP extensions that the UAC expects the UAS to support in order to process the request properly. Its format is described in Section 20.32. If a UAS does not understand an option-tag listed in a Require header field, it MUST respond by generating a response with status code 420 (Bad Extension). The UAS MUST add an Unsupported header field, and list in it those options it does not understand amongst those in the Require header field of the request.

Note that Require and Proxy-Require MUST NOT be used in a SIP CANCEL request, or in an ACK request sent for a non-2xx response. These header fields MUST be ignored if they are present in these requests.

An ACK request for a 2xx response MUST contain only those Require and Proxy-Require values that were present in the initial request.

Example:

```
UAC->UAS:   INVITE sip:watson@bell-telephone.com SIP/2.0
            Require: 100rel

UAS->UAC:   SIP/2.0 420 Bad Extension
            Unsupported: 100rel
```

This behavior ensures that the client-server interaction will proceed without delay when all options are understood by both sides, and only slow down if options are not understood (as in the example above). For a well-matched client-server pair, the interaction proceeds quickly, saving a round-trip often required by negotiation mechanisms. In addition, it also removes ambiguity when the client requires features that the server does not understand. Some features, such as call handling fields, are only of interest to end systems.

8.2.3 Content Processing

Assuming the UAS understands any extensions required by the client, the UAS examines the body of the message, and the header fields that describe it. If there are any bodies whose type (indicated by the Content-Type), language (indicated by the Content-Language) or encoding (indicated by the Content-Encoding) are not understood, and that body part is not optional (as indicated by the Content-Disposition header field), the UAS MUST reject the request with a 415 (Unsupported Media Type) response. The response MUST contain an Accept header field listing the types of all bodies it understands, in the event the request contained bodies of types not supported by the UAS. If the request contained content encodings not understood by the UAS, the response MUST contain an Accept-Encoding header field listing the encodings understood by the UAS. If the request contained content with languages not understood by the UAS, the response MUST contain an Accept-Language header field indicating the languages understood by the UAS. Beyond these checks, body handling depends on the method and type. For further information on the processing of content-specific header fields, see Section 7.4 as well as Section 20.11 through 20.15.

8.2.4 Applying Extensions

A UAS that wishes to apply some extension when generating the response MUST NOT do so unless support for that extension is indicated in the Supported header field in the request. If the desired extension is not supported, the server SHOULD rely only on baseline SIP and any other extensions supported by the client. In rare circumstances, where the server cannot process the request without the extension, the server MAY send a 421 (Extension Required) response. This response indicates that the proper response cannot be generated without support of a specific extension. The needed extension(s) MUST be included in a Require header field in the response. This behavior is NOT RECOMMENDED, as it will generally break interoperability.

Any extensions applied to a non-421 response MUST be listed in a Require header field included in the response. Of course, the server MUST NOT apply extensions not listed in the Supported header field in the request. As a result of this, the Require header field in a response will only ever contain option tags defined in standards-track RFCs.

8.2.5 Processing the Request

Assuming all of the checks in the previous subsections are passed, the UAS processing becomes method-specific. Section 10 covers the REGISTER request, Section 11 covers the OPTIONS request, Section 13 covers the INVITE request, and Section 15 covers the BYE request.

8.2.6 Generating the Response

When a UAS wishes to construct a response to a request, it follows the general procedures detailed in the following subsections. Additional behaviors specific to the response code in question, which are not detailed in this section, may also be required.

Once all procedures associated with the creation of a response have been completed, the UAS hands the response back to the server transaction from which it received the request.

8.2.6.1 Sending a Provisional Response

One largely non-method-specific guideline for the generation of responses is that UASs SHOULD NOT issue a provisional response for a non-INVITE request. Rather, UASs SHOULD generate a final response to a non-INVITE request as soon as possible.

When a 100 (Trying) response is generated, any Timestamp header field present in the request MUST be copied into this 100 (Trying) response. If there is a delay in generating the response, the UAS SHOULD add a delay value into the Timestamp value in the response. This value MUST contain the difference between the time of sending of the response and receipt of the request, measured in seconds.

8.2.6.2 Headers and Tags

The From field of the response MUST equal the From header field of the request. The Call-ID header field of the response MUST equal the Call-ID header field of the request. The CSeq header field of the response MUST equal the CSeq field of the request. The Via header field values in the response MUST equal the Via header field values in the request and MUST maintain the same ordering.

If a request contained a To tag in the request, the To header field in the response MUST equal that of the request. However, if the To header field in the request did not contain a tag, the URI in the To header field in the response MUST equal the URI in the To header field; additionally, the UAS MUST add a tag to the To header field in the response (with the exception of the 100 (Trying) response, in which a tag MAY be present). This serves to identify the UAS that is responding, possibly resulting in a component of a dialog ID. The same tag MUST be used for all responses to that request, both final and provisional (again excepting the 100 (Trying)). Procedures for the generation of tags are defined in Section 19.3.

8.2.7 Stateless UAS Behavior

A stateless UAS is a UAS that does not maintain transaction state. It replies to requests normally, but discards any state that would ordinarily be retained by a UAS after a response has been sent. If a stateless UAS receives a retransmission of a request, it regenerates the response and resends it, just as if it were replying to the first instance of the request. A UAS cannot be stateless unless the request processing for that method would always result in the same response if the requests are identical. This rules out stateless registrars, for example. Stateless UASs do not use a transaction layer; they receive requests directly from the transport layer and send responses directly to the transport layer.

The stateless UAS role is needed primarily to handle unauthenticated requests for which a challenge response is issued. If unauthenticated requests were handled statefully, then malicious floods of unauthenticated requests could create massive amounts of

transaction state that might slow or completely halt call processing in a UAS, effectively creating a denial of service condition; for more information see Section 26.1.5.

The most important behaviors of a stateless UAS are the following:

- A stateless UAS MUST NOT send provisional (1xx) responses.
- A stateless UAS MUST NOT retransmit responses.
- A stateless UAS MUST ignore ACK requests.
- A stateless UAS MUST ignore CANCEL requests.
- To header tags MUST be generated for responses in a stateless manner - in a manner that will generate the same tag for the same request consistently. For information on tag construction see Section 19.3.

In all other respects, a stateless UAS behaves in the same manner as a stateful UAS. A UAS can operate in either a stateful or stateless mode for each new request.

8.3 Redirect Servers

In some architectures it may be desirable to reduce the processing load on proxy servers that are responsible for routing requests, and improve signaling path robustness, by relying on redirection.

Redirection allows servers to push routing information for a request back in a response to the client, thereby taking themselves out of the loop of further messaging for this transaction while still aiding in locating the target of the request. When the originator of the request receives the redirection, it will send a new request based on the URI(s) it has received. By propagating URIs from the core of the network to its edges, redirection allows for considerable network scalability.

A redirect server is logically constituted of a server transaction layer and a transaction user that has access to a location service of some kind (see Section 10 for more on registrars and location services). This location service is effectively a database containing mappings between a single URI and a set of one or more alternative locations at which the target of that URI can be found.

A redirect server does not issue any SIP requests of its own. After receiving a request other than CANCEL, the server either refuses the request or gathers the list of alternative locations from the

location service and returns a final response of class 3xx. For well-formed CANCEL requests, it SHOULD return a 2xx response. This response ends the SIP transaction. The redirect server maintains transaction state for an entire SIP transaction. It is the responsibility of clients to detect forwarding loops between redirect servers.

When a redirect server returns a 3xx response to a request, it populates the list of (one or more) alternative locations into the Contact header field. An "expires" parameter to the Contact header field values may also be supplied to indicate the lifetime of the Contact data.

The Contact header field contains URIs giving the new locations or user names to try, or may simply specify additional transport parameters. A 301 (Moved Permanently) or 302 (Moved Temporarily) response may also give the same location and username that was targeted by the initial request but specify additional transport parameters such as a different server or multicast address to try, or a change of SIP transport from UDP to TCP or vice versa.

However, redirect servers MUST NOT redirect a request to a URI equal to the one in the Request-URI; instead, provided that the URI does not point to itself, the server MAY proxy the request to the destination URI, or MAY reject it with a 404.

> If a client is using an outbound proxy, and that proxy actually redirects requests, a potential arises for infinite redirection loops.

Note that a Contact header field value MAY also refer to a different resource than the one originally called. For example, a SIP call connected to PSTN gateway may need to deliver a special informational announcement such as "The number you have dialed has been changed."

A Contact response header field can contain any suitable URI indicating where the called party can be reached, not limited to SIP URIs. For example, it could contain URIs for phones, fax, or irc (if they were defined) or a mailto: (RFC 2368 [32]) URL. Section 26.4.4 discusses implications and limitations of redirecting a SIPS URI to a non-SIPS URI.

The "expires" parameter of a Contact header field value indicates how long the URI is valid. The value of the parameter is a number indicating seconds. If this parameter is not provided, the value of the Expires header field determines how long the URI is valid. Malformed values SHOULD be treated as equivalent to 3600.

This provides a modest level of backwards compatibility with RFC 2543, which allowed absolute times in this header field. If an absolute time is received, it will be treated as malformed, and then default to 3600.

Redirect servers MUST ignore features that are not understood (including unrecognized header fields, any unknown option tags in Require, or even method names) and proceed with the redirection of the request in question.

9 Canceling a Request

The previous section has discussed general UA behavior for generating requests and processing responses for requests of all methods. In this section, we discuss a general purpose method, called CANCEL.

The CANCEL request, as the name implies, is used to cancel a previous request sent by a client. Specifically, it asks the UAS to cease processing the request and to generate an error response to that request. CANCEL has no effect on a request to which a UAS has already given a final response. Because of this, it is most useful to CANCEL requests to which it can take a server long time to respond. For this reason, CANCEL is best for INVITE requests, which can take a long time to generate a response. In that usage, a UAS that receives a CANCEL request for an INVITE, but has not yet sent a final response, would "stop ringing", and then respond to the INVITE with a specific error response (a 487).

CANCEL requests can be constructed and sent by both proxies and user agent clients. Section 15 discusses under what conditions a UAC would CANCEL an INVITE request, and Section 16.10 discusses proxy usage of CANCEL.

A stateful proxy responds to a CANCEL, rather than simply forwarding a response it would receive from a downstream element. For that reason, CANCEL is referred to as a "hop-by-hop" request, since it is responded to at each stateful proxy hop.

9.1 Client Behavior

A CANCEL request SHOULD NOT be sent to cancel a request other than INVITE.

Since requests other than INVITE are responded to immediately, sending a CANCEL for a non-INVITE request would always create a race condition.

The following procedures are used to construct a CANCEL request. The Request-URI, Call-ID, To, the numeric part of CSeq, and From header fields in the CANCEL request MUST be identical to those in the request being cancelled, including tags. A CANCEL constructed by a client MUST have only a single Via header field value matching the top Via value in the request being cancelled. Using the same values for these header fields allows the CANCEL to be matched with the request it cancels (Section 9.2 indicates how such matching occurs). However, the method part of the CSeq header field MUST have a value of CANCEL. This allows it to be identified and processed as a transaction in its own right (See Section 17).

If the request being cancelled contains a Route header field, the CANCEL request MUST include that Route header field's values.

> This is needed so that stateless proxies are able to route CANCEL requests properly.

The CANCEL request MUST NOT contain any Require or Proxy-Require header fields.

Once the CANCEL is constructed, the client SHOULD check whether it has received any response (provisional or final) for the request being cancelled (herein referred to as the "original request").

If no provisional response has been received, the CANCEL request MUST NOT be sent; rather, the client MUST wait for the arrival of a provisional response before sending the request. If the original request has generated a final response, the CANCEL SHOULD NOT be sent, as it is an effective no-op, since CANCEL has no effect on requests that have already generated a final response. When the client decides to send the CANCEL, it creates a client transaction for the CANCEL and passes it the CANCEL request along with the destination address, port, and transport. The destination address, port, and transport for the CANCEL MUST be identical to those used to send the original request.

> If it was allowed to send the CANCEL before receiving a response for the previous request, the server could receive the CANCEL before the original request.

Note that both the transaction corresponding to the original request and the CANCEL transaction will complete independently. However, a UAC canceling a request cannot rely on receiving a 487 (Request Terminated) response for the original request, as an RFC 2543-compliant UAS will not generate such a response. If there is no final response for the original request in 64*T1 seconds (T1 is

defined in Section 17.1.1.1), the client SHOULD then consider the original transaction cancelled and SHOULD destroy the client transaction handling the original request.

9.2 Server Behavior

The CANCEL method requests that the TU at the server side cancel a pending transaction. The TU determines the transaction to be cancelled by taking the CANCEL request, and then assuming that the request method is anything but CANCEL or ACK and applying the transaction matching procedures of Section 17.2.3. The matching transaction is the one to be cancelled.

The processing of a CANCEL request at a server depends on the type of server. A stateless proxy will forward it, a stateful proxy might respond to it and generate some CANCEL requests of its own, and a UAS will respond to it. See Section 16.10 for proxy treatment of CANCEL.

A UAS first processes the CANCEL request according to the general UAS processing described in Section 8.2. However, since CANCEL requests are hop-by-hop and cannot be resubmitted, they cannot be challenged by the server in order to get proper credentials in an Authorization header field. Note also that CANCEL requests do not contain a Require header field.

If the UAS did not find a matching transaction for the CANCEL according to the procedure above, it SHOULD respond to the CANCEL with a 481 (Call Leg/Transaction Does Not Exist). If the transaction for the original request still exists, the behavior of the UAS on receiving a CANCEL request depends on whether it has already sent a final response for the original request. If it has, the CANCEL request has no effect on the processing of the original request, no effect on any session state, and no effect on the responses generated for the original request. If the UAS has not issued a final response for the original request, its behavior depends on the method of the original request. If the original request was an INVITE, the UAS SHOULD immediately respond to the INVITE with a 487 (Request Terminated). A CANCEL request has no impact on the processing of transactions with any other method defined in this specification.

Regardless of the method of the original request, as long as the CANCEL matched an existing transaction, the UAS answers the CANCEL request itself with a 200 (OK) response. This response is constructed following the procedures described in Section 8.2.6 noting that the To tag of the response to the CANCEL and the To tag in the response to the original request SHOULD be the same. The response to CANCEL is passed to the server transaction for transmission.

10 Registrations

10.1 Overview

SIP offers a discovery capability. If a user wants to initiate a session with another user, SIP must discover the current host(s) at which the destination user is reachable. This discovery process is frequently accomplished by SIP network elements such as proxy servers and redirect servers which are responsible for receiving a request, determining where to send it based on knowledge of the location of the user, and then sending it there. To do this, SIP network elements consult an abstract service known as a location service, which provides address bindings for a particular domain. These address bindings map an incoming SIP or SIPS URI, sip:bob@biloxi.com, for example, to one or more URIs that are somehow "closer" to the desired user, sip:bob@engineering.biloxi.com, for example. Ultimately, a proxy will consult a location service that maps a received URI to the user agent(s) at which the desired recipient is currently residing.

Registration creates bindings in a location service for a particular domain that associates an address-of-record URI with one or more contact addresses. Thus, when a proxy for that domain receives a request whose Request-URI matches the address-of-record, the proxy will forward the request to the contact addresses registered to that address-of-record. Generally, it only makes sense to register an address-of-record at a domain's location service when requests for that address-of-record would be routed to that domain. In most cases, this means that the domain of the registration will need to match the domain in the URI of the address-of-record.

There are many ways by which the contents of the location service can be established. One way is administratively. In the above example, Bob is known to be a member of the engineering department through access to a corporate database. However, SIP provides a mechanism for a UA to create a binding explicitly. This mechanism is known as registration.

Registration entails sending a REGISTER request to a special type of UAS known as a registrar. A registrar acts as the front end to the location service for a domain, reading and writing mappings based on the contents of REGISTER requests. This location service is then typically consulted by a proxy server that is responsible for routing requests for that domain.

An illustration of the overall registration process is given in Figure 2. Note that the registrar and proxy server are logical roles that can be played by a single device in a network; for purposes of

clarity the two are separated in this illustration. Also note that UAs may send requests through a proxy server in order to reach a registrar if the two are separate elements.

SIP does not mandate a particular mechanism for implementing the location service. The only requirement is that a registrar for some domain MUST be able to read and write data to the location service, and a proxy or a redirect server for that domain MUST be capable of reading that same data. A registrar MAY be co-located with a particular SIP proxy server for the same domain.

10.2 Constructing the REGISTER Request

REGISTER requests add, remove, and query bindings. A REGISTER request can add a new binding between an address-of-record and one or more contact addresses. Registration on behalf of a particular address-of-record can be performed by a suitably authorized third party. A client can also remove previous bindings or query to determine which bindings are currently in place for an address-of-record.

Except as noted, the construction of the REGISTER request and the behavior of clients sending a REGISTER request is identical to the general UAC behavior described in Section 8.1 and Section 17.1.

A REGISTER request does not establish a dialog. A UAC MAY include a Route header field in a REGISTER request based on a pre-existing route set as described in Section 8.1. The Record-Route header field has no meaning in REGISTER requests or responses, and MUST be ignored if present. In particular, the UAC MUST NOT create a new route set based on the presence or absence of a Record-Route header field in any response to a REGISTER request.

The following header fields, except Contact, MUST be included in a REGISTER request. A Contact header field MAY be included:

Request-URI: The Request-URI names the domain of the location service for which the registration is meant (for example, "sip:chicago.com"). The "userinfo" and "@" components of the SIP URI MUST NOT be present.

To: The To header field contains the address of record whose registration is to be created, queried, or modified. The To header field and the Request-URI field typically differ, as the former contains a user name. This address-of-record MUST be a SIP URI or SIPS URI.

From: The From header field contains the address-of-record of the person responsible for the registration. The value is the same as the To header field unless the request is a third-party registration.

Call-ID: All registrations from a UAC SHOULD use the same Call-ID header field value for registrations sent to a particular registrar.

If the same client were to use different Call-ID values, a registrar could not detect whether a delayed REGISTER request might have arrived out of order.

CSeq: The CSeq value guarantees proper ordering of REGISTER requests. A UA MUST increment the CSeq value by one for each REGISTER request with the same Call-ID.

Contact: REGISTER requests MAY contain a Contact header field with zero or more values containing address bindings.

UAs MUST NOT send a new registration (that is, containing new Contact header field values, as opposed to a retransmission) until they have received a final response from the registrar for the previous one or the previous REGISTER request has timed out.

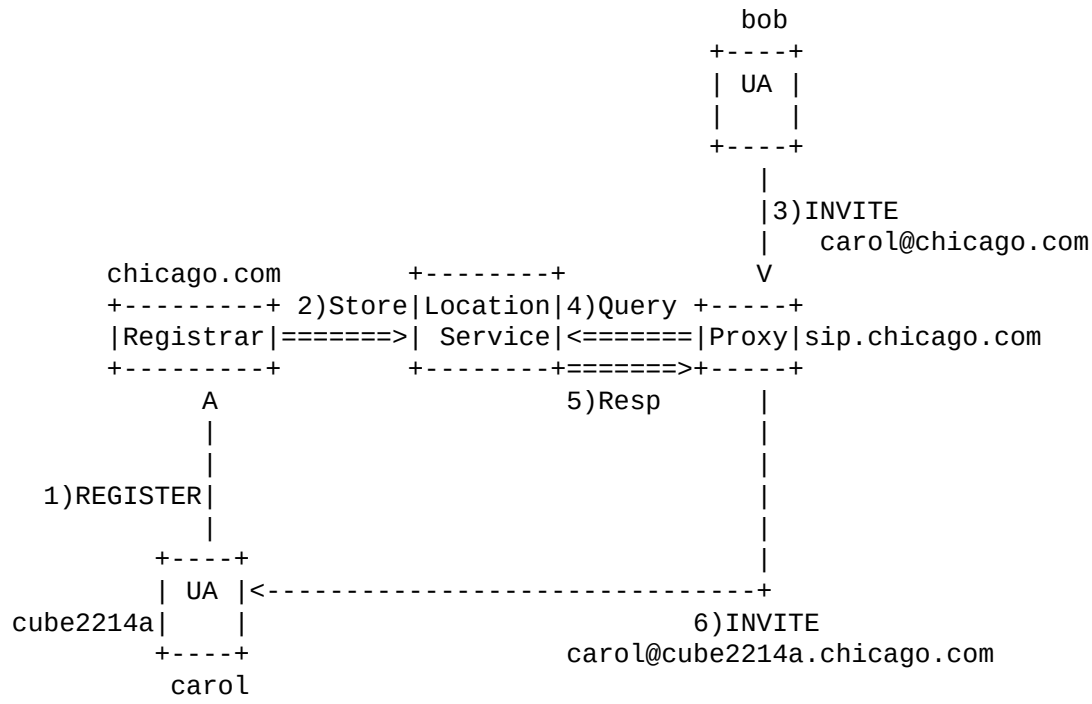


Figure 2: REGISTER example

The following Contact header parameters have a special meaning in REGISTER requests:

action: The "action" parameter from RFC 2543 has been deprecated. UACs SHOULD NOT use the "action" parameter.

expires: The "expires" parameter indicates how long the UA would like the binding to be valid. The value is a number indicating seconds. If this parameter is not provided, the value of the Expires header field is used instead. Implementations MAY treat values larger than 2**32-1 (4294967295 seconds or 136 years) as equivalent to 2**32-1. Malformed values SHOULD be treated as equivalent to 3600.

10.2.1 Adding Bindings

The REGISTER request sent to a registrar includes the contact address(es) to which SIP requests for the address-of-record should be forwarded. The address-of-record is included in the To header field of the REGISTER request.

The Contact header field values of the request typically consist of SIP or SIPS URIs that identify particular SIP endpoints (for example, "sip:carol@cube2214a.chicago.com"), but they MAY use any URI scheme. A SIP UA can choose to register telephone numbers (with the tel URL, RFC 2806 [9]) or email addresses (with a mailto URL, RFC 2368 [32]) as Contacts for an address-of-record, for example.

For example, Carol, with address-of-record "sip:carol@chicago.com", would register with the SIP registrar of the domain chicago.com. Her registrations would then be used by a proxy server in the chicago.com domain to route requests for Carol's address-of-record to her SIP endpoint.

Once a client has established bindings at a registrar, it MAY send subsequent registrations containing new bindings or modifications to existing bindings as necessary. The 2xx response to the REGISTER request will contain, in a Contact header field, a complete list of bindings that have been registered for this address-of-record at this registrar.

If the address-of-record in the To header field of a REGISTER request is a SIPS URI, then any Contact header field values in the request SHOULD also be SIPS URIs. Clients should only register non-SIPS URIs under a SIPS address-of-record when the security of the resource represented by the contact address is guaranteed by other means. This may be applicable to URIs that invoke protocols other than SIP, or SIP devices secured by protocols other than TLS.

Registrations do not need to update all bindings. Typically, a UA only updates its own contact addresses.

10.2.1.1 Setting the Expiration Interval of Contact Addresses

When a client sends a REGISTER request, it MAY suggest an expiration interval that indicates how long the client would like the registration to be valid. (As described in Section 10.3, the registrar selects the actual time interval based on its local policy.)

There are two ways in which a client can suggest an expiration interval for a binding: through an Expires header field or an "expires" Contact header parameter. The latter allows expiration intervals to be suggested on a per-binding basis when more than one binding is given in a single REGISTER request, whereas the former suggests an expiration interval for all Contact header field values that do not contain the "expires" parameter.

If neither mechanism for expressing a suggested expiration time is present in a REGISTER, the client is indicating its desire for the server to choose.

10.2.1.2 Preferences among Contact Addresses

If more than one Contact is sent in a REGISTER request, the registering UA intends to associate all of the URIs in these Contact header field values with the address-of-record present in the To field. This list can be prioritized with the "q" parameter in the Contact header field. The "q" parameter indicates a relative preference for the particular Contact header field value compared to other bindings for this address-of-record. Section 16.6 describes how a proxy server uses this preference indication.

10.2.2 Removing Bindings

Registrations are soft state and expire unless refreshed, but can also be explicitly removed. A client can attempt to influence the expiration interval selected by the registrar as described in Section 10.2.1. A UA requests the immediate removal of a binding by specifying an expiration interval of "0" for that contact address in a REGISTER request. UAs SHOULD support this mechanism so that bindings can be removed before their expiration interval has passed.

The REGISTER-specific Contact header field value of "*" applies to all registrations, but it MUST NOT be used unless the Expires header field is present with a value of "0".

> Use of the "*" Contact header field value allows a registering UA to remove all bindings associated with an address-of-record without knowing their precise values.

10.2.3 Fetching Bindings

A success response to any REGISTER request contains the complete list of existing bindings, regardless of whether the request contained a Contact header field. If no Contact header field is present in a REGISTER request, the list of bindings is left unchanged.

10.2.4 Refreshing Bindings

Each UA is responsible for refreshing the bindings that it has previously established. A UA SHOULD NOT refresh bindings set up by other UAs.

The 200 (OK) response from the registrar contains a list of Contact fields enumerating all current bindings. The UA compares each contact address to see if it created the contact address, using comparison rules in Section 19.1.4. If so, it updates the expiration time interval according to the expires parameter or, if absent, the Expires field value. The UA then issues a REGISTER request for each of its bindings before the expiration interval has elapsed. It MAY combine several updates into one REGISTER request.

A UA SHOULD use the same Call-ID for all registrations during a single boot cycle. Registration refreshes SHOULD be sent to the same network address as the original registration, unless redirected.

10.2.5 Setting the Internal Clock

If the response for a REGISTER request contains a Date header field, the client MAY use this header field to learn the current time in order to set any internal clocks.

10.2.6 Discovering a Registrar

UAs can use three ways to determine the address to which to send registrations: by configuration, using the address-of-record, and multicast. A UA can be configured, in ways beyond the scope of this specification, with a registrar address. If there is no configured registrar address, the UA SHOULD use the host part of the address-of-record as the Request-URI and address the request there, using the normal SIP server location mechanisms [4]. For example, the UA for the user "sip:carol@chicago.com" addresses the REGISTER request to "sip:chicago.com".

Finally, a UA can be configured to use multicast. Multicast registrations are addressed to the well-known "all SIP servers" multicast address "sip.mcast.net" (224.0.1.75 for IPv4). No well-known IPv6 multicast address has been allocated; such an allocation will be documented separately when needed. SIP UAs MAY listen to that address and use it to become aware of the location of other local users (see [33]); however, they do not respond to the request.

> Multicast registration may be inappropriate in some environments, for example, if multiple businesses share the same local area network.

10.2.7 Transmitting a Request

Once the REGISTER method has been constructed, and the destination of the message identified, UACs follow the procedures described in Section 8.1.2 to hand off the REGISTER to the transaction layer.

If the transaction layer returns a timeout error because the REGISTER yielded no response, the UAC SHOULD NOT immediately re-attempt a registration to the same registrar.

An immediate re-attempt is likely to also timeout. Waiting some reasonable time interval for the conditions causing the timeout to be corrected reduces unnecessary load on the network. No specific interval is mandated.

10.2.8 Error Responses

If a UA receives a 423 (Interval Too Brief) response, it MAY retry the registration after making the expiration interval of all contact addresses in the REGISTER request equal to or greater than the expiration interval within the Min-Expires header field of the 423 (Interval Too Brief) response.

10.3 Processing REGISTER Requests

A registrar is a UAS that responds to REGISTER requests and maintains a list of bindings that are accessible to proxy servers and redirect servers within its administrative domain. A registrar handles requests according to Section 8.2 and Section 17.2, but it accepts only REGISTER requests. A registrar MUST not generate 6xx responses.

A registrar MAY redirect REGISTER requests as appropriate. One common usage would be for a registrar listening on a multicast interface to redirect multicast REGISTER requests to its own unicast interface with a 302 (Moved Temporarily) response.

Registrars MUST ignore the Record-Route header field if it is included in a REGISTER request. Registrars MUST NOT include a Record-Route header field in any response to a REGISTER request.

A registrar might receive a request that traversed a proxy which treats REGISTER as an unknown request and which added a Record-Route header field value.

A registrar has to know (for example, through configuration) the set of domain(s) for which it maintains bindings. REGISTER requests MUST be processed by a registrar in the order that they are received. REGISTER requests MUST also be processed atomically, meaning that a particular REGISTER request is either processed completely or not at all. Each REGISTER message MUST be processed independently of any other registration or binding changes.

When receiving a REGISTER request, a registrar follows these steps:

1. The registrar inspects the Request-URI to determine whether it has access to bindings for the domain identified in the Request-URI. If not, and if the server also acts as a proxy server, the server SHOULD forward the request to the addressed domain, following the general behavior for proxying messages described in Section 16.

2. To guarantee that the registrar supports any necessary extensions, the registrar MUST process the Require header field values as described for UASs in Section 8.2.2.

3. A registrar SHOULD authenticate the UAC. Mechanisms for the authentication of SIP user agents are described in Section 22. Registration behavior in no way overrides the generic authentication framework for SIP. If no authentication mechanism is available, the registrar MAY take the From address as the asserted identity of the originator of the request.

4. The registrar SHOULD determine if the authenticated user is authorized to modify registrations for this address-of-record. For example, a registrar might consult an authorization database that maps user names to a list of addresses-of-record for which that user has authorization to modify bindings. If the authenticated user is not authorized to modify bindings, the registrar MUST return a 403 (Forbidden) and skip the remaining steps.

   In architectures that support third-party registration, one entity may be responsible for updating the registrations associated with multiple addresses-of-record.

5. The registrar extracts the address-of-record from the To header field of the request. If the address-of-record is not valid for the domain in the Request-URI, the registrar MUST send a 404 (Not Found) response and skip the remaining steps. The URI MUST then be converted to a canonical form. To do that, all URI parameters MUST be removed (including the user-param), and any escaped characters MUST be converted to their unescaped form. The result serves as an index into the list of bindings.

6. The registrar checks whether the request contains the Contact header field. If not, it skips to the last step. If the Contact header field is present, the registrar checks if there is one Contact field value that contains the special value "*" and an Expires field. If the request has additional Contact fields or an expiration time other than zero, the request is invalid, and the server MUST return a 400 (Invalid Request) and skip the remaining steps. If not, the registrar checks whether the Call-ID agrees with the value stored for each binding. If not, it MUST remove the binding. If it does agree, it MUST remove the binding only if the CSeq in the request is higher than the value stored for that binding. Otherwise, the update MUST be aborted and the request fails.

7. The registrar now processes each contact address in the Contact header field in turn. For each address, it determines the expiration interval as follows:

   - If the field value has an "expires" parameter, that value MUST be taken as the requested expiration.

   - If there is no such parameter, but the request has an Expires header field, that value MUST be taken as the requested expiration.

   - If there is neither, a locally-configured default value MUST be taken as the requested expiration.

   The registrar MAY choose an expiration less than the requested expiration interval. If and only if the requested expiration interval is greater than zero AND smaller than one hour AND less than a registrar-configured minimum, the registrar MAY reject the registration with a response of 423 (Interval Too Brief). This response MUST contain a Min-Expires header field that states the minimum expiration interval the registrar is willing to honor. It then skips the remaining steps.

   Allowing the registrar to set the registration interval protects it against excessively frequent registration refreshes while limiting the state that it needs to maintain and decreasing the likelihood of registrations going stale. The expiration interval of a registration is frequently used in the creation of services. An example is a follow-me service, where the user may only be available at a terminal for a brief period. Therefore, registrars should accept brief registrations; a request should only be rejected if the interval is so short that the refreshes would degrade registrar performance.

For each address, the registrar then searches the list of current bindings using the URI comparison rules. If the binding does not exist, it is tentatively added. If the binding does exist, the registrar checks the Call-ID value. If the Call-ID value in the existing binding differs from the Call-ID value in the request, the binding MUST be removed if the expiration time is zero and updated otherwise. If they are the same, the registrar compares the CSeq value. If the value is higher than that of the existing binding, it MUST update or remove the binding as above. If not, the update MUST be aborted and the request fails.

This algorithm ensures that out-of-order requests from the same UA are ignored.

Each binding record records the Call-ID and CSeq values from the request.

The binding updates MUST be committed (that is, made visible to the proxy or redirect server) if and only if all binding updates and additions succeed. If any one of them fails (for example, because the back-end database commit failed), the request MUST fail with a 500 (Server Error) response and all tentative binding updates MUST be removed.

8. The registrar returns a 200 (OK) response. The response MUST contain Contact header field values enumerating all current bindings. Each Contact value MUST feature an "expires" parameter indicating its expiration interval chosen by the registrar. The response SHOULD include a Date header field.

11 Querying for Capabilities

The SIP method OPTIONS allows a UA to query another UA or a proxy server as to its capabilities. This allows a client to discover information about the supported methods, content types, extensions, codecs, etc. without "ringing" the other party. For example, before a client inserts a Require header field into an INVITE listing an option that it is not certain the destination UAS supports, the client can query the destination UAS with an OPTIONS to see if this option is returned in a Supported header field. All UAs MUST support the OPTIONS method.

The target of the OPTIONS request is identified by the Request-URI, which could identify another UA or a SIP server. If the OPTIONS is addressed to a proxy server, the Request-URI is set without a user part, similar to the way a Request-URI is set for a REGISTER request.

Alternatively, a server receiving an OPTIONS request with a Max-Forwards header field value of 0 MAY respond to the request regardless of the Request-URI.

This behavior is common with HTTP/1.1. This behavior can be used as a "traceroute" functionality to check the capabilities of individual hop servers by sending a series of OPTIONS requests with incremented Max-Forwards values.

As is the case for general UA behavior, the transaction layer can return a timeout error if the OPTIONS yields no response. This may indicate that the target is unreachable and hence unavailable.

An OPTIONS request MAY be sent as part of an established dialog to query the peer on capabilities that may be utilized later in the dialog.

11.1 Construction of OPTIONS Request

An OPTIONS request is constructed using the standard rules for a SIP request as discussed in Section 8.1.1.

A Contact header field MAY be present in an OPTIONS.

An Accept header field SHOULD be included to indicate the type of message body the UAC wishes to receive in the response. Typically, this is set to a format that is used to describe the media capabilities of a UA, such as SDP (application/sdp).

The response to an OPTIONS request is assumed to be scoped to the Request-URI in the original request. However, only when an OPTIONS is sent as part of an established dialog is it guaranteed that future requests will be received by the server that generated the OPTIONS response.

Example OPTIONS request:

```
OPTIONS sip:carol@chicago.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
Max-Forwards: 70
To: <sip:carol@chicago.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:alice@pc33.atlanta.com>
Accept: application/sdp
Content-Length: 0
```

11.2 Processing of OPTIONS Request

The response to an OPTIONS is constructed using the standard rules for a SIP response as discussed in Section 8.2.6. The response code chosen MUST be the same that would have been chosen had the request been an INVITE. That is, a 200 (OK) would be returned if the UAS is ready to accept a call, a 486 (Busy Here) would be returned if the UAS is busy, etc. This allows an OPTIONS request to be used to determine the basic state of a UAS, which can be an indication of whether the UAS will accept an INVITE request.

An OPTIONS request received within a dialog generates a 200 (OK) response that is identical to one constructed outside a dialog and does not have any impact on the dialog.

This use of OPTIONS has limitations due to the differences in proxy handling of OPTIONS and INVITE requests. While a forked INVITE can result in multiple 200 (OK) responses being returned, a forked OPTIONS will only result in a single 200 (OK) response, since it is treated by proxies using the non-INVITE handling. See Section 16.7 for the normative details.

If the response to an OPTIONS is generated by a proxy server, the proxy returns a 200 (OK), listing the capabilities of the server. The response does not contain a message body.

Allow, Accept, Accept-Encoding, Accept-Language, and Supported header fields SHOULD be present in a 200 (OK) response to an OPTIONS request. If the response is generated by a proxy, the Allow header field SHOULD be omitted as it is ambiguous since a proxy is method agnostic. Contact header fields MAY be present in a 200 (OK) response and have the same semantics as in a 3xx response. That is, they may list a set of alternative names and methods of reaching the user. A Warning header field MAY be present.

A message body MAY be sent, the type of which is determined by the Accept header field in the OPTIONS request (application/sdp is the default if the Accept header field is not present). If the types include one that can describe media capabilities, the UAS SHOULD include a body in the response for that purpose. Details on the construction of such a body in the case of application/sdp are described in [13].

Example OPTIONS response generated by a UAS (corresponding to the request in Section 11.1):

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKhjhs8ass877
 ;received=192.0.2.4
To: <sip:carol@chicago.com>;tag=93810874
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 63104 OPTIONS
Contact: <sip:carol@chicago.com>
Contact: <mailto:carol@chicago.com>
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
Accept: application/sdp
Accept-Encoding: gzip
Accept-Language: en
Supported: foo
Content-Type: application/sdp
Content-Length: 274

(SDP not shown)
```

12 Dialogs

A key concept for a user agent is that of a dialog. A dialog represents a peer-to-peer SIP relationship between two user agents that persists for some time. The dialog facilitates sequencing of messages between the user agents and proper routing of requests between both of them. The dialog represents a context in which to interpret SIP messages. Section 8 discussed method independent UA processing for requests and responses outside of a dialog. This section discusses how those requests and responses are used to construct a dialog, and then how subsequent requests and responses are sent within a dialog.

A dialog is identified at each UA with a dialog ID, which consists of a Call-ID value, a local tag and a remote tag. The dialog ID at each UA involved in the dialog is not the same. Specifically, the local tag at one UA is identical to the remote tag at the peer UA. The tags are opaque tokens that facilitate the generation of unique dialog IDs.

A dialog ID is also associated with all responses and with any request that contains a tag in the To field. The rules for computing the dialog ID of a message depend on whether the SIP element is a UAC or UAS. For a UAC, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the To field of the message, and the local tag is set to the tag in the From

field of the message (these rules apply to both requests and responses). As one would expect for a UAS, the Call-ID value of the dialog ID is set to the Call-ID of the message, the remote tag is set to the tag in the From field of the message, and the local tag is set to the tag in the To field of the message.

A dialog contains certain pieces of state needed for further message transmissions within the dialog. This state consists of the dialog ID, a local sequence number (used to order requests from the UA to its peer), a remote sequence number (used to order requests from its peer to the UA), a local URI, a remote URI, remote target, a boolean flag called "secure", and a route set, which is an ordered list of URIs. The route set is the list of servers that need to be traversed to send a request to the peer. A dialog can also be in the "early" state, which occurs when it is created with a provisional response, and then transition to the "confirmed" state when a 2xx final response arrives. For other responses, or if no response arrives at all on that dialog, the early dialog terminates.

12.1 Creation of a Dialog

Dialogs are created through the generation of non-failure responses to requests with specific methods. Within this specification, only 2xx and 101-199 responses with a To tag, where the request was INVITE, will establish a dialog. A dialog established by a non-final response to a request is in the "early" state and it is called an early dialog. Extensions MAY define other means for creating dialogs. Section 13 gives more details that are specific to the INVITE method. Here, we describe the process for creation of dialog state that is not dependent on the method.

UAs MUST assign values to the dialog ID components as described below.

12.1.1 UAS behavior

When a UAS responds to a request with a response that establishes a dialog (such as a 2xx to INVITE), the UAS MUST copy all Record-Route header field values from the request into the response (including the URIs, URI parameters, and any Record-Route header field parameters, whether they are known or unknown to the UAS) and MUST maintain the order of those values. The UAS MUST add a Contact header field to the response. The Contact header field contains an address where the UAS would like to be contacted for subsequent requests in the dialog (which includes the ACK for a 2xx response in the case of an INVITE). Generally, the host portion of this URI is the IP address or FQDN of the host. The URI provided in the Contact header field MUST be a SIP or SIPS URI. If the request that initiated the dialog contained a

SIPS URI in the Request-URI or in the top Record-Route header field value, if there was any, or the Contact header field if there was no Record-Route header field, the Contact header field in the response MUST be a SIPS URI. The URI SHOULD have global scope (that is, the same URI can be used in messages outside this dialog). The same way, the scope of the URI in the Contact header field of the INVITE is not limited to this dialog either. It can therefore be used in messages to the UAC even outside this dialog.

The UAS then constructs the state of the dialog. This state MUST be maintained for the duration of the dialog.

If the request arrived over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE.

The route set MUST be set to the list of URIs in the Record-Route header field from the request, taken in order and preserving all URI parameters. If no Record-Route header field is present in the request, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the request.

The remote sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The local sequence number MUST be empty. The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request. The local tag component of the dialog ID MUST be set to the tag in the To field in the response to the request (which always includes a tag), and the remote tag component of the dialog ID MUST be set to the tag from the From field in the request. A UAS MUST be prepared to receive a request without a tag in the From field, in which case the tag is considered to have a value of null.

> This is to maintain backwards compatibility with RFC 2543, which did not mandate From tags.

The remote URI MUST be set to the URI in the From field, and the local URI MUST be set to the URI in the To field.

12.1.2 UAC Behavior

When a UAC sends a request that can establish a dialog (such as an INVITE) it MUST provide a SIP or SIPS URI with global scope (i.e., the same SIP URI can be used in messages outside this dialog) in the Contact header field of the request. If the request has a Request-URI or a topmost Route header field value with a SIPS URI, the Contact header field MUST contain a SIPS URI.

When a UAC receives a response that establishes a dialog, it constructs the state of the dialog. This state MUST be maintained for the duration of the dialog.

If the request was sent over TLS, and the Request-URI contained a SIPS URI, the "secure" flag is set to TRUE.

The route set MUST be set to the list of URIs in the Record-Route header field from the response, taken in reverse order and preserving all URI parameters. If no Record-Route header field is present in the response, the route set MUST be set to the empty set. This route set, even if empty, overrides any pre-existing route set for future requests in this dialog. The remote target MUST be set to the URI from the Contact header field of the response.

The local sequence number MUST be set to the value of the sequence number in the CSeq header field of the request. The remote sequence number MUST be empty (it is established when the remote UA sends a request within the dialog). The call identifier component of the dialog ID MUST be set to the value of the Call-ID in the request. The local tag component of the dialog ID MUST be set to the tag in the From field in the request, and the remote tag component of the dialog ID MUST be set to the tag in the To field of the response. A UAC MUST be prepared to receive a response without a tag in the To field, in which case the tag is considered to have a value of null.

> This is to maintain backwards compatibility with RFC 2543, which did not mandate To tags.

The remote URI MUST be set to the URI in the To field, and the local URI MUST be set to the URI in the From field.

12.2 Requests within a Dialog

Once a dialog has been established between two UAs, either of them MAY initiate new transactions as needed within the dialog. The UA sending the request will take the UAC role for the transaction. The UA receiving the request will take the UAS role. Note that these may be different roles than the UAs held during the transaction that established the dialog.

Requests within a dialog MAY contain Record-Route and Contact header fields. However, these requests do not cause the dialog's route set to be modified, although they may modify the remote target URI. Specifically, requests that are not target refresh requests do not modify the dialog's remote target URI, and requests that are target refresh requests do. For dialogs that have been established with an

INVITE, the only target refresh request defined is re-INVITE (see Section 14). Other extensions may define different target refresh requests for dialogs established in other ways.

Note that an ACK is NOT a target refresh request.

Target refresh requests only update the dialog's remote target URI, and not the route set formed from the Record-Route. Updating the latter would introduce severe backwards compatibility problems with RFC 2543-compliant systems.

12.2.1 UAC Behavior

12.2.1.1 Generating the Request

A request within a dialog is constructed by using many of the components of the state stored as part of the dialog.

The URI in the To field of the request MUST be set to the remote URI from the dialog state. The tag in the To header field of the request MUST be set to the remote tag of the dialog ID. The From URI of the request MUST be set to the local URI from the dialog state. The tag in the From header field of the request MUST be set to the local tag of the dialog ID. If the value of the remote or local tags is null, the tag parameter MUST be omitted from the To or From header fields, respectively.

Usage of the URI from the To and From fields in the original request within subsequent requests is done for backwards compatibility with RFC 2543, which used the URI for dialog identification. In this specification, only the tags are used for dialog identification. It is expected that mandatory reflection of the original To and From URI in mid-dialog requests will be deprecated in a subsequent revision of this specification.

The Call-ID of the request MUST be set to the Call-ID of the dialog. Requests within a dialog MUST contain strictly monotonically increasing and contiguous CSeq sequence numbers (increasing-by-one) in each direction (excepting ACK and CANCEL of course, whose numbers equal the requests being acknowledged or cancelled). Therefore, if the local sequence number is not empty, the value of the local sequence number MUST be incremented by one, and this value MUST be placed into the CSeq header field. If the local sequence number is empty, an initial value MUST be chosen using the guidelines of Section 8.1.1.5. The method field in the CSeq header field value MUST match the method of the request.

With a length of 32 bits, a client could generate, within a single call, one request a second for about 136 years before needing to wrap around. The initial value of the sequence number is chosen so that subsequent requests within the same call will not wrap around. A non-zero initial value allows clients to use a time-based initial sequence number. A client could, for example, choose the 31 most significant bits of a 32-bit second clock as an initial sequence number.

The UAC uses the remote target and route set to build the Request-URI and Route header field of the request.

If the route set is empty, the UAC MUST place the remote target URI into the Request-URI. The UAC MUST NOT add a Route header field to the request.

If the route set is not empty, and the first URI in the route set contains the lr parameter (see Section 19.1.1), the UAC MUST place the remote target URI into the Request-URI and MUST include a Route header field containing the route set values in order, including all parameters.

If the route set is not empty, and its first URI does not contain the lr parameter, the UAC MUST place the first URI from the route set into the Request-URI, stripping any parameters that are not allowed in a Request-URI. The UAC MUST add a Route header field containing the remainder of the route set values in order, including all parameters. The UAC MUST then place the remote target URI into the Route header field as the last value.

For example, if the remote target is sip:user@remoteua and the route set contains:

<sip:proxy1>,<sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>

The request will be formed with the following Request-URI and Route header field:

METHOD sip:proxy1
Route: <sip:proxy2>,<sip:proxy3;lr>,<sip:proxy4>,<sip:user@remoteua>

If the first URI of the route set does not contain the lr parameter, the proxy indicated does not understand the routing mechanisms described in this document and will act as specified in RFC 2543, replacing the Request-URI with the first Route header field value it receives while forwarding the message. Placing the Request-URI at the end of the Route header field preserves the

information in that Request-URI across the strict router (it will be returned to the Request-URI when the request reaches a loose-router).

A UAC SHOULD include a Contact header field in any target refresh requests within a dialog, and unless there is a need to change it, the URI SHOULD be the same as used in previous requests within the dialog. If the "secure" flag is true, that URI MUST be a SIPS URI. As discussed in Section 12.2.2, a Contact header field in a target refresh request updates the remote target URI. This allows a UA to provide a new contact address, should its address change during the duration of the dialog.

However, requests that are not target refresh requests do not affect the remote target URI for the dialog.

The rest of the request is formed as described in Section 8.1.1.

Once the request has been constructed, the address of the server is computed and the request is sent, using the same procedures for requests outside of a dialog (Section 8.1.2).

The procedures in Section 8.1.2 will normally result in the request being sent to the address indicated by the topmost Route header field value or the Request-URI if no Route header field is present. Subject to certain restrictions, they allow the request to be sent to an alternate address (such as a default outbound proxy not represented in the route set).

12.2.1.2 Processing the Responses

The UAC will receive responses to the request from the transaction layer. If the client transaction returns a timeout, this is treated as a 408 (Request Timeout) response.

The behavior of a UAC that receives a 3xx response for a request sent within a dialog is the same as if the request had been sent outside a dialog. This behavior is described in Section 8.1.3.4.

Note, however, that when the UAC tries alternative locations, it still uses the route set for the dialog to build the Route header of the request.

When a UAC receives a 2xx response to a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that response, if present.

If the response for a request within a dialog is a 481 (Call/Transaction Does Not Exist) or a 408 (Request Timeout), the UAC SHOULD terminate the dialog. A UAC SHOULD also terminate a dialog if no response at all is received for the request (the client transaction would inform the TU about the timeout.)

For INVITE initiated dialogs, terminating the dialog consists of sending a BYE.

12.2.2 UAS Behavior

Requests sent within a dialog, as any other requests, are atomic. If a particular request is accepted by the UAS, all the state changes associated with it are performed. If the request is rejected, none of the state changes are performed.

Note that some requests, such as INVITEs, affect several pieces of state.

The UAS will receive the request from the transaction layer. If the request has a tag in the To header field, the UAS core computes the dialog identifier corresponding to the request and compares it with existing dialogs. If there is a match, this is a mid-dialog request. In that case, the UAS first applies the same processing rules for requests outside of a dialog, discussed in Section 8.2.

If the request has a tag in the To header field, but the dialog identifier does not match any existing dialogs, the UAS may have crashed and restarted, or it may have received a request for a different (possibly failed) UAS (the UASs can construct the To tags so that a UAS can identify that the tag was for a UAS for which it is providing recovery). Another possibility is that the incoming request has been simply misrouted. Based on the To tag, the UAS MAY either accept or reject the request. Accepting the request for acceptable To tags provides robustness, so that dialogs can persist even through crashes. UAs wishing to support this capability must take into consideration some issues such as choosing monotonically increasing CSeq sequence numbers even across reboots, reconstructing the route set, and accepting out-of-range RTP timestamps and sequence numbers.

If the UAS wishes to reject the request because it does not wish to recreate the dialog, it MUST respond to the request with a 481 (Call/Transaction Does Not Exist) status code and pass that to the server transaction.

Requests that do not change in any way the state of a dialog may be received within a dialog (for example, an OPTIONS request). They are processed as if they had been received outside the dialog.

If the remote sequence number is empty, it MUST be set to the value of the sequence number in the CSeq header field value in the request. If the remote sequence number was not empty, but the sequence number of the request is lower than the remote sequence number, the request is out of order and MUST be rejected with a 500 (Server Internal Error) response. If the remote sequence number was not empty, and the sequence number of the request is greater than the remote sequence number, the request is in order. It is possible for the CSeq sequence number to be higher than the remote sequence number by more than one. This is not an error condition, and a UAS SHOULD be prepared to receive and process requests with CSeq values more than one higher than the previous received request. The UAS MUST then set the remote sequence number to the value of the sequence number in the CSeq header field value in the request.

> If a proxy challenges a request generated by the UAC, the UAC has to resubmit the request with credentials. The resubmitted request will have a new CSeq number. The UAS will never see the first request, and thus, it will notice a gap in the CSeq number space. Such a gap does not represent any error condition.

When a UAS receives a target refresh request, it MUST replace the dialog's remote target URI with the URI from the Contact header field in that request, if present.

12.3 Termination of a Dialog

Independent of the method, if a request outside of a dialog generates a non-2xx final response, any early dialogs created through provisional responses to that request are terminated. The mechanism for terminating confirmed dialogs is method specific. In this specification, the BYE method terminates a session and the dialog associated with it. See Section 15 for details.

13 Initiating a Session

13.1 Overview

When a user agent client desires to initiate a session (for example, audio, video, or a game), it formulates an INVITE request. The INVITE request asks a server to establish a session. This request may be forwarded by proxies, eventually arriving at one or more UAS that can potentially accept the invitation. These UASs will frequently need to query the user about whether to accept the

invitation. After some time, those UASs can accept the invitation (meaning the session is to be established) by sending a 2xx response. If the invitation is not accepted, a 3xx, 4xx, 5xx or 6xx response is sent, depending on the reason for the rejection. Before sending a final response, the UAS can also send provisional responses (1xx) to advise the UAC of progress in contacting the called user.

After possibly receiving one or more provisional responses, the UAC will get one or more 2xx responses or one non-2xx final response. Because of the protracted amount of time it can take to receive final responses to INVITE, the reliability mechanisms for INVITE transactions differ from those of other requests (like OPTIONS). Once it receives a final response, the UAC needs to send an ACK for every final response it receives. The procedure for sending this ACK depends on the type of response. For final responses between 300 and 699, the ACK processing is done in the transaction layer and follows one set of rules (See Section 17). For 2xx responses, the ACK is generated by the UAC core.

A 2xx response to an INVITE establishes a session, and it also creates a dialog between the UA that issued the INVITE and the UA that generated the 2xx response. Therefore, when multiple 2xx responses are received from different remote UAs (because the INVITE forked), each 2xx establishes a different dialog. All these dialogs are part of the same call.

This section provides details on the establishment of a session using INVITE. A UA that supports INVITE MUST also support ACK, CANCEL and BYE.

13.2 UAC Processing

13.2.1 Creating the Initial INVITE

Since the initial INVITE represents a request outside of a dialog, its construction follows the procedures of Section 8.1.1. Additional processing is required for the specific case of INVITE.

An Allow header field (Section 20.5) SHOULD be present in the INVITE. It indicates what methods can be invoked within a dialog, on the UA sending the INVITE, for the duration of the dialog. For example, a UA capable of receiving INFO requests within a dialog [34] SHOULD include an Allow header field listing the INFO method.

A Supported header field (Section 20.37) SHOULD be present in the INVITE. It enumerates all the extensions understood by the UAC.

An Accept (Section 20.1) header field MAY be present in the INVITE. It indicates which Content-Types are acceptable to the UA, in both the response received by it, and in any subsequent requests sent to it within dialogs established by the INVITE. The Accept header field is especially useful for indicating support of various session description formats.

The UAC MAY add an Expires header field (Section 20.19) to limit the validity of the invitation. If the time indicated in the Expires header field is reached and no final answer for the INVITE has been received, the UAC core SHOULD generate a CANCEL request for the INVITE, as per Section 9.

A UAC MAY also find it useful to add, among others, Subject (Section 20.36), Organization (Section 20.25) and User-Agent (Section 20.41) header fields. They all contain information related to the INVITE.

The UAC MAY choose to add a message body to the INVITE. Section 8.1.1.10 deals with how to construct the header fields -- Content-Type among others -- needed to describe the message body.

There are special rules for message bodies that contain a session description - their corresponding Content-Disposition is "session". SIP uses an offer/answer model where one UA sends a session description, called the offer, which contains a proposed description of the session. The offer indicates the desired communications means (audio, video, games), parameters of those means (such as codec types) and addresses for receiving media from the answerer. The other UA responds with another session description, called the answer, which indicates which communications means are accepted, the parameters that apply to those means, and addresses for receiving media from the offerer. An offer/answer exchange is within the context of a dialog, so that if a SIP INVITE results in multiple dialogs, each is a separate offer/answer exchange. The offer/answer model defines restrictions on when offers and answers can be made (for example, you cannot make a new offer while one is in progress). This results in restrictions on where the offers and answers can appear in SIP messages. In this specification, offers and answers can only appear in INVITE requests and responses, and ACK. The usage of offers and answers is further restricted. For the initial INVITE transaction, the rules are:

- The initial offer MUST be in either an INVITE or, if not there, in the first reliable non-failure message from the UAS back to the UAC. In this specification, that is the final 2xx response.

- If the initial offer is in an INVITE, the answer MUST be in a reliable non-failure message from UAS back to UAC which is correlated to that INVITE. For this specification, that is only the final 2xx response to that INVITE. That same exact answer MAY also be placed in any provisional responses sent prior to the answer. The UAC MUST treat the first session description it receives as the answer, and MUST ignore any session descriptions in subsequent responses to the initial INVITE.

- If the initial offer is in the first reliable non-failure message from the UAS back to UAC, the answer MUST be in the acknowledgement for that message (in this specification, ACK for a 2xx response).

- After having sent or received an answer to the first offer, the UAC MAY generate subsequent offers in requests based on rules specified for that method, but only if it has received answers to any previous offers, and has not sent any offers to which it hasn't gotten an answer.

- Once the UAS has sent or received an answer to the initial offer, it MUST NOT generate subsequent offers in any responses to the initial INVITE. This means that a UAS based on this specification alone can never generate subsequent offers until completion of the initial transaction.

Concretely, the above rules specify two exchanges for UAs compliant to this specification alone - the offer is in the INVITE, and the answer in the 2xx (and possibly in a 1xx as well, with the same value), or the offer is in the 2xx, and the answer is in the ACK. All user agents that support INVITE MUST support these two exchanges.

The Session Description Protocol (SDP) (RFC 2327 [1]) MUST be supported by all user agents as a means to describe sessions, and its usage for constructing offers and answers MUST follow the procedures defined in [13].

The restrictions of the offer-answer model just described only apply to bodies whose Content-Disposition header field value is "session". Therefore, it is possible that both the INVITE and the ACK contain a body message (for example, the INVITE carries a photo (Content-Disposition: render) and the ACK a session description (Content-Disposition: session)).

If the Content-Disposition header field is missing, bodies of Content-Type application/sdp imply the disposition "session", while other content types imply "render".

Once the INVITE has been created, the UAC follows the procedures defined for sending requests outside of a dialog (Section 8). This results in the construction of a client transaction that will ultimately send the request and deliver responses to the UAC.

13.2.2 Processing INVITE Responses

Once the INVITE has been passed to the INVITE client transaction, the UAC waits for responses for the INVITE. If the INVITE client transaction returns a timeout rather than a response the TU acts as if a 408 (Request Timeout) response had been received, as described in Section 8.1.3.

13.2.2.1 1xx Responses

Zero, one or multiple provisional responses may arrive before one or more final responses are received. Provisional responses for an INVITE request can create "early dialogs". If a provisional response has a tag in the To field, and if the dialog ID of the response does not match an existing dialog, one is constructed using the procedures defined in Section 12.1.2.

The early dialog will only be needed if the UAC needs to send a request to its peer within the dialog before the initial INVITE transaction completes. Header fields present in a provisional response are applicable as long as the dialog is in the early state (for example, an Allow header field in a provisional response contains the methods that can be used in the dialog while this is in the early state).

13.2.2.2 3xx Responses

A 3xx response may contain one or more Contact header field values providing new addresses where the callee might be reachable. Depending on the status code of the 3xx response (see Section 21.3), the UAC MAY choose to try those new addresses.

13.2.2.3 4xx, 5xx and 6xx Responses

A single non-2xx final response may be received for the INVITE. 4xx, 5xx and 6xx responses may contain a Contact header field value indicating the location where additional information about the error can be found. Subsequent final responses (which would only arrive under error conditions) MUST be ignored.

All early dialogs are considered terminated upon reception of the non-2xx final response.

After having received the non-2xx final response the UAC core considers the INVITE transaction completed. The INVITE client transaction handles the generation of ACKs for the response (see Section 17).

13.2.2.4 2xx Responses

Multiple 2xx responses may arrive at the UAC for a single INVITE request due to a forking proxy. Each response is distinguished by the tag parameter in the To header field, and each represents a distinct dialog, with a distinct dialog identifier.

If the dialog identifier in the 2xx response matches the dialog identifier of an existing dialog, the dialog MUST be transitioned to the "confirmed" state, and the route set for the dialog MUST be recomputed based on the 2xx response using the procedures of Section 12.2.1.2. Otherwise, a new dialog in the "confirmed" state MUST be constructed using the procedures of Section 12.1.2.

> Note that the only piece of state that is recomputed is the route set. Other pieces of state such as the highest sequence numbers (remote and local) sent within the dialog are not recomputed. The route set only is recomputed for backwards compatibility. RFC 2543 did not mandate mirroring of the Record-Route header field in a 1xx, only 2xx. However, we cannot update the entire state of the dialog, since mid-dialog requests may have been sent within the early dialog, modifying the sequence numbers, for example.

The UAC core MUST generate an ACK request for each 2xx received from the transaction layer. The header fields of the ACK are constructed in the same way as for any request sent within a dialog (see Section 12) with the exception of the CSeq and the header fields related to authentication. The sequence number of the CSeq header field MUST be the same as the INVITE being acknowledged, but the CSeq method MUST be ACK. The ACK MUST contain the same credentials as the INVITE. If the 2xx contains an offer (based on the rules above), the ACK MUST carry an answer in its body. If the offer in the 2xx response is not acceptable, the UAC core MUST generate a valid answer in the ACK and then send a BYE immediately.

Once the ACK has been constructed, the procedures of [4] are used to determine the destination address, port and transport. However, the request is passed to the transport layer directly for transmission, rather than a client transaction. This is because the UAC core handles retransmissions of the ACK, not the transaction layer. The ACK MUST be passed to the client transport every time a retransmission of the 2xx final response that triggered the ACK arrives.

The UAC core considers the INVITE transaction completed 64*T1 seconds after the reception of the first 2xx response. At this point all the early dialogs that have not transitioned to established dialogs are terminated. Once the INVITE transaction is considered completed by the UAC core, no more new 2xx responses are expected to arrive.

If, after acknowledging any 2xx response to an INVITE, the UAC does not want to continue with that dialog, then the UAC MUST terminate the dialog by sending a BYE request as described in Section 15.

13.3 UAS Processing

13.3.1 Processing of the INVITE

The UAS core will receive INVITE requests from the transaction layer. It first performs the request processing procedures of Section 8.2, which are applied for both requests inside and outside of a dialog.

Assuming these processing states are completed without generating a response, the UAS core performs the additional processing steps:

1. If the request is an INVITE that contains an Expires header field, the UAS core sets a timer for the number of seconds indicated in the header field value. When the timer fires, the invitation is considered to be expired. If the invitation expires before the UAS has generated a final response, a 487 (Request Terminated) response SHOULD be generated.

2. If the request is a mid-dialog request, the method-independent processing described in Section 12.2.2 is first applied. It might also modify the session; Section 14 provides details.

3. If the request has a tag in the To header field but the dialog identifier does not match any of the existing dialogs, the UAS may have crashed and restarted, or may have received a request for a different (possibly failed) UAS. Section 12.2.2 provides guidelines to achieve a robust behavior under such a situation.

Processing from here forward assumes that the INVITE is outside of a dialog, and is thus for the purposes of establishing a new session.

The INVITE may contain a session description, in which case the UAS is being presented with an offer for that session. It is possible that the user is already a participant in that session, even though the INVITE is outside of a dialog. This can happen when a user is invited to the same multicast conference by multiple other participants. If desired, the UAS MAY use identifiers within the session description to detect this duplication. For example, SDP

contains a session id and version number in the origin (o) field. If the user is already a member of the session, and the session parameters contained in the session description have not changed, the UAS MAY silently accept the INVITE (that is, send a 2xx response without prompting the user).

If the INVITE does not contain a session description, the UAS is being asked to participate in a session, and the UAC has asked that the UAS provide the offer of the session. It MUST provide the offer in its first non-failure reliable message back to the UAC. In this specification, that is a 2xx response to the INVITE.

The UAS can indicate progress, accept, redirect, or reject the invitation. In all of these cases, it formulates a response using the procedures described in Section 8.2.6.

13.3.1.1 Progress

If the UAS is not able to answer the invitation immediately, it can choose to indicate some kind of progress to the UAC (for example, an indication that a phone is ringing). This is accomplished with a provisional response between 101 and 199. These provisional responses establish early dialogs and therefore follow the procedures of Section 12.1.1 in addition to those of Section 8.2.6. A UAS MAY send as many provisional responses as it likes. Each of these MUST indicate the same dialog ID. However, these will not be delivered reliably.

If the UAS desires an extended period of time to answer the INVITE, it will need to ask for an "extension" in order to prevent proxies from canceling the transaction. A proxy has the option of canceling a transaction when there is a gap of 3 minutes between responses in a transaction. To prevent cancellation, the UAS MUST send a non-100 provisional response at every minute, to handle the possibility of lost provisional responses.

> An INVITE transaction can go on for extended durations when the user is placed on hold, or when interworking with PSTN systems which allow communications to take place without answering the call. The latter is common in Interactive Voice Response (IVR) systems.

13.3.1.2 The INVITE is Redirected

If the UAS decides to redirect the call, a 3xx response is sent. A 300 (Multiple Choices), 301 (Moved Permanently) or 302 (Moved Temporarily) response SHOULD contain a Contact header field

containing one or more URIs of new addresses to be tried. The response is passed to the INVITE server transaction, which will deal with its retransmissions.

13.3.1.3 The INVITE is Rejected

A common scenario occurs when the callee is currently not willing or able to take additional calls at this end system. A 486 (Busy Here) SHOULD be returned in such a scenario. If the UAS knows that no other end system will be able to accept this call, a 600 (Busy Everywhere) response SHOULD be sent instead. However, it is unlikely that a UAS will be able to know this in general, and thus this response will not usually be used. The response is passed to the INVITE server transaction, which will deal with its retransmissions.

A UAS rejecting an offer contained in an INVITE SHOULD return a 488 (Not Acceptable Here) response. Such a response SHOULD include a Warning header field value explaining why the offer was rejected.

13.3.1.4 The INVITE is Accepted

The UAS core generates a 2xx response. This response establishes a dialog, and therefore follows the procedures of Section 12.1.1 in addition to those of Section 8.2.6.

A 2xx response to an INVITE SHOULD contain the Allow header field and the Supported header field, and MAY contain the Accept header field. Including these header fields allows the UAC to determine the features and extensions supported by the UAS for the duration of the call, without probing.

If the INVITE request contained an offer, and the UAS had not yet sent an answer, the 2xx MUST contain an answer. If the INVITE did not contain an offer, the 2xx MUST contain an offer if the UAS had not yet sent an offer.

Once the response has been constructed, it is passed to the INVITE server transaction. Note, however, that the INVITE server transaction will be destroyed as soon as it receives this final response and passes it to the transport. Therefore, it is necessary to periodically pass the response directly to the transport until the ACK arrives. The 2xx response is passed to the transport with an interval that starts at T1 seconds and doubles for each retransmission until it reaches T2 seconds (T1 and T2 are defined in Section 17). Response retransmissions cease when an ACK request for the response is received. This is independent of whatever transport protocols are used to send the response.

Since 2xx is retransmitted end-to-end, there may be hops between UAS and UAC that are UDP. To ensure reliable delivery across these hops, the response is retransmitted periodically even if the transport at the UAS is reliable.

If the server retransmits the 2xx response for 64*T1 seconds without receiving an ACK, the dialog is confirmed, but the session SHOULD be terminated. This is accomplished with a BYE, as described in Section 15.

14 Modifying an Existing Session

A successful INVITE request (see Section 13) establishes both a dialog between two user agents and a session using the offer-answer model. Section 12 explains how to modify an existing dialog using a target refresh request (for example, changing the remote target URI of the dialog). This section describes how to modify the actual session. This modification can involve changing addresses or ports, adding a media stream, deleting a media stream, and so on. This is accomplished by sending a new INVITE request within the same dialog that established the session. An INVITE request sent within an existing dialog is known as a re-INVITE.

Note that a single re-INVITE can modify the dialog and the parameters of the session at the same time.

Either the caller or callee can modify an existing session.

The behavior of a UA on detection of media failure is a matter of local policy. However, automated generation of re-INVITE or BYE is NOT RECOMMENDED to avoid flooding the network with traffic when there is congestion. In any case, if these messages are sent automatically, they SHOULD be sent after some randomized interval.

Note that the paragraph above refers to automatically generated BYEs and re-INVITEs. If the user hangs up upon media failure, the UA would send a BYE request as usual.

14.1 UAC Behavior

The same offer-answer model that applies to session descriptions in INVITEs (Section 13.2.1) applies to re-INVITEs. As a result, a UAC that wants to add a media stream, for example, will create a new offer that contains this media stream, and send that in an INVITE request to its peer. It is important to note that the full description of the session, not just the change, is sent. This supports stateless session processing in various elements, and supports failover and recovery capabilities. Of course, a UAC MAY

send a re-INVITE with no session description, in which case the first reliable non-failure response to the re-INVITE will contain the offer (in this specification, that is a 2xx response).

If the session description format has the capability for version numbers, the offerer SHOULD indicate that the version of the session description has changed.

The To, From, Call-ID, CSeq, and Request-URI of a re-INVITE are set following the same rules as for regular requests within an existing dialog, described in Section 12.

A UAC MAY choose not to add an Alert-Info header field or a body with Content-Disposition "alert" to re-INVITEs because UASs do not typically alert the user upon reception of a re-INVITE.

Unlike an INVITE, which can fork, a re-INVITE will never fork, and therefore, only ever generate a single final response. The reason a re-INVITE will never fork is that the Request-URI identifies the target as the UA instance it established the dialog with, rather than identifying an address-of-record for the user.

Note that a UAC MUST NOT initiate a new INVITE transaction within a dialog while another INVITE transaction is in progress in either direction.

1. If there is an ongoing INVITE client transaction, the TU MUST wait until the transaction reaches the completed or terminated state before initiating the new INVITE.

2. If there is an ongoing INVITE server transaction, the TU MUST wait until the transaction reaches the confirmed or terminated state before initiating the new INVITE.

However, a UA MAY initiate a regular transaction while an INVITE transaction is in progress. A UA MAY also initiate an INVITE transaction while a regular transaction is in progress.

If a UA receives a non-2xx final response to a re-INVITE, the session parameters MUST remain unchanged, as if no re-INVITE had been issued. Note that, as stated in Section 12.2.1.2, if the non-2xx final response is a 481 (Call/Transaction Does Not Exist), or a 408 (Request Timeout), or no response at all is received for the re-INVITE (that is, a timeout is returned by the INVITE client transaction), the UAC will terminate the dialog.

If a UAC receives a 491 response to a re-INVITE, it SHOULD start a timer with a value T chosen as follows:

1. If the UAC is the owner of the Call-ID of the dialog ID (meaning it generated the value), T has a randomly chosen value between 2.1 and 4 seconds in units of 10 ms.

2. If the UAC is not the owner of the Call-ID of the dialog ID, T has a randomly chosen value of between 0 and 2 seconds in units of 10 ms.

When the timer fires, the UAC SHOULD attempt the re-INVITE once more, if it still desires for that session modification to take place. For example, if the call was already hung up with a BYE, the re-INVITE would not take place.

The rules for transmitting a re-INVITE and for generating an ACK for a 2xx response to re-INVITE are the same as for the initial INVITE (Section 13.2.1).

14.2 UAS Behavior

Section 13.3.1 describes the procedure for distinguishing incoming re-INVITEs from incoming initial INVITEs and handling a re-INVITE for an existing dialog.

A UAS that receives a second INVITE before it sends the final response to a first INVITE with a lower CSeq sequence number on the same dialog MUST return a 500 (Server Internal Error) response to the second INVITE and MUST include a Retry-After header field with a randomly chosen value of between 0 and 10 seconds.

A UAS that receives an INVITE on a dialog while an INVITE it had sent on that dialog is in progress MUST return a 491 (Request Pending) response to the received INVITE.

If a UA receives a re-INVITE for an existing dialog, it MUST check any version identifiers in the session description or, if there are no version identifiers, the content of the session description to see if it has changed. If the session description has changed, the UAS MUST adjust the session parameters accordingly, possibly after asking the user for confirmation.

> Versioning of the session description can be used to accommodate the capabilities of new arrivals to a conference, add or delete media, or change from a unicast to a multicast conference.

If the new session description is not acceptable, the UAS can reject it by returning a 488 (Not Acceptable Here) response for the re-INVITE. This response SHOULD include a Warning header field.

If a UAS generates a 2xx response and never receives an ACK, it SHOULD generate a BYE to terminate the dialog.

A UAS MAY choose not to generate 180 (Ringing) responses for a re-INVITE because UACs do not typically render this information to the user. For the same reason, UASs MAY choose not to use an Alert-Info header field or a body with Content-Disposition "alert" in responses to a re-INVITE.

A UAS providing an offer in a 2xx (because the INVITE did not contain an offer) SHOULD construct the offer as if the UAS were making a brand new call, subject to the constraints of sending an offer that updates an existing session, as described in [13] in the case of SDP. Specifically, this means that it SHOULD include as many media formats and media types that the UA is willing to support. The UAS MUST ensure that the session description overlaps with its previous session description in media formats, transports, or other parameters that require support from the peer. This is to avoid the need for the peer to reject the session description. If, however, it is unacceptable to the UAC, the UAC SHOULD generate an answer with a valid session description, and then send a BYE to terminate the session.

15 Terminating a Session

This section describes the procedures for terminating a session established by SIP. The state of the session and the state of the dialog are very closely related. When a session is initiated with an INVITE, each 1xx or 2xx response from a distinct UAS creates a dialog, and if that response completes the offer/answer exchange, it also creates a session. As a result, each session is "associated" with a single dialog - the one which resulted in its creation. If an initial INVITE generates a non-2xx final response, that terminates all sessions (if any) and all dialogs (if any) that were created through responses to the request. By virtue of completing the transaction, a non-2xx final response also prevents further sessions from being created as a result of the INVITE. The BYE request is used to terminate a specific session or attempted session. In this case, the specific session is the one with the peer UA on the other side of the dialog. When a BYE is received on a dialog, any session associated with that dialog SHOULD terminate. A UA MUST NOT send a BYE outside of a dialog. The caller's UA MAY send a BYE for either confirmed or early dialogs, and the callee's UA MAY send a BYE on confirmed dialogs, but MUST NOT send a BYE on early dialogs.

However, the callee's UA MUST NOT send a BYE on a confirmed dialog until it has received an ACK for its 2xx response or until the server transaction times out. If no SIP extensions have defined other application layer states associated with the dialog, the BYE also terminates the dialog.

The impact of a non-2xx final response to INVITE on dialogs and sessions makes the use of CANCEL attractive. The CANCEL attempts to force a non-2xx response to the INVITE (in particular, a 487). Therefore, if a UAC wishes to give up on its call attempt entirely, it can send a CANCEL. If the INVITE results in 2xx final response(s) to the INVITE, this means that a UAS accepted the invitation while the CANCEL was in progress. The UAC MAY continue with the sessions established by any 2xx responses, or MAY terminate them with BYE.

> The notion of "hanging up" is not well defined within SIP. It is specific to a particular, albeit common, user interface. Typically, when the user hangs up, it indicates a desire to terminate the attempt to establish a session, and to terminate any sessions already created. For the caller's UA, this would imply a CANCEL request if the initial INVITE has not generated a final response, and a BYE to all confirmed dialogs after a final response. For the callee's UA, it would typically imply a BYE; presumably, when the user picked up the phone, a 2xx was generated, and so hanging up would result in a BYE after the ACK is received. This does not mean a user cannot hang up before receipt of the ACK, it just means that the software in his phone needs to maintain state for a short while in order to clean up properly. If the particular UI allows for the user to reject a call before its answered, a 403 (Forbidden) is a good way to express that. As per the rules above, a BYE can't be sent.

15.1 Terminating a Session with a BYE Request

15.1.1 UAC Behavior

A BYE request is constructed as would any other request within a dialog, as described in Section 12.

Once the BYE is constructed, the UAC core creates a new non-INVITE client transaction, and passes it the BYE request. The UAC MUST consider the session terminated (and therefore stop sending or listening for media) as soon as the BYE request is passed to the client transaction. If the response for the BYE is a 481 (Call/Transaction Does Not Exist) or a 408 (Request Timeout) or no

response at all is received for the BYE (that is, a timeout is returned by the client transaction), the UAC MUST consider the session and the dialog terminated.

15.1.2 UAS Behavior

A UAS first processes the BYE request according to the general UAS processing described in Section 8.2. A UAS core receiving a BYE request checks if it matches an existing dialog. If the BYE does not match an existing dialog, the UAS core SHOULD generate a 481 (Call/Transaction Does Not Exist) response and pass that to the server transaction.

> This rule means that a BYE sent without tags by a UAC will be rejected. This is a change from RFC 2543, which allowed BYE without tags.

A UAS core receiving a BYE request for an existing dialog MUST follow the procedures of Section 12.2.2 to process the request. Once done, the UAS SHOULD terminate the session (and therefore stop sending and listening for media). The only case where it can elect not to are multicast sessions, where participation is possible even if the other participant in the dialog has terminated its involvement in the session. Whether or not it ends its participation on the session, the UAS core MUST generate a 2xx response to the BYE, and MUST pass that to the server transaction for transmission.

The UAS MUST still respond to any pending requests received for that dialog. It is RECOMMENDED that a 487 (Request Terminated) response be generated to those pending requests.

16 Proxy Behavior

16.1 Overview

SIP proxies are elements that route SIP requests to user agent servers and SIP responses to user agent clients. A request may traverse several proxies on its way to a UAS. Each will make routing decisions, modifying the request before forwarding it to the next element. Responses will route through the same set of proxies traversed by the request in the reverse order.

Being a proxy is a logical role for a SIP element. When a request arrives, an element that can play the role of a proxy first decides if it needs to respond to the request on its own. For instance, the request may be malformed or the element may need credentials from the client before acting as a proxy. The element MAY respond with any

appropriate error code. When responding directly to a request, the element is playing the role of a UAS and MUST behave as described in Section 8.2.

A proxy can operate in either a stateful or stateless mode for each new request. When stateless, a proxy acts as a simple forwarding element. It forwards each request downstream to a single element determined by making a targeting and routing decision based on the request. It simply forwards every response it receives upstream. A stateless proxy discards information about a message once the message has been forwarded. A stateful proxy remembers information (specifically, transaction state) about each incoming request and any requests it sends as a result of processing the incoming request. It uses this information to affect the processing of future messages associated with that request. A stateful proxy MAY choose to "fork" a request, routing it to multiple destinations. Any request that is forwarded to more than one location MUST be handled statefully.

In some circumstances, a proxy MAY forward requests using stateful transports (such as TCP) without being transaction-stateful. For instance, a proxy MAY forward a request from one TCP connection to another transaction statelessly as long as it places enough information in the message to be able to forward the response down the same connection the request arrived on. Requests forwarded between different types of transports where the proxy's TU must take an active role in ensuring reliable delivery on one of the transports MUST be forwarded transaction statefully.

A stateful proxy MAY transition to stateless operation at any time during the processing of a request, so long as it did not do anything that would otherwise prevent it from being stateless initially (forking, for example, or generation of a 100 response). When performing such a transition, all state is simply discarded. The proxy SHOULD NOT initiate a CANCEL request.

Much of the processing involved when acting statelessly or statefully for a request is identical. The next several subsections are written from the point of view of a stateful proxy. The last section calls out those places where a stateless proxy behaves differently.

16.2 Stateful Proxy

When stateful, a proxy is purely a SIP transaction processing engine. Its behavior is modeled here in terms of the server and client transactions defined in Section 17. A stateful proxy has a server transaction associated with one or more client transactions by a higher layer proxy processing component (see figure 3), known as a proxy core. An incoming request is processed by a server

transaction. Requests from the server transaction are passed to a proxy core. The proxy core determines where to route the request, choosing one or more next-hop locations. An outgoing request for each next-hop location is processed by its own associated client transaction. The proxy core collects the responses from the client transactions and uses them to send responses to the server transaction.

A stateful proxy creates a new server transaction for each new request received. Any retransmissions of the request will then be handled by that server transaction per Section 17. The proxy core MUST behave as a UAS with respect to sending an immediate provisional on that server transaction (such as 100 Trying) as described in Section 8.2.6. Thus, a stateful proxy SHOULD NOT generate 100 (Trying) responses to non-INVITE requests.

This is a model of proxy behavior, not of software. An implementation is free to take any approach that replicates the external behavior this model defines.

For all new requests, including any with unknown methods, an element intending to proxy the request MUST:

1. Validate the request (Section 16.3)

2. Preprocess routing information (Section 16.4)

3. Determine target(s) for the request (Section 16.5)

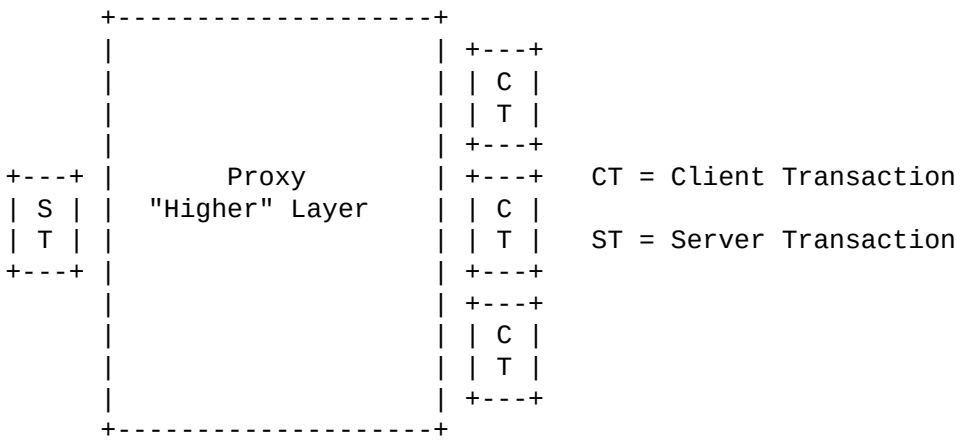


Figure 3: Stateful Proxy Model

4. Forward the request to each target (Section 16.6)

5. Process all responses (Section 16.7)

16.3 Request Validation

Before an element can proxy a request, it MUST verify the message's validity. A valid message must pass the following checks:

1. Reasonable Syntax

2. URI scheme

3. Max-Forwards

4. (Optional) Loop Detection

5. Proxy-Require

6. Proxy-Authorization

If any of these checks fail, the element MUST behave as a user agent server (see Section 8.2) and respond with an error code.

Notice that a proxy is not required to detect merged requests and MUST NOT treat merged requests as an error condition. The endpoints receiving the requests will resolve the merge as described in Section 8.2.2.2.

1. Reasonable syntax check

The request MUST be well-formed enough to be handled with a server transaction. Any components involved in the remainder of these Request Validation steps or the Request Forwarding section MUST be well-formed. Any other components, well-formed or not, SHOULD be ignored and remain unchanged when the message is forwarded. For instance, an element would not reject a request because of a malformed Date header field. Likewise, a proxy would not remove a malformed Date header field before forwarding a request.

This protocol is designed to be extended. Future extensions may define new methods and header fields at any time. An element MUST NOT refuse to proxy a request because it contains a method or header field it does not know about.

2. URI scheme check

   If the Request-URI has a URI whose scheme is not understood by the proxy, the proxy SHOULD reject the request with a 416 (Unsupported URI Scheme) response.

3. Max-Forwards check

   The Max-Forwards header field (Section 20.22) is used to limit the number of elements a SIP request can traverse.

   If the request does not contain a Max-Forwards header field, this check is passed.

   If the request contains a Max-Forwards header field with a field value greater than zero, the check is passed.

   If the request contains a Max-Forwards header field with a field value of zero (0), the element MUST NOT forward the request. If the request was for OPTIONS, the element MAY act as the final recipient and respond per Section 11. Otherwise, the element MUST return a 483 (Too many hops) response.

4. Optional Loop Detection check

   An element MAY check for forwarding loops before forwarding a request. If the request contains a Via header field with a sent-by value that equals a value placed into previous requests by the proxy, the request has been forwarded by this element before. The request has either looped or is legitimately spiraling through the element. To determine if the request has looped, the element MAY perform the branch parameter calculation described in Step 8 of Section 16.6 on this message and compare it to the parameter received in that Via header field. If the parameters match, the request has looped. If they differ, the request is spiraling, and processing continues. If a loop is detected, the element MAY return a 482 (Loop Detected) response.

5. Proxy-Require check

   Future extensions to this protocol may introduce features that require special handling by proxies. Endpoints will include a Proxy-Require header field in requests that use these features, telling the proxy not to process the request unless the feature is understood.

If the request contains a Proxy-Require header field (Section 20.29) with one or more option-tags this element does not understand, the element MUST return a 420 (Bad Extension) response. The response MUST include an Unsupported (Section 20.40) header field listing those option-tags the element did not understand.

6. Proxy-Authorization check

If an element requires credentials before forwarding a request, the request MUST be inspected as described in Section 22.3. That section also defines what the element must do if the inspection fails.

16.4 Route Information Preprocessing

The proxy MUST inspect the Request-URI of the request. If the Request-URI of the request contains a value this proxy previously placed into a Record-Route header field (see Section 16.6 item 4), the proxy MUST replace the Request-URI in the request with the last value from the Route header field, and remove that value from the Route header field. The proxy MUST then proceed as if it received this modified request.

This will only happen when the element sending the request to the proxy (which may have been an endpoint) is a strict router. This rewrite on receive is necessary to enable backwards compatibility with those elements. It also allows elements following this specification to preserve the Request-URI through strict-routing proxies (see Section 12.2.1.1).

This requirement does not obligate a proxy to keep state in order to detect URIs it previously placed in Record-Route header fields. Instead, a proxy need only place enough information in those URIs to recognize them as values it provided when they later appear.

If the Request-URI contains a maddr parameter, the proxy MUST check to see if its value is in the set of addresses or domains the proxy is configured to be responsible for. If the Request-URI has a maddr parameter with a value the proxy is responsible for, and the request was received using the port and transport indicated (explicitly or by default) in the Request-URI, the proxy MUST strip the maddr and any non-default port or transport parameter and continue processing as if those values had not been present in the request.

A request may arrive with a maddr matching the proxy, but on a port or transport different from that indicated in the URI. Such a request needs to be forwarded to the proxy using the indicated port and transport.

If the first value in the Route header field indicates this proxy, the proxy MUST remove that value from the request.

16.5 Determining Request Targets

Next, the proxy calculates the target(s) of the request. The set of targets will either be predetermined by the contents of the request or will be obtained from an abstract location service. Each target in the set is represented as a URI.

If the Request-URI of the request contains an maddr parameter, the Request-URI MUST be placed into the target set as the only target URI, and the proxy MUST proceed to Section 16.6.

If the domain of the Request-URI indicates a domain this element is not responsible for, the Request-URI MUST be placed into the target set as the only target, and the element MUST proceed to the task of Request Forwarding (Section 16.6).

There are many circumstances in which a proxy might receive a request for a domain it is not responsible for. A firewall proxy handling outgoing calls (the way HTTP proxies handle outgoing requests) is an example of where this is likely to occur.

If the target set for the request has not been predetermined as described above, this implies that the element is responsible for the domain in the Request-URI, and the element MAY use whatever mechanism it desires to determine where to send the request. Any of these mechanisms can be modeled as accessing an abstract Location Service. This may consist of obtaining information from a location service created by a SIP Registrar, reading a database, consulting a presence server, utilizing other protocols, or simply performing an algorithmic substitution on the Request-URI. When accessing the location service constructed by a registrar, the Request-URI MUST first be canonicalized as described in Section 10.3 before being used as an index. The output of these mechanisms is used to construct the target set.

If the Request-URI does not provide sufficient information for the proxy to determine the target set, it SHOULD return a 485 (Ambiguous) response. This response SHOULD contain a Contact header field containing URIs of new addresses to be tried. For example, an INVITE

to sip:John.Smith@company.com may be ambiguous at a proxy whose location service has multiple John Smiths listed. See Section 21.4.23 for details.

Any information in or about the request or the current environment of the element MAY be used in the construction of the target set. For instance, different sets may be constructed depending on contents or the presence of header fields and bodies, the time of day of the request's arrival, the interface on which the request arrived, failure of previous requests, or even the element's current level of utilization.

As potential targets are located through these services, their URIs are added to the target set. Targets can only be placed in the target set once. If a target URI is already present in the set (based on the definition of equality for the URI type), it MUST NOT be added again.

A proxy MUST NOT add additional targets to the target set if the Request-URI of the original request does not indicate a resource this proxy is responsible for.

> A proxy can only change the Request-URI of a request during forwarding if it is responsible for that URI. If the proxy is not responsible for that URI, it will not recurse on 3xx or 416 responses as described below.

If the Request-URI of the original request indicates a resource this proxy is responsible for, the proxy MAY continue to add targets to the set after beginning Request Forwarding. It MAY use any information obtained during that processing to determine new targets. For instance, a proxy may choose to incorporate contacts obtained in a redirect response (3xx) into the target set. If a proxy uses a dynamic source of information while building the target set (for instance, if it consults a SIP Registrar), it SHOULD monitor that source for the duration of processing the request. New locations SHOULD be added to the target set as they become available. As above, any given URI MUST NOT be added to the set more than once.

> Allowing a URI to be added to the set only once reduces unnecessary network traffic, and in the case of incorporating contacts from redirect requests prevents infinite recursion.

For example, a trivial location service is a "no-op", where the target URI is equal to the incoming request URI. The request is sent to a specific next hop proxy for further processing. During request

forwarding of Section 16.6, Item 6, the identity of that next hop, expressed as a SIP or SIPS URI, is inserted as the top-most Route header field value into the request.

If the Request-URI indicates a resource at this proxy that does not exist, the proxy MUST return a 404 (Not Found) response.

If the target set remains empty after applying all of the above, the proxy MUST return an error response, which SHOULD be the 480 (Temporarily Unavailable) response.

16.6 Request Forwarding

As soon as the target set is non-empty, a proxy MAY begin forwarding the request. A stateful proxy MAY process the set in any order. It MAY process multiple targets serially, allowing each client transaction to complete before starting the next. It MAY start client transactions with every target in parallel. It also MAY arbitrarily divide the set into groups, processing the groups serially and processing the targets in each group in parallel.

A common ordering mechanism is to use the qvalue parameter of targets obtained from Contact header fields (see Section 20.10). Targets are processed from highest qvalue to lowest. Targets with equal qvalues may be processed in parallel.

A stateful proxy must have a mechanism to maintain the target set as responses are received and associate the responses to each forwarded request with the original request. For the purposes of this model, this mechanism is a "response context" created by the proxy layer before forwarding the first request.

For each target, the proxy forwards the request following these steps:

1. Make a copy of the received request
2. Update the Request-URI
3. Update the Max-Forwards header field
4. Optionally add a Record-route header field value
5. Optionally add additional header fields
6. Postprocess routing information
7. Determine the next-hop address, port, and transport

8. Add a Via header field value

9. Add a Content-Length header field if necessary

10. Forward the new request

11. Set timer C

Each of these steps is detailed below:

1. Copy request

   The proxy starts with a copy of the received request. The copy MUST initially contain all of the header fields from the received request. Fields not detailed in the processing described below MUST NOT be removed. The copy SHOULD maintain the ordering of the header fields as in the received request. The proxy MUST NOT reorder field values with a common field name (See Section 7.3.1). The proxy MUST NOT add to, modify, or remove the message body.

   An actual implementation need not perform a copy; the primary requirement is that the processing for each next hop begin with the same request.

2. Request-URI

   The Request-URI in the copy's start line MUST be replaced with the URI for this target. If the URI contains any parameters not allowed in a Request-URI, they MUST be removed.

   This is the essence of a proxy's role. This is the mechanism through which a proxy routes a request toward its destination.

   In some circumstances, the received Request-URI is placed into the target set without being modified. For that target, the replacement above is effectively a no-op.

3. Max-Forwards

   If the copy contains a Max-Forwards header field, the proxy MUST decrement its value by one (1).

   If the copy does not contain a Max-Forwards header field, the proxy MUST add one with a field value, which SHOULD be 70.

   Some existing UAs will not provide a Max-Forwards header field in a request.

4. Record-Route

If this proxy wishes to remain on the path of future requests in a dialog created by this request (assuming the request creates a dialog), it MUST insert a Record-Route header field value into the copy before any existing Record-Route header field values, even if a Route header field is already present.

Requests establishing a dialog may contain a preloaded Route header field.

If this request is already part of a dialog, the proxy SHOULD insert a Record-Route header field value if it wishes to remain on the path of future requests in the dialog. In normal endpoint operation as described in Section 12, these Record-Route header field values will not have any effect on the route sets used by the endpoints.

The proxy will remain on the path if it chooses to not insert a Record-Route header field value into requests that are already part of a dialog. However, it would be removed from the path when an endpoint that has failed reconstitutes the dialog.

A proxy MAY insert a Record-Route header field value into any request. If the request does not initiate a dialog, the endpoints will ignore the value. See Section 12 for details on how endpoints use the Record-Route header field values to construct Route header fields.

Each proxy in the path of a request chooses whether to add a Record-Route header field value independently - the presence of a Record-Route header field in a request does not obligate this proxy to add a value.

The URI placed in the Record-Route header field value MUST be a SIP or SIPS URI. This URI MUST contain an lr parameter (see Section 19.1.1). This URI MAY be different for each destination the request is forwarded to. The URI SHOULD NOT contain the transport parameter unless the proxy has knowledge (such as in a private network) that the next downstream element that will be in the path of subsequent requests supports that transport.

The URI this proxy provides will be used by some other element to make a routing decision. This proxy, in general, has no way of knowing the capabilities of that element, so it must restrict itself to the mandatory elements of a SIP implementation: SIP URIs and either the TCP or UDP transports.

The URI placed in the Record-Route header field MUST resolve to the element inserting it (or a suitable stand-in) when the server location procedures of [4] are applied to it, so that subsequent requests reach the same SIP element. If the Request-URI contains a SIPS URI, or the topmost Route header field value (after the post processing of bullet 6) contains a SIPS URI, the URI placed into the Record-Route header field MUST be a SIPS URI. Furthermore, if the request was not received over TLS, the proxy MUST insert a Record-Route header field. In a similar fashion, a proxy that receives a request over TLS, but generates a request without a SIPS URI in the Request-URI or topmost Route header field value (after the post processing of bullet 6), MUST insert a Record-Route header field that is not a SIPS URI.

A proxy at a security perimeter must remain on the perimeter throughout the dialog.

If the URI placed in the Record-Route header field needs to be rewritten when it passes back through in a response, the URI MUST be distinct enough to locate at that time. (The request may spiral through this proxy, resulting in more than one Record-Route header field value being added). Item 8 of Section 16.7 recommends a mechanism to make the URI sufficiently distinct.

The proxy MAY include parameters in the Record-Route header field value. These will be echoed in some responses to the request such as the 200 (OK) responses to INVITE. Such parameters may be useful for keeping state in the message rather than the proxy.

If a proxy needs to be in the path of any type of dialog (such as one straddling a firewall), it SHOULD add a Record-Route header field value to every request with a method it does not understand since that method may have dialog semantics.

The URI a proxy places into a Record-Route header field is only valid for the lifetime of any dialog created by the transaction in which it occurs. A dialog-stateful proxy, for example, MAY refuse to accept future requests with that value in the Request-URI after the dialog has terminated. Non-dialog-stateful proxies, of course, have no concept of when the dialog has terminated, but they MAY encode enough information in the value to compare it against the dialog identifier of future requests and MAY reject requests not matching that information. Endpoints MUST NOT use a URI obtained from a Record-Route header field outside the dialog in which it was provided. See

Section 12 for more information on an endpoint's use of Record-Route header fields.

Record-routing may be required by certain services where the proxy needs to observe all messages in a dialog. However, it slows down processing and impairs scalability and thus proxies should only record-route if required for a particular service.

The Record-Route process is designed to work for any SIP request that initiates a dialog. INVITE is the only such request in this specification, but extensions to the protocol MAY define others.

5. Add Additional Header Fields

The proxy MAY add any other appropriate header fields to the copy at this point.

6. Postprocess routing information

A proxy MAY have a local policy that mandates that a request visit a specific set of proxies before being delivered to the destination. A proxy MUST ensure that all such proxies are loose routers. Generally, this can only be known with certainty if the proxies are within the same administrative domain. This set of proxies is represented by a set of URIs (each of which contains the lr parameter). This set MUST be pushed into the Route header field of the copy ahead of any existing values, if present. If the Route header field is absent, it MUST be added, containing that list of URIs.

If the proxy has a local policy that mandates that the request visit one specific proxy, an alternative to pushing a Route value into the Route header field is to bypass the forwarding logic of item 10 below, and instead just send the request to the address, port, and transport for that specific proxy. If the request has a Route header field, this alternative MUST NOT be used unless it is known that next hop proxy is a loose router. Otherwise, this approach MAY be used, but the Route insertion mechanism above is preferred for its robustness, flexibility, generality and consistency of operation. Furthermore, if the Request-URI contains a SIPS URI, TLS MUST be used to communicate with that proxy.

If the copy contains a Route header field, the proxy MUST inspect the URI in its first value. If that URI does not contain an lr parameter, the proxy MUST modify the copy as follows:

- The proxy MUST place the Request-URI into the Route header field as the last value.

- The proxy MUST then place the first Route header field value into the Request-URI and remove that value from the Route header field.

Appending the Request-URI to the Route header field is part of a mechanism used to pass the information in that Request-URI through strict-routing elements. "Popping" the first Route header field value into the Request-URI formats the message the way a strict-routing element expects to receive it (with its own URI in the Request-URI and the next location to visit in the first Route header field value).

7. Determine Next-Hop Address, Port, and Transport

The proxy MAY have a local policy to send the request to a specific IP address, port, and transport, independent of the values of the Route and Request-URI. Such a policy MUST NOT be used if the proxy is not certain that the IP address, port, and transport correspond to a server that is a loose router. However, this mechanism for sending the request through a specific next hop is NOT RECOMMENDED; instead a Route header field should be used for that purpose as described above.

In the absence of such an overriding mechanism, the proxy applies the procedures listed in [4] as follows to determine where to send the request. If the proxy has reformatted the request to send to a strict-routing element as described in step 6 above, the proxy MUST apply those procedures to the Request-URI of the request. Otherwise, the proxy MUST apply the procedures to the first value in the Route header field, if present, else the Request-URI. The procedures will produce an ordered set of (address, port, transport) tuples. Independently of which URI is being used as input to the procedures of [4], if the Request-URI specifies a SIPS resource, the proxy MUST follow the procedures of [4] as if the input URI were a SIPS URI.

As described in [4], the proxy MUST attempt to deliver the message to the first tuple in that set, and proceed through the set in order until the delivery attempt succeeds.

For each tuple attempted, the proxy MUST format the message as appropriate for the tuple and send the request using a new client transaction as detailed in steps 8 through 10.

Since each attempt uses a new client transaction, it represents a new branch. Thus, the branch parameter provided with the Via header field inserted in step 8 MUST be different for each attempt.

If the client transaction reports failure to send the request or a timeout from its state machine, the proxy continues to the next address in that ordered set. If the ordered set is exhausted, the request cannot be forwarded to this element in the target set. The proxy does not need to place anything in the response context, but otherwise acts as if this element of the target set returned a 408 (Request Timeout) final response.

8. Add a Via header field value

The proxy MUST insert a Via header field value into the copy before the existing Via header field values. The construction of this value follows the same guidelines of Section 8.1.1.7. This implies that the proxy will compute its own branch parameter, which will be globally unique for that branch, and contain the requisite magic cookie. Note that this implies that the branch parameter will be different for different instances of a spiraled or looped request through a proxy.

Proxies choosing to detect loops have an additional constraint in the value they use for construction of the branch parameter. A proxy choosing to detect loops SHOULD create a branch parameter separable into two parts by the implementation. The first part MUST satisfy the constraints of Section 8.1.1.7 as described above. The second is used to perform loop detection and distinguish loops from spirals.

Loop detection is performed by verifying that, when a request returns to a proxy, those fields having an impact on the processing of the request have not changed. The value placed in this part of the branch parameter SHOULD reflect all of those fields (including any Route, Proxy-Require and Proxy-Authorization header fields). This is to ensure that if the request is routed back to the proxy and one of those fields changes, it is treated as a spiral and not a loop (see Section 16.3). A common way to create this value is to compute a cryptographic hash of the To tag, From tag, Call-ID header field, the Request-URI of the request received (before translation), the topmost Via header, and the sequence number from the CSeq header field, in addition to any Proxy-Require and Proxy-Authorization header fields that may be present. The

algorithm used to compute the hash is implementation-dependent, but MD5 (RFC 1321 [35]), expressed in hexadecimal, is a reasonable choice. (Base64 is not permissible for a token.)

If a proxy wishes to detect loops, the "branch" parameter it supplies MUST depend on all information affecting processing of a request, including the incoming Request-URI and any header fields affecting the request's admission or routing. This is necessary to distinguish looped requests from requests whose routing parameters have changed before returning to this server.

The request method MUST NOT be included in the calculation of the branch parameter. In particular, CANCEL and ACK requests (for non-2xx responses) MUST have the same branch value as the corresponding request they cancel or acknowledge. The branch parameter is used in correlating those requests at the server handling them (see Sections 17.2.3 and 9.2).

9. Add a Content-Length header field if necessary

If the request will be sent to the next hop using a stream-based transport and the copy contains no Content-Length header field, the proxy MUST insert one with the correct value for the body of the request (see Section 20.14).

10. Forward Request

A stateful proxy MUST create a new client transaction for this request as described in Section 17.1 and instructs the transaction to send the request using the address, port and transport determined in step 7.

11. Set timer C

In order to handle the case where an INVITE request never generates a final response, the TU uses a timer which is called timer C. Timer C MUST be set for each client transaction when an INVITE request is proxied. The timer MUST be larger than 3 minutes. Section 16.7 bullet 2 discusses how this timer is updated with provisional responses, and Section 16.8 discusses processing when it fires.

16.7 Response Processing

When a response is received by an element, it first tries to locate a client transaction (Section 17.1.3) matching the response. If none is found, the element MUST process the response (even if it is an informational response) as a stateless proxy (described below). If a match is found, the response is handed to the client transaction.

> Forwarding responses for which a client transaction (or more generally any knowledge of having sent an associated request) is not found improves robustness. In particular, it ensures that "late" 2xx responses to INVITE requests are forwarded properly.

As client transactions pass responses to the proxy layer, the following processing MUST take place:

1. Find the appropriate response context
2. Update timer C for provisional responses
3. Remove the topmost Via
4. Add the response to the response context
5. Check to see if this response should be forwarded immediately
6. When necessary, choose the best final response from the response context

If no final response has been forwarded after every client transaction associated with the response context has been terminated, the proxy must choose and forward the "best" response from those it has seen so far.

The following processing MUST be performed on each response that is forwarded. It is likely that more than one response to each request will be forwarded: at least each provisional and one final response.

7. Aggregate authorization header field values if necessary
8. Optionally rewrite Record-Route header field values
9. Forward the response
10. Generate any necessary CANCEL requests

Each of the above steps are detailed below:

1. Find Context

   The proxy locates the "response context" it created before forwarding the original request using the key described in Section 16.6. The remaining processing steps take place in this context.

2. Update timer C for provisional responses

   For an INVITE transaction, if the response is a provisional response with status codes 101 to 199 inclusive (i.e., anything but 100), the proxy MUST reset timer C for that client transaction. The timer MAY be reset to a different value, but this value MUST be greater than 3 minutes.

3. Via

   The proxy removes the topmost Via header field value from the response.

   If no Via header field values remain in the response, the response was meant for this element and MUST NOT be forwarded. The remainder of the processing described in this section is not performed on this message, the UAC processing rules described in Section 8.1.3 are followed instead (transport layer processing has already occurred).

   This will happen, for instance, when the element generates CANCEL requests as described in Section 10.

4. Add response to context

   Final responses received are stored in the response context until a final response is generated on the server transaction associated with this context. The response may be a candidate for the best final response to be returned on that server transaction. Information from this response may be needed in forming the best response, even if this response is not chosen.

   If the proxy chooses to recurse on any contacts in a 3xx response by adding them to the target set, it MUST remove them from the response before adding the response to the response context. However, a proxy SHOULD NOT recurse to a non-SIPS URI if the Request-URI of the original request was a SIPS URI. If

the proxy recurses on all of the contacts in a 3xx response, the proxy SHOULD NOT add the resulting contactless response to the response context.

Removing the contact before adding the response to the response context prevents the next element upstream from retrying a location this proxy has already attempted.

3xx responses may contain a mixture of SIP, SIPS, and non-SIP URIs. A proxy may choose to recurse on the SIP and SIPS URIs and place the remainder into the response context to be returned, potentially in the final response.

If a proxy receives a 416 (Unsupported URI Scheme) response to a request whose Request-URI scheme was not SIP, but the scheme in the original received request was SIP or SIPS (that is, the proxy changed the scheme from SIP or SIPS to something else when it proxied a request), the proxy SHOULD add a new URI to the target set. This URI SHOULD be a SIP URI version of the non-SIP URI that was just tried. In the case of the tel URL, this is accomplished by placing the telephone-subscriber part of the tel URL into the user part of the SIP URI, and setting the hostpart to the domain where the prior request was sent. See Section 19.1.6 for more detail on forming SIP URIs from tel URLs.

As with a 3xx response, if a proxy "recurses" on the 416 by trying a SIP or SIPS URI instead, the 416 response SHOULD NOT be added to the response context.

5. Check response for forwarding

Until a final response has been sent on the server transaction, the following responses MUST be forwarded immediately:

- Any provisional response other than 100 (Trying)

- Any 2xx response

If a 6xx response is received, it is not immediately forwarded, but the stateful proxy SHOULD cancel all client pending transactions as described in Section 10, and it MUST NOT create any new branches in this context.

This is a change from RFC 2543, which mandated that the proxy was to forward the 6xx response immediately. For an INVITE transaction, this approach had the problem that a 2xx response could arrive on another branch, in which case the proxy would

have to forward the 2xx. The result was that the UAC could receive a 6xx response followed by a 2xx response, which should never be allowed to happen. Under the new rules, upon receiving a 6xx, a proxy will issue a CANCEL request, which will generally result in 487 responses from all outstanding client transactions, and then at that point the 6xx is forwarded upstream.

After a final response has been sent on the server transaction, the following responses MUST be forwarded immediately:

- Any 2xx response to an INVITE request

A stateful proxy MUST NOT immediately forward any other responses. In particular, a stateful proxy MUST NOT forward any 100 (Trying) response. Those responses that are candidates for forwarding later as the "best" response have been gathered as described in step "Add Response to Context".

Any response chosen for immediate forwarding MUST be processed as described in steps "Aggregate Authorization Header Field Values" through "Record-Route".

This step, combined with the next, ensures that a stateful proxy will forward exactly one final response to a non-INVITE request, and either exactly one non-2xx response or one or more 2xx responses to an INVITE request.

6. Choosing the best response

A stateful proxy MUST send a final response to a response context's server transaction if no final responses have been immediately forwarded by the above rules and all client transactions in this response context have been terminated.

The stateful proxy MUST choose the "best" final response among those received and stored in the response context.

If there are no final responses in the context, the proxy MUST send a 408 (Request Timeout) response to the server transaction.

Otherwise, the proxy MUST forward a response from the responses stored in the response context. It MUST choose from the 6xx class responses if any exist in the context. If no 6xx class responses are present, the proxy SHOULD choose from the lowest response class stored in the response context. The proxy MAY select any response within that chosen class. The proxy SHOULD

give preference to responses that provide information affecting resubmission of this request, such as 401, 407, 415, 420, and 484 if the 4xx class is chosen.

A proxy which receives a 503 (Service Unavailable) response SHOULD NOT forward it upstream unless it can determine that any subsequent requests it might proxy will also generate a 503. In other words, forwarding a 503 means that the proxy knows it cannot service any requests, not just the one for the Request-URI in the request which generated the 503. If the only response that was received is a 503, the proxy SHOULD generate a 500 response and forward that upstream.

The forwarded response MUST be processed as described in steps "Aggregate Authorization Header Field Values" through "Record-Route".

For example, if a proxy forwarded a request to 4 locations, and received 503, 407, 501, and 404 responses, it may choose to forward the 407 (Proxy Authentication Required) response.

1xx and 2xx responses may be involved in the establishment of dialogs. When a request does not contain a To tag, the To tag in the response is used by the UAC to distinguish multiple responses to a dialog creating request. A proxy MUST NOT insert a tag into the To header field of a 1xx or 2xx response if the request did not contain one. A proxy MUST NOT modify the tag in the To header field of a 1xx or 2xx response.

Since a proxy may not insert a tag into the To header field of a 1xx response to a request that did not contain one, it cannot issue non-100 provisional responses on its own. However, it can branch the request to a UAS sharing the same element as the proxy. This UAS can return its own provisional responses, entering into an early dialog with the initiator of the request. The UAS does not have to be a discreet process from the proxy. It could be a virtual UAS implemented in the same code space as the proxy.

3-6xx responses are delivered hop-by-hop. When issuing a 3-6xx response, the element is effectively acting as a UAS, issuing its own response, usually based on the responses received from downstream elements. An element SHOULD preserve the To tag when simply forwarding a 3-6xx response to a request that did not contain a To tag.

A proxy MUST NOT modify the To tag in any forwarded response to a request that contains a To tag.

While it makes no difference to the upstream elements if the proxy replaced the To tag in a forwarded 3-6xx response, preserving the original tag may assist with debugging.

When the proxy is aggregating information from several responses, choosing a To tag from among them is arbitrary, and generating a new To tag may make debugging easier. This happens, for instance, when combining 401 (Unauthorized) and 407 (Proxy Authentication Required) challenges, or combining Contact values from unencrypted and unauthenticated 3xx responses.

7. Aggregate Authorization Header Field Values

If the selected response is a 401 (Unauthorized) or 407 (Proxy Authentication Required), the proxy MUST collect any WWW-Authenticate and Proxy-Authenticate header field values from all other 401 (Unauthorized) and 407 (Proxy Authentication Required) responses received so far in this response context and add them to this response without modification before forwarding. The resulting 401 (Unauthorized) or 407 (Proxy Authentication Required) response could have several WWW-Authenticate AND Proxy-Authenticate header field values.

This is necessary because any or all of the destinations the request was forwarded to may have requested credentials. The client needs to receive all of those challenges and supply credentials for each of them when it retries the request. Motivation for this behavior is provided in Section 26.

8. Record-Route

If the selected response contains a Record-Route header field value originally provided by this proxy, the proxy MAY choose to rewrite the value before forwarding the response. This allows the proxy to provide different URIs for itself to the next upstream and downstream elements. A proxy may choose to use this mechanism for any reason. For instance, it is useful for multi-homed hosts.

If the proxy received the request over TLS, and sent it out over a non-TLS connection, the proxy MUST rewrite the URI in the Record-Route header field to be a SIPS URI. If the proxy received the request over a non-TLS connection, and sent it out over TLS, the proxy MUST rewrite the URI in the Record-Route header field to be a SIP URI.

The new URI provided by the proxy MUST satisfy the same constraints on URIs placed in Record-Route header fields in requests (see Step 4 of Section 16.6) with the following modifications:

The URI SHOULD NOT contain the transport parameter unless the proxy has knowledge that the next upstream (as opposed to downstream) element that will be in the path of subsequent requests supports that transport.

When a proxy does decide to modify the Record-Route header field in the response, one of the operations it performs is locating the Record-Route value that it had inserted. If the request spiraled, and the proxy inserted a Record-Route value in each iteration of the spiral, locating the correct value in the response (which must be the proper iteration in the reverse direction) is tricky. The rules above recommend that a proxy wishing to rewrite Record-Route header field values insert sufficiently distinct URIs into the Record-Route header field so that the right one may be selected for rewriting. A RECOMMENDED mechanism to achieve this is for the proxy to append a unique identifier for the proxy instance to the user portion of the URI.

When the response arrives, the proxy modifies the first Record-Route whose identifier matches the proxy instance. The modification results in a URI without this piece of data appended to the user portion of the URI. Upon the next iteration, the same algorithm (find the topmost Record-Route header field value with the parameter) will correctly extract the next Record-Route header field value inserted by that proxy.

Not every response to a request to which a proxy adds a Record-Route header field value will contain a Record-Route header field. If the response does contain a Record-Route header field, it will contain the value the proxy added.

9. Forward response

After performing the processing described in steps "Aggregate Authorization Header Field Values" through "Record-Route", the proxy MAY perform any feature specific manipulations on the selected response. The proxy MUST NOT add to, modify, or remove the message body. Unless otherwise specified, the proxy MUST NOT remove any header field values other than the Via header field value discussed in Section 16.7 Item 3. In particular, the proxy MUST NOT remove any "received" parameter

it may have added to the next Via header field value while processing the request associated with this response. The proxy MUST pass the response to the server transaction associated with the response context. This will result in the response being sent to the location now indicated in the topmost Via header field value. If the server transaction is no longer available to handle the transmission, the element MUST forward the response statelessly by sending it to the server transport. The server transaction might indicate failure to send the response or signal a timeout in its state machine. These errors would be logged for diagnostic purposes as appropriate, but the protocol requires no remedial action from the proxy.

The proxy MUST maintain the response context until all of its associated transactions have been terminated, even after forwarding a final response.

10. Generate CANCELs

If the forwarded response was a final response, the proxy MUST generate a CANCEL request for all pending client transactions associated with this response context. A proxy SHOULD also generate a CANCEL request for all pending client transactions associated with this response context when it receives a 6xx response. A pending client transaction is one that has received a provisional response, but no final response (it is in the proceeding state) and has not had an associated CANCEL generated for it. Generating CANCEL requests is described in Section 9.1.

The requirement to CANCEL pending client transactions upon forwarding a final response does not guarantee that an endpoint will not receive multiple 200 (OK) responses to an INVITE. 200 (OK) responses on more than one branch may be generated before the CANCEL requests can be sent and processed. Further, it is reasonable to expect that a future extension may override this requirement to issue CANCEL requests.

16.8 Processing Timer C

If timer C should fire, the proxy MUST either reset the timer with any value it chooses, or terminate the client transaction. If the client transaction has received a provisional response, the proxy MUST generate a CANCEL request matching that transaction. If the client transaction has not received a provisional response, the proxy MUST behave as if the transaction received a 408 (Request Timeout) response.

Allowing the proxy to reset the timer allows the proxy to dynamically extend the transaction's lifetime based on current conditions (such as utilization) when the timer fires.

16.9 Handling Transport Errors

If the transport layer notifies a proxy of an error when it tries to forward a request (see Section 18.4), the proxy MUST behave as if the forwarded request received a 503 (Service Unavailable) response.

If the proxy is notified of an error when forwarding a response, it drops the response. The proxy SHOULD NOT cancel any outstanding client transactions associated with this response context due to this notification.

> If a proxy cancels its outstanding client transactions, a single malicious or misbehaving client can cause all transactions to fail through its Via header field.

16.10 CANCEL Processing

A stateful proxy MAY generate a CANCEL to any other request it has generated at any time (subject to receiving a provisional response to that request as described in section 9.1). A proxy MUST cancel any pending client transactions associated with a response context when it receives a matching CANCEL request.

A stateful proxy MAY generate CANCEL requests for pending INVITE client transactions based on the period specified in the INVITE's Expires header field elapsing. However, this is generally unnecessary since the endpoints involved will take care of signaling the end of the transaction.

While a CANCEL request is handled in a stateful proxy by its own server transaction, a new response context is not created for it. Instead, the proxy layer searches its existing response contexts for the server transaction handling the request associated with this CANCEL. If a matching response context is found, the element MUST immediately return a 200 (OK) response to the CANCEL request. In this case, the element is acting as a user agent server as defined in Section 8.2. Furthermore, the element MUST generate CANCEL requests for all pending client transactions in the context as described in Section 16.7 step 10.

If a response context is not found, the element does not have any knowledge of the request to apply the CANCEL to. It MUST statelessly forward the CANCEL request (it may have statelessly forwarded the associated request previously).

16.11 Stateless Proxy

When acting statelessly, a proxy is a simple message forwarder. Much of the processing performed when acting statelessly is the same as when behaving statefully. The differences are detailed here.

A stateless proxy does not have any notion of a transaction, or of the response context used to describe stateful proxy behavior. Instead, the stateless proxy takes messages, both requests and responses, directly from the transport layer (See section 18). As a result, stateless proxies do not retransmit messages on their own. They do, however, forward all retransmissions they receive (they do not have the ability to distinguish a retransmission from the original message). Furthermore, when handling a request statelessly, an element MUST NOT generate its own 100 (Trying) or any other provisional response.

A stateless proxy MUST validate a request as described in Section 16.3

A stateless proxy MUST follow the request processing steps described in Sections 16.4 through 16.5 with the following exception:

- A stateless proxy MUST choose one and only one target from the target set. This choice MUST only rely on fields in the message and time-invariant properties of the server. In particular, a retransmitted request MUST be forwarded to the same destination each time it is processed. Furthermore, CANCEL and non-Routed ACK requests MUST generate the same choice as their associated INVITE.

A stateless proxy MUST follow the request processing steps described in Section 16.6 with the following exceptions:

- The requirement for unique branch IDs across space and time applies to stateless proxies as well. However, a stateless proxy cannot simply use a random number generator to compute the first component of the branch ID, as described in Section 16.6 bullet 8. This is because retransmissions of a request need to have the same value, and a stateless proxy cannot tell a retransmission from the original request. Therefore, the component of the branch parameter that makes it unique MUST be the same each time a retransmitted request is forwarded. Thus for a stateless proxy, the branch parameter MUST be computed as a combinatoric function of message parameters which are invariant on retransmission.

The stateless proxy MAY use any technique it likes to guarantee uniqueness of its branch IDs across transactions. However, the following procedure is RECOMMENDED. The proxy examines the branch ID in the topmost Via header field of the received request. If it begins with the magic cookie, the first component of the branch ID of the outgoing request is computed as a hash of the received branch ID. Otherwise, the first component of the branch ID is computed as a hash of the topmost Via, the tag in the To header field, the tag in the From header field, the Call-ID header field, the CSeq number (but not method), and the Request-URI from the received request. One of these fields will always vary across two different transactions.

- All other message transformations specified in Section 16.6 MUST result in the same transformation of a retransmitted request. In particular, if the proxy inserts a Record-Route value or pushes URIs into the Route header field, it MUST place the same values in retransmissions of the request. As for the Via branch parameter, this implies that the transformations MUST be based on time-invariant configuration or retransmission-invariant properties of the request.

- A stateless proxy determines where to forward the request as described for stateful proxies in Section 16.6 Item 10. The request is sent directly to the transport layer instead of through a client transaction.

  Since a stateless proxy must forward retransmitted requests to the same destination and add identical branch parameters to each of them, it can only use information from the message itself and time-invariant configuration data for those calculations. If the configuration state is not time-invariant (for example, if a routing table is updated) any requests that could be affected by the change may not be forwarded statelessly during an interval equal to the transaction timeout window before or after the change. The method of processing the affected requests in that interval is an implementation decision. A common solution is to forward them transaction statefully.

Stateless proxies MUST NOT perform special processing for CANCEL requests. They are processed by the above rules as any other requests. In particular, a stateless proxy applies the same Route header field processing to CANCEL requests that it applies to any other request.

Response processing as described in Section 16.7 does not apply to a proxy behaving statelessly. When a response arrives at a stateless proxy, the proxy MUST inspect the sent-by value in the first (topmost) Via header field value. If that address matches the proxy, (it equals a value this proxy has inserted into previous requests) the proxy MUST remove that header field value from the response and forward the result to the location indicated in the next Via header field value. The proxy MUST NOT add to, modify, or remove the message body. Unless specified otherwise, the proxy MUST NOT remove any other header field values. If the address does not match the proxy, the message MUST be silently discarded.

16.12 Summary of Proxy Route Processing

In the absence of local policy to the contrary, the processing a proxy performs on a request containing a Route header field can be summarized in the following steps.

1. The proxy will inspect the Request-URI. If it indicates a resource owned by this proxy, the proxy will replace it with the results of running a location service. Otherwise, the proxy will not change the Request-URI.

2. The proxy will inspect the URI in the topmost Route header field value. If it indicates this proxy, the proxy removes it from the Route header field (this route node has been reached).

3. The proxy will forward the request to the resource indicated by the URI in the topmost Route header field value or in the Request-URI if no Route header field is present. The proxy determines the address, port and transport to use when forwarding the request by applying the procedures in [4] to that URI.

If no strict-routing elements are encountered on the path of the request, the Request-URI will always indicate the target of the request.

16.12.1 Examples

16.12.1.1 Basic SIP Trapezoid

This scenario is the basic SIP trapezoid, U1 -> P1 -> P2 -> U2, with both proxies record-routing. Here is the flow.

U1 sends:

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
```

to P1. P1 is an outbound proxy. P1 is not responsible for domain.com, so it looks it up in DNS and sends it there. It also adds a Record-Route header field value:

```
INVITE sip:callee@domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p1.example.com;lr>
```

P2 gets this. It is responsible for domain.com so it runs a location service and rewrites the Request-URI. It also adds a Record-Route header field value. There is no Route header field, so it resolves the new Request-URI to determine where to send the request:

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

The callee at u2.domain.com gets this and responds with a 200 OK:

```
SIP/2.0 200 OK
Contact: sip:callee@u2.domain.com
Record-Route: <sip:p2.domain.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

The callee at u2 also sets its dialog state's remote target URI to sip:caller@u1.example.com and its route set to:

```
(<sip:p2.domain.com;lr>,<sip:p1.example.com;lr>)
```

This is forwarded by P2 to P1 to U1 as normal. Now, U1 sets its dialog state's remote target URI to sip:callee@u2.domain.com and its route set to:

```
(<sip:p1.example.com;lr>,<sip:p2.domain.com;lr>)
```

Since all the route set elements contain the lr parameter, U1 constructs the following BYE request:

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p1.example.com;lr>,<sip:p2.domain.com;lr>
```

As any other element (including proxies) would do, it resolves the URI in the topmost Route header field value using DNS to determine where to send the request. This goes to P1. P1 notices that it is not responsible for the resource indicated in the Request-URI so it doesn't change it. It does see that it is the first value in the Route header field, so it removes that value, and forwards the request to P2:

```
BYE sip:callee@u2.domain.com SIP/2.0
Route: <sip:p2.domain.com;lr>
```

P2 also notices it is not responsible for the resource indicated by the Request-URI (it is responsible for domain.com, not u2.domain.com), so it doesn't change it. It does see itself in the first Route header field value, so it removes it and forwards the following to u2.domain.com based on a DNS lookup against the Request-URI:

```
BYE sip:callee@u2.domain.com SIP/2.0
```

16.12.1.2 Traversing a Strict-Routing Proxy

In this scenario, a dialog is established across four proxies, each of which adds Record-Route header field values. The third proxy implements the strict-routing procedures specified in RFC 2543 and many works in progress.

```
U1->P1->P2->P3->P4->U2
```

The INVITE arriving at U2 contains:

```
INVITE sip:callee@u2.domain.com SIP/2.0
Contact: sip:caller@u1.example.com
Record-Route: <sip:p4.domain.com;lr>
Record-Route: <sip:p3.middle.com>
Record-Route: <sip:p2.example.com;lr>
Record-Route: <sip:p1.example.com;lr>
```

Which U2 responds to with a 200 OK. Later, U2 sends the following BYE request to P4 based on the first Route header field value.

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p4.domain.com;lr>
Route: <sip:p3.middle.com>
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
```

P4 is not responsible for the resource indicated in the Request-URI so it will leave it alone. It notices that it is the element in the first Route header field value so it removes it. It then prepares to send the request based on the now first Route header field value of sip:p3.middle.com, but it notices that this URI does not contain the lr parameter, so before sending, it reformats the request to be:

```
BYE sip:p3.middle.com SIP/2.0
Route: <sip:p2.example.com;lr>
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P3 is a strict router, so it forwards the following to P2:

```
BYE sip:p2.example.com;lr SIP/2.0
Route: <sip:p1.example.com;lr>
Route: <sip:caller@u1.example.com>
```

P2 sees the request-URI is a value it placed into a Record-Route header field, so before further processing, it rewrites the request to be:

```
BYE sip:caller@u1.example.com SIP/2.0
Route: <sip:p1.example.com;lr>
```

P2 is not responsible for u1.example.com, so it sends the request to P1 based on the resolution of the Route header field value.

P1 notices itself in the topmost Route header field value, so it removes it, resulting in:

```
BYE sip:caller@u1.example.com SIP/2.0
```

Since P1 is not responsible for u1.example.com and there is no Route header field, P1 will forward the request to u1.example.com based on the Request-URI.

16.12.1.3 Rewriting Record-Route Header Field Values

In this scenario, U1 and U2 are in different private namespaces and they enter a dialog through a proxy P1, which acts as a gateway between the namespaces.

```
U1->P1->U2
```

U1 sends:

```
INVITE sip:callee@gateway.leftprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
```

P1 uses its location service and sends the following to U2:

```
INVITE sip:callee@rightprivatespace.com SIP/2.0
Contact: <sip:caller@u1.leftprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

U2 sends this 200 (OK) back to P1:

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.rightprivatespace.com;lr>
```

P1 rewrites its Record-Route header parameter to provide a value that U1 will find useful, and sends the following to U1:

```
SIP/2.0 200 OK
Contact: <sip:callee@u2.rightprivatespace.com>
Record-Route: <sip:gateway.leftprivatespace.com;lr>
```

Later, U1 sends the following BYE request to P1:

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
Route: <sip:gateway.leftprivatespace.com;lr>
```

which P1 forwards to U2 as:

```
BYE sip:callee@u2.rightprivatespace.com SIP/2.0
```

17 Transactions

SIP is a transactional protocol: interactions between components take place in a series of independent message exchanges. Specifically, a SIP transaction consists of a single request and any responses to that request, which include zero or more provisional responses and one or more final responses. In the case of a transaction where the request was an INVITE (known as an INVITE transaction), the transaction also includes the ACK only if the final response was not a 2xx response. If the response was a 2xx, the ACK is not considered part of the transaction.

The reason for this separation is rooted in the importance of delivering all 200 (OK) responses to an INVITE to the UAC. To deliver them all to the UAC, the UAS alone takes responsibility

for retransmitting them (see Section 13.3.1.4), and the UAC alone takes responsibility for acknowledging them with ACK (see Section 13.2.2.4). Since this ACK is retransmitted only by the UAC, it is effectively considered its own transaction.

Transactions have a client side and a server side. The client side is known as a client transaction and the server side as a server transaction. The client transaction sends the request, and the server transaction sends the response. The client and server transactions are logical functions that are embedded in any number of elements. Specifically, they exist within user agents and stateful proxy servers. Consider the example in Section 4. In this example, the UAC executes the client transaction, and its outbound proxy executes the server transaction. The outbound proxy also executes a client transaction, which sends the request to a server transaction in the inbound proxy. That proxy also executes a client transaction, which in turn sends the request to a server transaction in the UAS. This is shown in Figure 4.

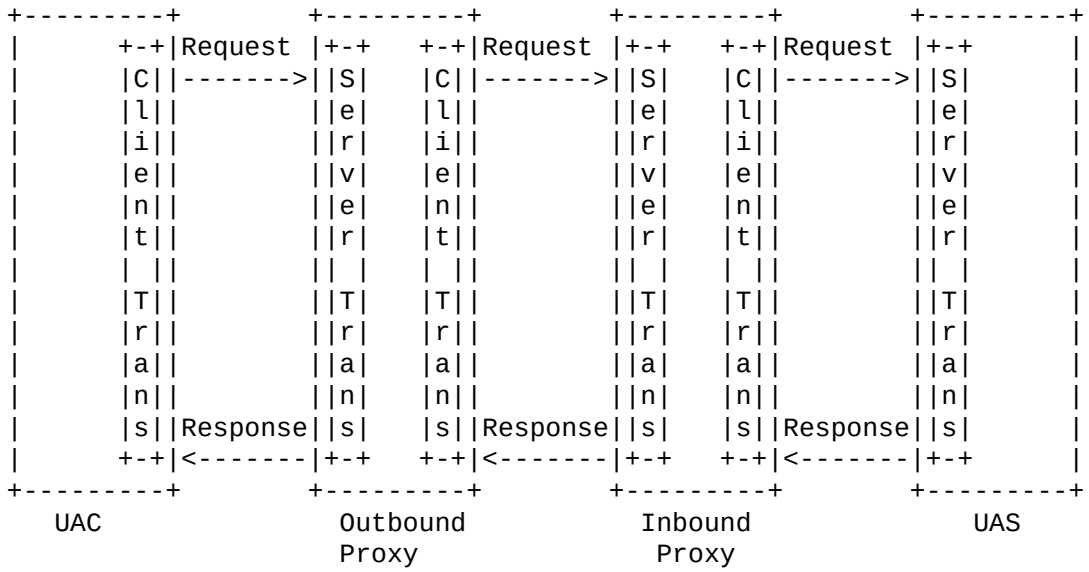


Figure 4: Transaction relationships

A stateless proxy does not contain a client or server transaction. The transaction exists between the UA or stateful proxy on one side, and the UA or stateful proxy on the other side. As far as SIP transactions are concerned, stateless proxies are effectively transparent. The purpose of the client transaction is to receive a request from the element in which the client is embedded (call this element the "Transaction User" or TU; it can be a UA or a stateful proxy), and reliably deliver the request to a server transaction.

The client transaction is also responsible for receiving responses and delivering them to the TU, filtering out any response retransmissions or disallowed responses (such as a response to ACK). Additionally, in the case of an INVITE request, the client transaction is responsible for generating the ACK request for any final response accepting a 2xx response.

Similarly, the purpose of the server transaction is to receive requests from the transport layer and deliver them to the TU. The server transaction filters any request retransmissions from the network. The server transaction accepts responses from the TU and delivers them to the transport layer for transmission over the network. In the case of an INVITE transaction, it absorbs the ACK request for any final response excepting a 2xx response.

The 2xx response and its ACK receive special treatment. This response is retransmitted only by a UAS, and its ACK generated only by the UAC. This end-to-end treatment is needed so that a caller knows the entire set of users that have accepted the call. Because of this special handling, retransmissions of the 2xx response are handled by the UA core, not the transaction layer. Similarly, generation of the ACK for the 2xx is handled by the UA core. Each proxy along the path merely forwards each 2xx response to INVITE and its corresponding ACK.

17.1 Client Transaction

The client transaction provides its functionality through the maintenance of a state machine.

The TU communicates with the client transaction through a simple interface. When the TU wishes to initiate a new transaction, it creates a client transaction and passes it the SIP request to send and an IP address, port, and transport to which to send it. The client transaction begins execution of its state machine. Valid responses are passed up to the TU from the client transaction.

There are two types of client transaction state machines, depending on the method of the request passed by the TU. One handles client transactions for INVITE requests. This type of machine is referred to as an INVITE client transaction. Another type handles client transactions for all requests except INVITE and ACK. This is referred to as a non-INVITE client transaction. There is no client transaction for ACK. If the TU wishes to send an ACK, it passes one directly to the transport layer for transmission.

The INVITE transaction is different from those of other methods because of its extended duration. Normally, human input is required in order to respond to an INVITE. The long delays expected for sending a response argue for a three-way handshake. On the other hand, requests of other methods are expected to complete rapidly. Because of the non-INVITE transaction's reliance on a two-way handshake, TUs SHOULD respond immediately to non-INVITE requests.

17.1.1 INVITE Client Transaction

17.1.1.1 Overview of INVITE Transaction

The INVITE transaction consists of a three-way handshake. The client transaction sends an INVITE, the server transaction sends responses, and the client transaction sends an ACK. For unreliable transports (such as UDP), the client transaction retransmits requests at an interval that starts at T1 seconds and doubles after every retransmission. T1 is an estimate of the round-trip time (RTT), and it defaults to 500 ms. Nearly all of the transaction timers described here scale with T1, and changing T1 adjusts their values. The request is not retransmitted over reliable transports. After receiving a 1xx response, any retransmissions cease altogether, and the client waits for further responses. The server transaction can send additional 1xx responses, which are not transmitted reliably by the server transaction. Eventually, the server transaction decides to send a final response. For unreliable transports, that response is retransmitted periodically, and for reliable transports, it is sent once. For each final response that is received at the client transaction, the client transaction sends an ACK, the purpose of which is to quench retransmissions of the response.

17.1.1.2 Formal Description

The state machine for the INVITE client transaction is shown in Figure 5. The initial state, "calling", MUST be entered when the TU initiates a new client transaction with an INVITE request. The client transaction MUST pass the request to the transport layer for transmission (see Section 18). If an unreliable transport is being used, the client transaction MUST start timer A with a value of T1. If a reliable transport is being used, the client transaction SHOULD NOT start timer A (Timer A controls request retransmissions). For any transport, the client transaction MUST start timer B with a value of 64*T1 seconds (Timer B controls transaction timeouts).

When timer A fires, the client transaction MUST retransmit the request by passing it to the transport layer, and MUST reset the timer with a value of 2*T1. The formal definition of retransmit

within the context of the transaction layer is to take the message previously sent to the transport layer and pass it to the transport layer once more.

When timer A fires 2*T1 seconds later, the request MUST be retransmitted again (assuming the client transaction is still in this state). This process MUST continue so that the request is retransmitted with intervals that double after each transmission. These retransmissions SHOULD only be done while the client transaction is in the "calling" state.

The default value for T1 is 500 ms. T1 is an estimate of the RTT between the client and server transactions. Elements MAY (though it is NOT RECOMMENDED) use smaller values of T1 within closed, private networks that do not permit general Internet connection. T1 MAY be chosen larger, and this is RECOMMENDED if it is known in advance (such as on high latency access links) that the RTT is larger. Whatever the value of T1, the exponential backoffs on retransmissions described in this section MUST be used.

If the client transaction is still in the "Calling" state when timer B fires, the client transaction SHOULD inform the TU that a timeout has occurred. The client transaction MUST NOT generate an ACK. The value of 64*T1 is equal to the amount of time required to send seven requests in the case of an unreliable transport.

If the client transaction receives a provisional response while in the "Calling" state, it transitions to the "Proceeding" state. In the "Proceeding" state, the client transaction SHOULD NOT retransmit the request any longer. Furthermore, the provisional response MUST be passed to the TU. Any further provisional responses MUST be passed up to the TU while in the "Proceeding" state.

When in either the "Calling" or "Proceeding" states, reception of a response with status code from 300-699 MUST cause the client transaction to transition to "Completed". The client transaction MUST pass the received response up to the TU, and the client transaction MUST generate an ACK request, even if the transport is reliable (guidelines for constructing the ACK from the response are given in Section 17.1.1.3) and then pass the ACK to the transport layer for transmission. The ACK MUST be sent to the same address, port, and transport to which the original request was sent. The client transaction SHOULD start timer D when it enters the "Completed" state, with a value of at least 32 seconds for unreliable transports, and a value of zero seconds for reliable transports. Timer D reflects the amount of time that the server transaction can remain in the "Completed" state when unreliable transports are used. This is equal to Timer H in the INVITE server transaction, whose

default is 64*T1. However, the client transaction does not know the value of T1 in use by the server transaction, so an absolute minimum of 32s is used instead of basing Timer D on T1.

Any retransmissions of the final response that are received while in the "Completed" state MUST cause the ACK to be re-passed to the transport layer for retransmission, but the newly received response MUST NOT be passed up to the TU. A retransmission of the response is defined as any response which would match the same client transaction based on the rules of Section 17.1.3.

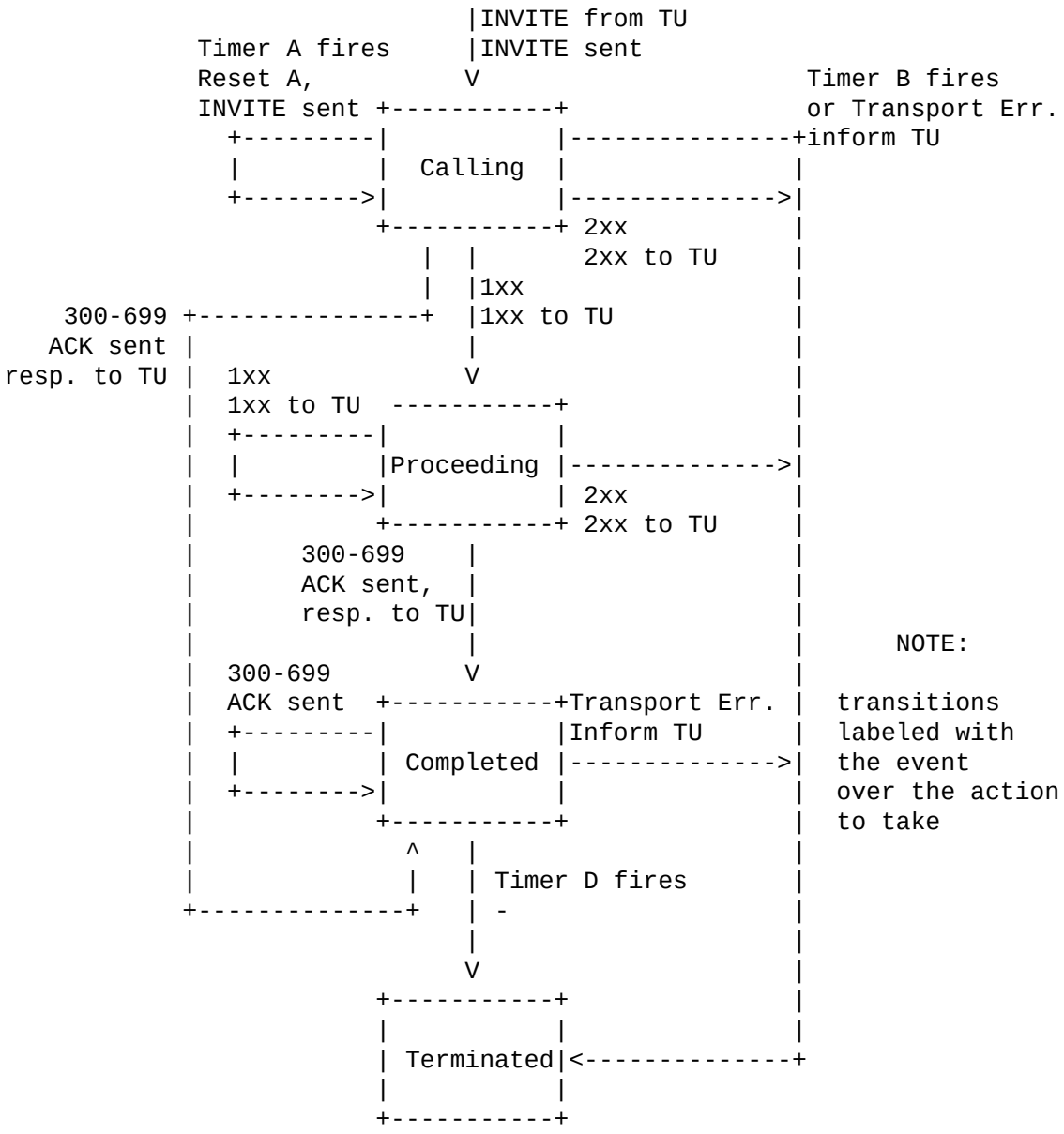


Figure 5: INVITE client transaction

If timer D fires while the client transaction is in the "Completed" state, the client transaction MUST move to the terminated state.

When in either the "Calling" or "Proceeding" states, reception of a 2xx response MUST cause the client transaction to enter the "Terminated" state, and the response MUST be passed up to the TU. The handling of this response depends on whether the TU is a proxy

core or a UAC core. A UAC core will handle generation of the ACK for this response, while a proxy core will always forward the 200 (OK) upstream. The differing treatment of 200 (OK) between proxy and UAC is the reason that handling of it does not take place in the transaction layer.

The client transaction MUST be destroyed the instant it enters the "Terminated" state. This is actually necessary to guarantee correct operation. The reason is that 2xx responses to an INVITE are treated differently; each one is forwarded by proxies, and the ACK handling in a UAC is different. Thus, each 2xx needs to be passed to a proxy core (so that it can be forwarded) and to a UAC core (so it can be acknowledged). No transaction layer processing takes place. Whenever a response is received by the transport, if the transport layer finds no matching client transaction (using the rules of Section 17.1.3), the response is passed directly to the core. Since the matching client transaction is destroyed by the first 2xx, subsequent 2xx will find no match and therefore be passed to the core.

17.1.1.3 Construction of the ACK Request

This section specifies the construction of ACK requests sent within the client transaction. A UAC core that generates an ACK for 2xx MUST instead follow the rules described in Section 13.

The ACK request constructed by the client transaction MUST contain values for the Call-ID, From, and Request-URI that are equal to the values of those header fields in the request passed to the transport by the client transaction (call this the "original request"). The To header field in the ACK MUST equal the To header field in the response being acknowledged, and therefore will usually differ from the To header field in the original request by the addition of the tag parameter. The ACK MUST contain a single Via header field, and this MUST be equal to the top Via header field of the original request. The CSeq header field in the ACK MUST contain the same value for the sequence number as was present in the original request, but the method parameter MUST be equal to "ACK".

If the INVITE request whose response is being acknowledged had Route header fields, those header fields MUST appear in the ACK. This is to ensure that the ACK can be routed properly through any downstream stateless proxies.

Although any request MAY contain a body, a body in an ACK is special since the request cannot be rejected if the body is not understood. Therefore, placement of bodies in ACK for non-2xx is NOT RECOMMENDED, but if done, the body types are restricted to any that appeared in the INVITE, assuming that the response to the INVITE was not 415. If it was, the body in the ACK MAY be any type listed in the Accept header field in the 415.

For example, consider the following request:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 INVITE
```

The ACK request for a non-2xx final response to this request would look like this:

```
ACK sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKkjshdyff
To: Bob <sip:bob@biloxi.com>;tag=99sa0xk
From: Alice <sip:alice@atlanta.com>;tag=88sja8x
Max-Forwards: 70
Call-ID: 987asjd97y7atg
CSeq: 986759 ACK
```

17.1.2 Non-INVITE Client Transaction

17.1.2.1 Overview of the non-INVITE Transaction

Non-INVITE transactions do not make use of ACK. They are simple request-response interactions. For unreliable transports, requests are retransmitted at an interval which starts at T1 and doubles until it hits T2. If a provisional response is received, retransmissions continue for unreliable transports, but at an interval of T2. The server transaction retransmits the last response it sent, which can be a provisional or final response, only when a retransmission of the request is received. This is why request retransmissions need to continue even after a provisional response; they are to ensure reliable delivery of the final response.

Unlike an INVITE transaction, a non-INVITE transaction has no special handling for the 2xx response. The result is that only a single 2xx response to a non-INVITE is ever delivered to a UAC.

17.1.2.2 Formal Description

The state machine for the non-INVITE client transaction is shown in Figure 6. It is very similar to the state machine for INVITE.

The "Trying" state is entered when the TU initiates a new client transaction with a request. When entering this state, the client transaction SHOULD set timer F to fire in 64*T1 seconds. The request MUST be passed to the transport layer for transmission. If an unreliable transport is in use, the client transaction MUST set timer E to fire in T1 seconds. If timer E fires while still in this state, the timer is reset, but this time with a value of MIN(2*T1, T2). When the timer fires again, it is reset to a MIN(4*T1, T2). This process continues so that retransmissions occur with an exponentially increasing interval that caps at T2. The default value of T2 is 4s, and it represents the amount of time a non-INVITE server transaction will take to respond to a request, if it does not respond immediately. For the default values of T1 and T2, this results in intervals of 500 ms, 1 s, 2 s, 4 s, 4 s, 4 s, etc.

If Timer F fires while the client transaction is still in the "Trying" state, the client transaction SHOULD inform the TU about the timeout, and then it SHOULD enter the "Terminated" state. If a provisional response is received while in the "Trying" state, the response MUST be passed to the TU, and then the client transaction SHOULD move to the "Proceeding" state. If a final response (status codes 200-699) is received while in the "Trying" state, the response MUST be passed to the TU, and the client transaction MUST transition to the "Completed" state.

If Timer E fires while in the "Proceeding" state, the request MUST be passed to the transport layer for retransmission, and Timer E MUST be reset with a value of T2 seconds. If timer F fires while in the "Proceeding" state, the TU MUST be informed of a timeout, and the client transaction MUST transition to the terminated state. If a final response (status codes 200-699) is received while in the "Proceeding" state, the response MUST be passed to the TU, and the client transaction MUST transition to the "Completed" state.

Once the client transaction enters the "Completed" state, it MUST set Timer K to fire in T4 seconds for unreliable transports, and zero seconds for reliable transports. The "Completed" state exists to buffer any additional response retransmissions that may be received (which is why the client transaction remains there only for

unreliable transports). T4 represents the amount of time the network will take to clear messages between client and server transactions. The default value of T4 is 5s. A response is a retransmission when it matches the same transaction, using the rules specified in Section 17.1.3. If Timer K fires while in this state, the client transaction MUST transition to the "Terminated" state.

Once the transaction is in the terminated state, it MUST be destroyed immediately.

17.1.3 Matching Responses to Client Transactions

When the transport layer in the client receives a response, it has to determine which client transaction will handle the response, so that the processing of Sections 17.1.1 and 17.1.2 can take place. The branch parameter in the top Via header field is used for this purpose. A response matches a client transaction under two conditions:

1. If the response has the same value of the branch parameter in the top Via header field as the branch parameter in the top Via header field of the request that created the transaction.

2. If the method parameter in the CSeq header field matches the method of the request that created the transaction. The method is needed since a CANCEL request constitutes a different transaction, but shares the same value of the branch parameter.

If a request is sent via multicast, it is possible that it will generate multiple responses from different servers. These responses will all have the same branch parameter in the topmost Via, but vary in the To tag. The first response received, based on the rules above, will be used, and others will be viewed as retransmissions. That is not an error; multicast SIP provides only a rudimentary "single-hop-discovery-like" service that is limited to processing a single response. See Section 18.1.1 for details.

17.1.4 Handling Transport Errors

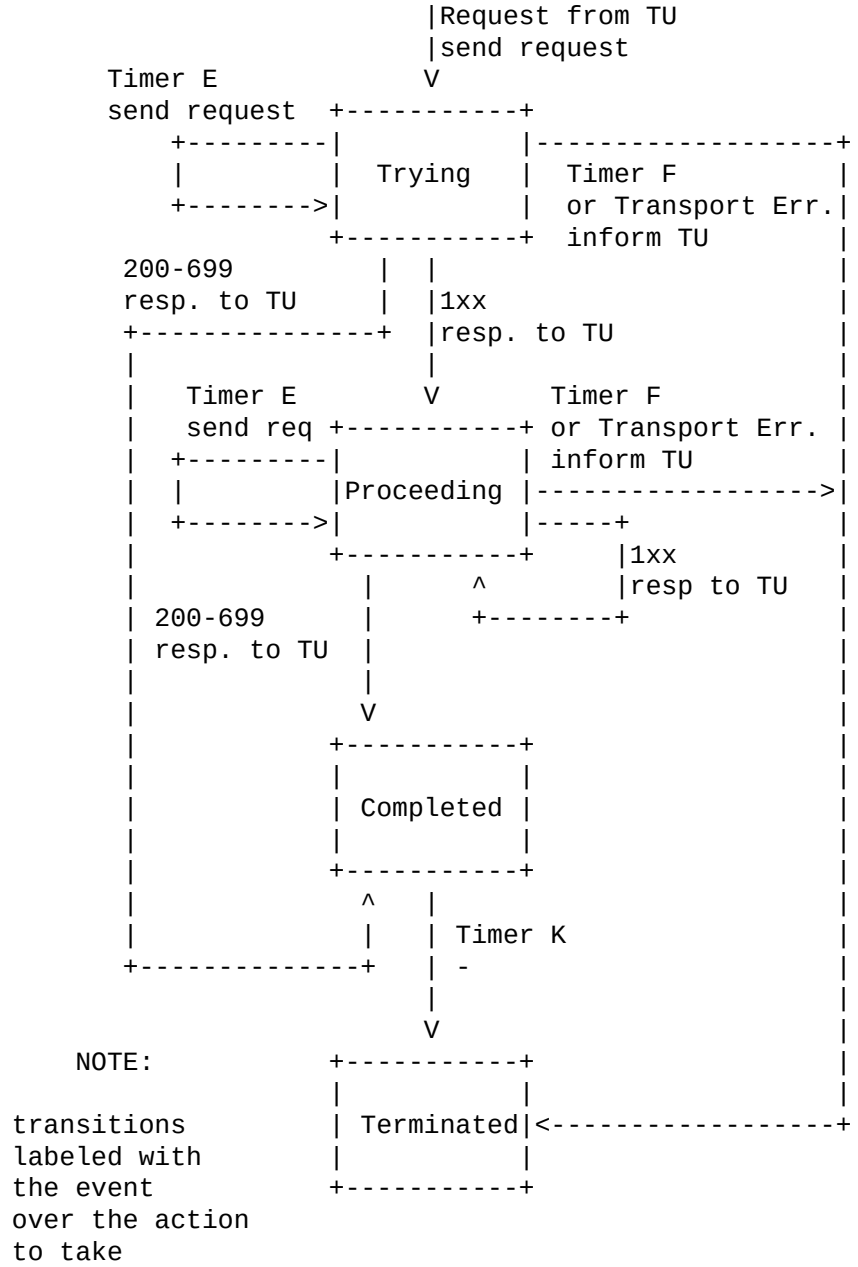


Figure 6: non-INVITE client transaction

When the client transaction sends a request to the transport layer to be sent, the following procedures are followed if the transport layer indicates a failure.

The client transaction SHOULD inform the TU that a transport failure has occurred, and the client transaction SHOULD transition directly to the "Terminated" state. The TU will handle the failover mechanisms described in [4].

17.2 Server Transaction

The server transaction is responsible for the delivery of requests to the TU and the reliable transmission of responses. It accomplishes this through a state machine. Server transactions are created by the core when a request is received, and transaction handling is desired for that request (this is not always the case).

As with the client transactions, the state machine depends on whether the received request is an INVITE request.

17.2.1 INVITE Server Transaction

The state diagram for the INVITE server transaction is shown in Figure 7.

When a server transaction is constructed for a request, it enters the "Proceeding" state. The server transaction MUST generate a 100 (Trying) response unless it knows that the TU will generate a provisional or final response within 200 ms, in which case it MAY generate a 100 (Trying) response. This provisional response is needed to quench request retransmissions rapidly in order to avoid network congestion. The 100 (Trying) response is constructed according to the procedures in Section 8.2.6, except that the insertion of tags in the To header field of the response (when none was present in the request) is downgraded from MAY to SHOULD NOT. The request MUST be passed to the TU.

The TU passes any number of provisional responses to the server transaction. So long as the server transaction is in the "Proceeding" state, each of these MUST be passed to the transport layer for transmission. They are not sent reliably by the transaction layer (they are not retransmitted by it) and do not cause a change in the state of the server transaction. If a request retransmission is received while in the "Proceeding" state, the most recent provisional response that was received from the TU MUST be passed to the transport layer for retransmission. A request is a retransmission if it matches the same server transaction based on the rules of Section 17.2.3.

If, while in the "Proceeding" state, the TU passes a 2xx response to the server transaction, the server transaction MUST pass this response to the transport layer for transmission. It is not

retransmitted by the server transaction; retransmissions of 2xx responses are handled by the TU. The server transaction MUST then transition to the "Terminated" state.

While in the "Proceeding" state, if the TU passes a response with status code from 300 to 699 to the server transaction, the response MUST be passed to the transport layer for transmission, and the state machine MUST enter the "Completed" state. For unreliable transports, timer G is set to fire in T1 seconds, and is not set to fire for reliable transports.

> This is a change from RFC 2543, where responses were always retransmitted, even over reliable transports.

When the "Completed" state is entered, timer H MUST be set to fire in 64*T1 seconds for all transports. Timer H determines when the server transaction abandons retransmitting the response. Its value is chosen to equal Timer B, the amount of time a client transaction will continue to retry sending a request. If timer G fires, the response is passed to the transport layer once more for retransmission, and timer G is set to fire in MIN(2*T1, T2) seconds. From then on, when timer G fires, the response is passed to the transport again for transmission, and timer G is reset with a value that doubles, unless that value exceeds T2, in which case it is reset with the value of T2. This is identical to the retransmit behavior for requests in the "Trying" state of the non-INVITE client transaction. Furthermore, while in the "Completed" state, if a request retransmission is received, the server SHOULD pass the response to the transport for retransmission.

If an ACK is received while the server transaction is in the "Completed" state, the server transaction MUST transition to the "Confirmed" state. As Timer G is ignored in this state, any retransmissions of the response will cease.

If timer H fires while in the "Completed" state, it implies that the ACK was never received. In this case, the server transaction MUST transition to the "Terminated" state, and MUST indicate to the TU that a transaction failure has occurred.

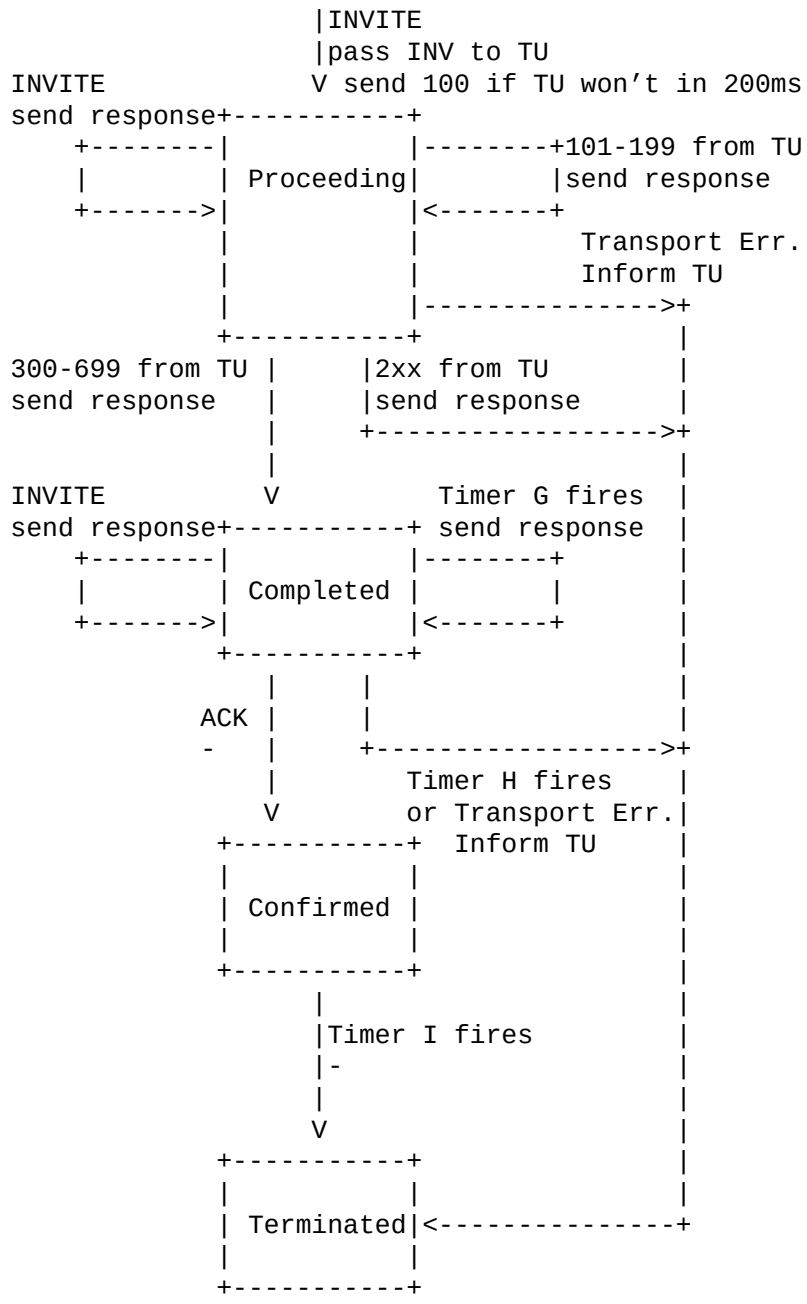


Figure 7: INVITE server transaction

The purpose of the "Confirmed" state is to absorb any additional ACK messages that arrive, triggered from retransmissions of the final response. When this state is entered, timer I is set to fire in T4 seconds for unreliable transports, and zero seconds for reliable transports. Once timer I fires, the server MUST transition to the "Terminated" state.

Once the transaction is in the "Terminated" state, it MUST be destroyed immediately. As with client transactions, this is needed to ensure reliability of the 2xx responses to INVITE.

17.2.2 Non-INVITE Server Transaction

The state machine for the non-INVITE server transaction is shown in Figure 8.

The state machine is initialized in the "Trying" state and is passed a request other than INVITE or ACK when initialized. This request is passed up to the TU. Once in the "Trying" state, any further request retransmissions are discarded. A request is a retransmission if it matches the same server transaction, using the rules specified in Section 17.2.3.

While in the "Trying" state, if the TU passes a provisional response to the server transaction, the server transaction MUST enter the "Proceeding" state. The response MUST be passed to the transport layer for transmission. Any further provisional responses that are received from the TU while in the "Proceeding" state MUST be passed to the transport layer for transmission. If a retransmission of the request is received while in the "Proceeding" state, the most recently sent provisional response MUST be passed to the transport layer for retransmission. If the TU passes a final response (status codes 200-699) to the server while in the "Proceeding" state, the transaction MUST enter the "Completed" state, and the response MUST be passed to the transport layer for transmission.

When the server transaction enters the "Completed" state, it MUST set Timer J to fire in 64*T1 seconds for unreliable transports, and zero seconds for reliable transports. While in the "Completed" state, the server transaction MUST pass the final response to the transport layer for retransmission whenever a retransmission of the request is received. Any other final responses passed by the TU to the server transaction MUST be discarded while in the "Completed" state. The server transaction remains in this state until Timer J fires, at which point it MUST transition to the "Terminated" state.

The server transaction MUST be destroyed the instant it enters the "Terminated" state.

17.2.3 Matching Requests to Server Transactions

When a request is received from the network by the server, it has to be matched to an existing transaction. This is accomplished in the following manner.

The branch parameter in the topmost Via header field of the request is examined. If it is present and begins with the magic cookie "z9hG4bK", the request was generated by a client transaction compliant to this specification. Therefore, the branch parameter will be unique across all transactions sent by that client. The request matches a transaction if:

1. the branch parameter in the request is equal to the one in the top Via header field of the request that created the transaction, and

2. the sent-by value in the top Via of the request is equal to the one in the request that created the transaction, and

3. the method of the request matches the one that created the transaction, except for ACK, where the method of the request that created the transaction is INVITE.

This matching rule applies to both INVITE and non-INVITE transactions alike.

The sent-by value is used as part of the matching process because there could be accidental or malicious duplication of branch parameters from different clients.

If the branch parameter in the top Via header field is not present, or does not contain the magic cookie, the following procedures are used. These exist to handle backwards compatibility with RFC 2543 compliant implementations.

The INVITE request matches a transaction if the Request-URI, To tag, From tag, Call-ID, CSeq, and top Via header field match those of the INVITE request which created the transaction. In this case, the INVITE is a retransmission of the original one that created the transaction. The ACK request matches a transaction if the Request-URI, From tag, Call-ID, CSeq number (not the method), and top Via header field match those of the INVITE request which created the transaction, and the To tag of the ACK matches the To tag of the response sent by the server transaction. Matching is done based on the matching rules defined for each of those header fields. Inclusion of the tag in the To header field in the ACK matching process helps disambiguate ACK for 2xx from ACK for other responses

at a proxy, which may have forwarded both responses (This can occur in unusual conditions. Specifically, when a proxy forked a request, and then crashes, the responses may be delivered to another proxy, which might end up forwarding multiple responses upstream). An ACK request that matches an INVITE transaction matched by a previous ACK is considered a retransmission of that previous ACK.

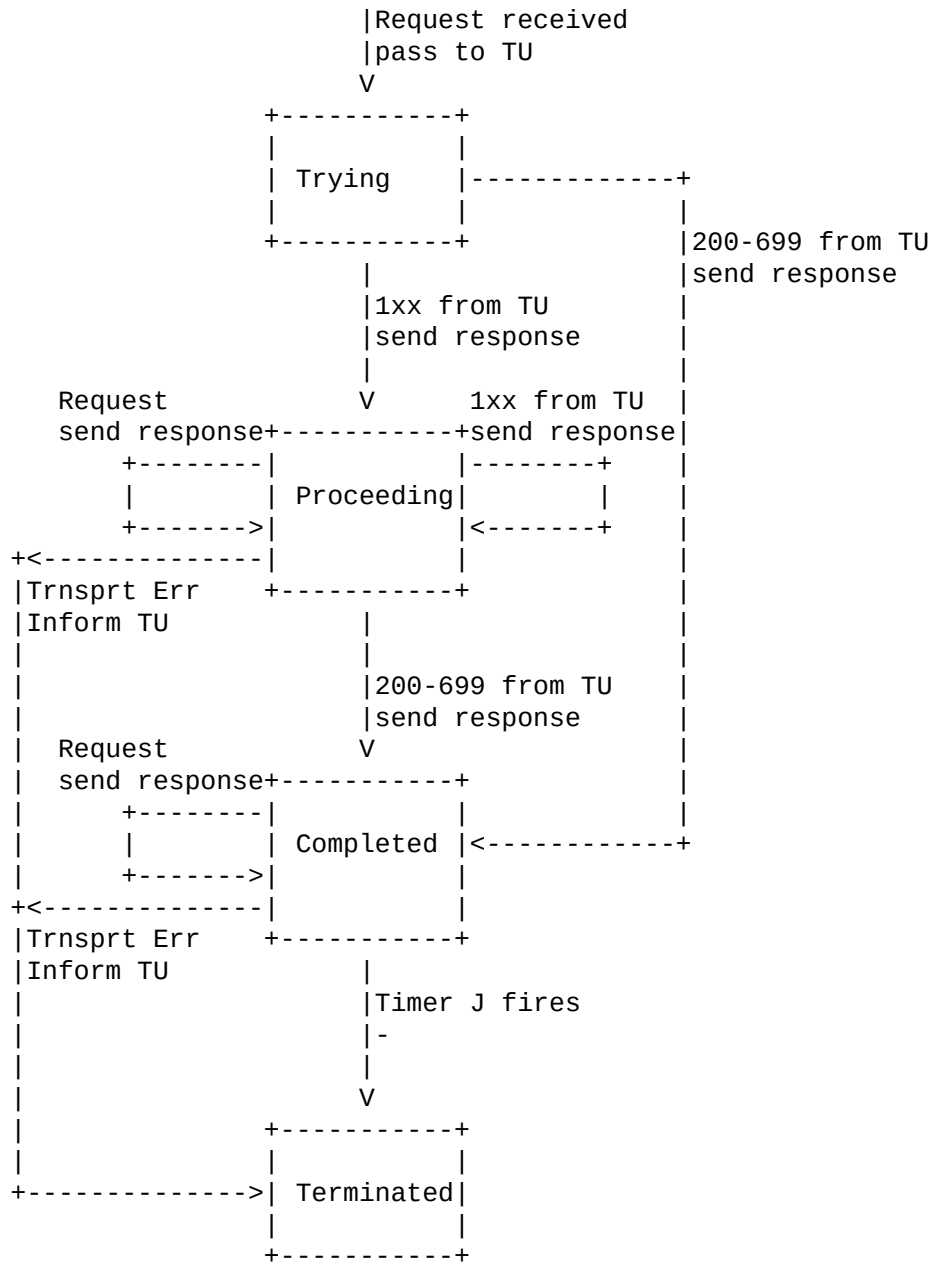


Figure 8: non-INVITE server transaction

For all other request methods, a request is matched to a transaction if the Request-URI, To tag, From tag, Call-ID, CSeq (including the method), and top Via header field match those of the request that created the transaction. Matching is done based on the matching

rules defined for each of those header fields. When a non-INVITE request matches an existing transaction, it is a retransmission of the request that created that transaction.

Because the matching rules include the Request-URI, the server cannot match a response to a transaction. When the TU passes a response to the server transaction, it must pass it to the specific server transaction for which the response is targeted.

17.2.4 Handling Transport Errors

When the server transaction sends a response to the transport layer to be sent, the following procedures are followed if the transport layer indicates a failure.

First, the procedures in [4] are followed, which attempt to deliver the response to a backup. If those should all fail, based on the definition of failure in [4], the server transaction SHOULD inform the TU that a failure has occurred, and SHOULD transition to the terminated state.

18 Transport

The transport layer is responsible for the actual transmission of requests and responses over network transports. This includes determination of the connection to use for a request or response in the case of connection-oriented transports.

The transport layer is responsible for managing persistent connections for transport protocols like TCP and SCTP, or TLS over those, including ones opened to the transport layer. This includes connections opened by the client or server transports, so that connections are shared between client and server transport functions. These connections are indexed by the tuple formed from the address, port, and transport protocol at the far end of the connection. When a connection is opened by the transport layer, this index is set to the destination IP, port and transport. When the connection is accepted by the transport layer, this index is set to the source IP address, port number, and transport. Note that, because the source port is often ephemeral, but it cannot be known whether it is ephemeral or selected through procedures in [4], connections accepted by the transport layer will frequently not be reused. The result is that two proxies in a "peering" relationship using a connection-oriented transport frequently will have two connections in use, one for transactions initiated in each direction.

It is RECOMMENDED that connections be kept open for some implementation-defined duration after the last message was sent or received over that connection. This duration SHOULD at least equal the longest amount of time the element would need in order to bring a transaction from instantiation to the terminated state. This is to make it likely that transactions are completed over the same connection on which they are initiated (for example, request, response, and in the case of INVITE, ACK for non-2xx responses). This usually means at least 64*T1 (see Section 17.1.1.1 for a definition of T1). However, it could be larger in an element that has a TU using a large value for timer C (bullet 11 of Section 16.6), for example.

All SIP elements MUST implement UDP and TCP. SIP elements MAY implement other protocols.

Making TCP mandatory for the UA is a substantial change from RFC 2543. It has arisen out of the need to handle larger messages, which MUST use TCP, as discussed below. Thus, even if an element never sends large messages, it may receive one and needs to be able to handle them.

18.1 Clients

18.1.1 Sending Requests

The client side of the transport layer is responsible for sending the request and receiving responses. The user of the transport layer passes the client transport the request, an IP address, port, transport, and possibly TTL for multicast destinations.

If a request is within 200 bytes of the path MTU, or if it is larger than 1300 bytes and the path MTU is unknown, the request MUST be sent using an RFC 2914 [43] congestion controlled transport protocol, such as TCP. If this causes a change in the transport protocol from the one indicated in the top Via, the value in the top Via MUST be changed. This prevents fragmentation of messages over UDP and provides congestion control for larger messages. However, implementations MUST be able to handle messages up to the maximum datagram packet size. For UDP, this size is 65,535 bytes, including IP and UDP headers.

The 200 byte "buffer" between the message size and the MTU accommodates the fact that the response in SIP can be larger than the request. This happens due to the addition of Record-Route header field values to the responses to INVITE, for example. With the extra buffer, the response can be about 170 bytes larger than the request, and still not be fragmented on IPv4 (about 30 bytes

is consumed by IP/UDP, assuming no IPSec). 1300 is chosen when path MTU is not known, based on the assumption of a 1500 byte Ethernet MTU.

If an element sends a request over TCP because of these message size constraints, and that request would have otherwise been sent over UDP, if the attempt to establish the connection generates either an ICMP Protocol Not Supported, or results in a TCP reset, the element SHOULD retry the request, using UDP. This is only to provide backwards compatibility with RFC 2543 compliant implementations that do not support TCP. It is anticipated that this behavior will be deprecated in a future revision of this specification.

A client that sends a request to a multicast address MUST add the "maddr" parameter to its Via header field value containing the destination multicast address, and for IPv4, SHOULD add the "ttl" parameter with a value of 1. Usage of IPv6 multicast is not defined in this specification, and will be a subject of future standardization when the need arises.

These rules result in a purposeful limitation of multicast in SIP. Its primary function is to provide a "single-hop-discovery-like" service, delivering a request to a group of homogeneous servers, where it is only required to process the response from any one of them. This functionality is most useful for registrations. In fact, based on the transaction processing rules in Section 17.1.3, the client transaction will accept the first response, and view any others as retransmissions because they all contain the same Via branch identifier.

Before a request is sent, the client transport MUST insert a value of the "sent-by" field into the Via header field. This field contains an IP address or host name, and port. The usage of an FQDN is RECOMMENDED. This field is used for sending responses under certain conditions, described below. If the port is absent, the default value depends on the transport. It is 5060 for UDP, TCP and SCTP, 5061 for TLS.

For reliable transports, the response is normally sent on the connection on which the request was received. Therefore, the client transport MUST be prepared to receive the response on the same connection used to send the request. Under error conditions, the server may attempt to open a new connection to send the response. To handle this case, the transport layer MUST also be prepared to receive an incoming connection on the source IP address from which the request was sent and port number in the "sent-by" field. It also

MUST be prepared to receive incoming connections on any address and port that would be selected by a server based on the procedures described in Section 5 of [4].

For unreliable unicast transports, the client transport MUST be prepared to receive responses on the source IP address from which the request is sent (as responses are sent back to the source address) and the port number in the "sent-by" field. Furthermore, as with reliable transports, in certain cases the response will be sent elsewhere. The client MUST be prepared to receive responses on any address and port that would be selected by a server based on the procedures described in Section 5 of [4].

For multicast, the client transport MUST be prepared to receive responses on the same multicast group and port to which the request is sent (that is, it needs to be a member of the multicast group it sent the request to.)

If a request is destined to an IP address, port, and transport to which an existing connection is open, it is RECOMMENDED that this connection be used to send the request, but another connection MAY be opened and used.

If a request is sent using multicast, it is sent to the group address, port, and TTL provided by the transport user. If a request is sent using unicast unreliable transports, it is sent to the IP address and port provided by the transport user.

18.1.2 Receiving Responses

When a response is received, the client transport examines the top Via header field value. If the value of the "sent-by" parameter in that header field value does not correspond to a value that the client transport is configured to insert into requests, the response MUST be silently discarded.

If there are any client transactions in existence, the client transport uses the matching procedures of Section 17.1.3 to attempt to match the response to an existing transaction. If there is a match, the response MUST be passed to that transaction. Otherwise, the response MUST be passed to the core (whether it be stateless proxy, stateful proxy, or UA) for further processing. Handling of these "stray" responses is dependent on the core (a proxy will forward them, while a UA will discard, for example).

18.2 Servers

18.2.1 Receiving Requests

A server SHOULD be prepared to receive requests on any IP address, port and transport combination that can be the result of a DNS lookup on a SIP or SIPS URI [4] that is handed out for the purposes of communicating with that server. In this context, "handing out" includes placing a URI in a Contact header field in a REGISTER request or a redirect response, or in a Record-Route header field in a request or response. A URI can also be "handed out" by placing it on a web page or business card. It is also RECOMMENDED that a server listen for requests on the default SIP ports (5060 for TCP and UDP, 5061 for TLS over TCP) on all public interfaces. The typical exception would be private networks, or when multiple server instances are running on the same host. For any port and interface that a server listens on for UDP, it MUST listen on that same port and interface for TCP. This is because a message may need to be sent using TCP, rather than UDP, if it is too large. As a result, the converse is not true. A server need not listen for UDP on a particular address and port just because it is listening on that same address and port for TCP. There may, of course, be other reasons why a server needs to listen for UDP on a particular address and port.

When the server transport receives a request over any transport, it MUST examine the value of the "sent-by" parameter in the top Via header field value. If the host portion of the "sent-by" parameter contains a domain name, or if it contains an IP address that differs from the packet source address, the server MUST add a "received" parameter to that Via header field value. This parameter MUST contain the source address from which the packet was received. This is to assist the server transport layer in sending the response, since it must be sent to the source IP address from which the request came.

Consider a request received by the server transport which looks like, in part:

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060
```

The request is received with a source IP address of 192.0.2.4. Before passing the request up, the transport adds a "received" parameter, so that the request would look like, in part:

```
INVITE sip:bob@Biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;received=192.0.2.4
```

Next, the server transport attempts to match the request to a server transaction. It does so using the matching rules described in Section 17.2.3. If a matching server transaction is found, the request is passed to that transaction for processing. If no match is found, the request is passed to the core, which may decide to construct a new server transaction for that request. Note that when a UAS core sends a 2xx response to INVITE, the server transaction is destroyed. This means that when the ACK arrives, there will be no matching server transaction, and based on this rule, the ACK is passed to the UAS core, where it is processed.

18.2.2 Sending Responses

The server transport uses the value of the top Via header field in order to determine where to send a response. It MUST follow the following process:

- If the "sent-protocol" is a reliable transport protocol such as TCP or SCTP, or TLS over those, the response MUST be sent using the existing connection to the source of the original request that created the transaction, if that connection is still open. This requires the server transport to maintain an association between server transactions and transport connections. If that connection is no longer open, the server SHOULD open a connection to the IP address in the "received" parameter, if present, using the port in the "sent-by" value, or the default port for that transport, if no port is specified. If that connection attempt fails, the server SHOULD use the procedures in [4] for servers in order to determine the IP address and port to open the connection and send the response to.

- Otherwise, if the Via header field value contains a "maddr" parameter, the response MUST be forwarded to the address listed there, using the port indicated in "sent-by", or port 5060 if none is present. If the address is a multicast address, the response SHOULD be sent using the TTL indicated in the "ttl" parameter, or with a TTL of 1 if that parameter is not present.

- Otherwise (for unreliable unicast transports), if the top Via has a "received" parameter, the response MUST be sent to the address in the "received" parameter, using the port indicated in the "sent-by" value, or using port 5060 if none is specified explicitly. If this fails, for example, elicits an ICMP "port unreachable" response, the procedures of Section 5 of [4] SHOULD be used to determine where to send the response.

- Otherwise, if it is not receiver-tagged, the response MUST be sent to the address indicated by the "sent-by" value, using the procedures in Section 5 of [4].

18.3 Framing

In the case of message-oriented transports (such as UDP), if the message has a Content-Length header field, the message body is assumed to contain that many bytes. If there are additional bytes in the transport packet beyond the end of the body, they MUST be discarded. If the transport packet ends before the end of the message body, this is considered an error. If the message is a response, it MUST be discarded. If the message is a request, the element SHOULD generate a 400 (Bad Request) response. If the message has no Content-Length header field, the message body is assumed to end at the end of the transport packet.

In the case of stream-oriented transports such as TCP, the Content-Length header field indicates the size of the body. The Content-Length header field MUST be used with stream oriented transports.

18.4 Error Handling

Error handling is independent of whether the message was a request or response.

If the transport user asks for a message to be sent over an unreliable transport, and the result is an ICMP error, the behavior depends on the type of ICMP error. Host, network, port or protocol unreachable errors, or parameter problem errors SHOULD cause the transport layer to inform the transport user of a failure in sending. Source quench and TTL exceeded ICMP errors SHOULD be ignored.

If the transport user asks for a request to be sent over a reliable transport, and the result is a connection failure, the transport layer SHOULD inform the transport user of a failure in sending.

19 Common Message Components

There are certain components of SIP messages that appear in various places within SIP messages (and sometimes, outside of them) that merit separate discussion.

19.1 SIP and SIPS Uniform Resource Indicators

A SIP or SIPS URI identifies a communications resource. Like all URIs, SIP and SIPS URIs may be placed in web pages, email messages, or printed literature. They contain sufficient information to initiate and maintain a communication session with the resource.

Examples of communications resources include the following:

- a user of an online service
- an appearance on a multi-line phone
- a mailbox on a messaging system
- a PSTN number at a gateway service
- a group (such as "sales" or "helpdesk") in an organization

A SIPS URI specifies that the resource be contacted securely. This means, in particular, that TLS is to be used between the UAC and the domain that owns the URI. From there, secure communications are used to reach the user, where the specific security mechanism depends on the policy of the domain. Any resource described by a SIP URI can be "upgraded" to a SIPS URI by just changing the scheme, if it is desired to communicate with that resource securely.

19.1.1 SIP and SIPS URI Components

The "sip:" and "sips:" schemes follow the guidelines in RFC 2396 [5]. They use a form similar to the mailto URL, allowing the specification of SIP request-header fields and the SIP message-body. This makes it possible to specify the subject, media type, or urgency of sessions initiated by using a URI on a web page or in an email message. The formal syntax for a SIP or SIPS URI is presented in Section 25. Its general form, in the case of a SIP URI, is:

```
sip:user:password@host:port;uri-parameters?headers
```

The format for a SIPS URI is the same, except that the scheme is "sips" instead of sip. These tokens, and some of the tokens in their expansions, have the following meanings:

user: The identifier of a particular resource at the host being addressed. The term "host" in this context frequently refers to a domain. The "userinfo" of a URI consists of this user field, the password field, and the @ sign following them. The userinfo part of a URI is optional and MAY be absent when the

destination host does not have a notion of users or when the host itself is the resource being identified. If the @ sign is present in a SIP or SIPS URI, the user field MUST NOT be empty.

If the host being addressed can process telephone numbers, for instance, an Internet telephony gateway, a telephone-subscriber field defined in RFC 2806 [9] MAY be used to populate the user field. There are special escaping rules for encoding telephone-subscriber fields in SIP and SIPS URIs described in Section 19.1.2.

password: A password associated with the user. While the SIP and SIPS URI syntax allows this field to be present, its use is NOT RECOMMENDED, because the passing of authentication information in clear text (such as URIs) has proven to be a security risk in almost every case where it has been used. For instance, transporting a PIN number in this field exposes the PIN.

Note that the password field is just an extension of the user portion. Implementations not wishing to give special significance to the password portion of the field MAY simply treat "user:password" as a single string.

host: The host providing the SIP resource. The host part contains either a fully-qualified domain name or numeric IPv4 or IPv6 address. Using the fully-qualified domain name form is RECOMMENDED whenever possible.

port: The port number where the request is to be sent.

URI parameters: Parameters affecting a request constructed from the URI.

URI parameters are added after the hostport component and are separated by semi-colons.

URI parameters take the form:

parameter-name "=" parameter-value

Even though an arbitrary number of URI parameters may be included in a URI, any given parameter-name MUST NOT appear more than once.

This extensible mechanism includes the transport, maddr, ttl, user, method and lr parameters.

The transport parameter determines the transport mechanism to be used for sending SIP messages, as specified in [4]. SIP can use any network transport protocol. Parameter names are defined for UDP (RFC 768 [14]), TCP (RFC 761 [15]), and SCTP (RFC 2960 [16]). For a SIPS URI, the transport parameter MUST indicate a reliable transport.

The maddr parameter indicates the server address to be contacted for this user, overriding any address derived from the host field. When an maddr parameter is present, the port and transport components of the URI apply to the address indicated in the maddr parameter value. [4] describes the proper interpretation of the transport, maddr, and hostport in order to obtain the destination address, port, and transport for sending a request.

The maddr field has been used as a simple form of loose source routing. It allows a URI to specify a proxy that must be traversed en-route to the destination. Continuing to use the maddr parameter this way is strongly discouraged (the mechanisms that enable it are deprecated). Implementations should instead use the Route mechanism described in this document, establishing a pre-existing route set if necessary (see Section 8.1.1.1). This provides a full URI to describe the node to be traversed.

The ttl parameter determines the time-to-live value of the UDP multicast packet and MUST only be used if maddr is a multicast address and the transport protocol is UDP. For example, to specify a call to alice@atlanta.com using multicast to 239.255.255.1 with a ttl of 15, the following URI would be used:

```
sip:alice@atlanta.com;maddr=239.255.255.1;ttl=15
```

The set of valid telephone-subscriber strings is a subset of valid user strings. The user URI parameter exists to distinguish telephone numbers from user names that happen to look like telephone numbers. If the user string contains a telephone number formatted as a telephone-subscriber, the user parameter value "phone" SHOULD be present. Even without this parameter, recipients of SIP and SIPS URIs MAY interpret the pre-@ part as a telephone number if local restrictions on the name space for user name allow it.

The method of the SIP request constructed from the URI can be specified with the method parameter.

The lr parameter, when present, indicates that the element responsible for this resource implements the routing mechanisms specified in this document. This parameter will be used in the URIs proxies place into Record-Route header field values, and may appear in the URIs in a pre-existing route set.

This parameter is used to achieve backwards compatibility with systems implementing the strict-routing mechanisms of RFC 2543 and the rfc2543bis drafts up to bis-05. An element preparing to send a request based on a URI not containing this parameter can assume the receiving element implements strict-routing and reformat the message to preserve the information in the Request-URI.

Since the uri-parameter mechanism is extensible, SIP elements MUST silently ignore any uri-parameters that they do not understand.

Headers: Header fields to be included in a request constructed from the URI.

Headers fields in the SIP request can be specified with the "?" mechanism within a URI. The header names and values are encoded in ampersand separated hname = hvalue pairs. The special hname "body" indicates that the associated hvalue is the message-body of the SIP request.

Table 1 summarizes the use of SIP and SIPS URI components based on the context in which the URI appears. The external column describes URIs appearing anywhere outside of a SIP message, for instance on a web page or business card. Entries marked "m" are mandatory, those marked "o" are optional, and those marked "-" are not allowed. Elements processing URIs SHOULD ignore any disallowed components if they are present. The second column indicates the default value of an optional element if it is not present. "--" indicates that the element is either not optional, or has no default value.

URIs in Contact header fields have different restrictions depending on the context in which the header field appears. One set applies to messages that establish and maintain dialogs (INVITE and its 200 (OK) response). The other applies to registration and redirection messages (REGISTER, its 200 (OK) response, and 3xx class responses to any method).

19.1.2 Character Escaping Requirements

	default	Req.-URI	To	From	reg./redir. Contact	dialog Contact/ R-R/Route	external
user	--	o	o	o	o	o	o
password	--	o	o	o	o	o	o
host	--	m	m	m	m	m	m
port	(1)	o	-	-	o	o	o
user-param	ip	o	o	o	o	o	o
method	INVITE	-	-	-	-	-	o
maddr-param	--	o	-	-	o	o	o
ttl-param	1	o	-	-	o	-	o
transp.-param	(2)	o	-	-	o	o	o
lr-param	--	o	-	-	-	o	o
other-param	--	o	o	o	o	o	o
headers	--	-	-	-	o	-	o

(1): The default port value is transport and scheme dependent. The default is 5060 for sip: using UDP, TCP, or SCTP. The default is 5061 for sip: using TLS over TCP and sips: over TCP.

(2): The default transport is scheme dependent. For sip:, it is UDP. For sips:, it is TCP.

Table 1: Use and default values of URI components for SIP header field values, Request-URI and references

SIP follows the requirements and guidelines of RFC 2396 [5] when defining the set of characters that must be escaped in a SIP URI, and uses its ""%" HEX HEX" mechanism for escaping. From RFC 2396 [5]:

> The set of characters actually reserved within any given URI component is defined by that component. In general, a character is reserved if the semantics of the URI changes if the character is replaced with its escaped US-ASCII encoding [5]. Excluded US-ASCII characters (RFC 2396 [5]), such as space and control characters and characters used as URI delimiters, also MUST be escaped. URIs MUST NOT contain unescaped space and control characters.

For each component, the set of valid BNF expansions defines exactly which characters may appear unescaped. All other characters MUST be escaped.

For example, "@" is not in the set of characters in the user component, so the user "j@s0n" must have at least the @ sign encoded, as in "j%40s0n".

Expanding the hname and hvalue tokens in Section 25 show that all URI reserved characters in header field names and values MUST be escaped.

The telephone-subscriber subset of the user component has special escaping considerations. The set of characters not reserved in the RFC 2806 [9] description of telephone-subscriber contains a number of characters in various syntax elements that need to be escaped when used in SIP URIs. Any characters occurring in a telephone-subscriber that do not appear in an expansion of the BNF for the user rule MUST be escaped.

Note that character escaping is not allowed in the host component of a SIP or SIPS URI (the % character is not valid in its expansion). This is likely to change in the future as requirements for Internationalized Domain Names are finalized. Current implementations MUST NOT attempt to improve robustness by treating received escaped characters in the host component as literally equivalent to their unescaped counterpart. The behavior required to meet the requirements of IDN may be significantly different.

19.1.3 Example SIP and SIPS URIs

```
sip:alice@atlanta.com
sip:alice:secretword@atlanta.com;transport=tcp
sips:alice@atlanta.com?subject=project%20x&priority=urgent
sip:+1-212-555-1212:1234@gateway.com;user=phone
sips:1212@gateway.com
sip:alice@192.0.2.4
sip:atlanta.com;method=REGISTER?to=alice%40atlanta.com
sip:alice;day=tuesday@atlanta.com
```

The last sample URI above has a user field value of "alice;day=tuesday". The escaping rules defined above allow a semicolon to appear unescaped in this field. For the purposes of this protocol, the field is opaque. The structure of that value is only useful to the SIP element responsible for the resource.

19.1.4 URI Comparison

Some operations in this specification require determining whether two SIP or SIPS URIs are equivalent. In this specification, registrars need to compare bindings in Contact URIs in REGISTER requests (see Section 10.3.). SIP and SIPS URIs are compared for equality according to the following rules:

- A SIP and SIPS URI are never equivalent.

- Comparison of the userinfo of SIP and SIPS URIs is case-sensitive. This includes userinfo containing passwords or formatted as telephone-subscribers. Comparison of all other components of the URI is case-insensitive unless explicitly defined otherwise.

- The ordering of parameters and header fields is not significant in comparing SIP and SIPS URIs.

- Characters other than those in the "reserved" set (see RFC 2396 [5]) are equivalent to their ""%" HEX HEX" encoding.

- An IP address that is the result of a DNS lookup of a host name does not match that host name.

- For two URIs to be equal, the user, password, host, and port components must match.

  A URI omitting the user component will not match a URI that includes one. A URI omitting the password component will not match a URI that includes one.

  A URI omitting any component with a default value will not match a URI explicitly containing that component with its default value. For instance, a URI omitting the optional port component will not match a URI explicitly declaring port 5060. The same is true for the transport-parameter, ttl-parameter, user-parameter, and method components.

  Defining sip:user@host to not be equivalent to sip:user@host:5060 is a change from RFC 2543. When deriving addresses from URIs, equivalent addresses are expected from equivalent URIs. The URI sip:user@host:5060 will always resolve to port 5060. The URI sip:user@host may resolve to other ports through the DNS SRV mechanisms detailed in [4].

- URI uri-parameter components are compared as follows:

  - Any uri-parameter appearing in both URIs must match.

  - A user, ttl, or method uri-parameter appearing in only one URI never matches, even if it contains the default value.

  - A URI that includes an maddr parameter will not match a URI that contains no maddr parameter.

  - All other uri-parameters appearing in only one URI are ignored when comparing the URIs.

- URI header components are never ignored. Any present header component MUST be present in both URIs and match for the URIs to match. The matching rules are defined for each header field in Section 20.

The URIs within each of the following sets are equivalent:

```
sip:%61lice@atlanta.com;transport=TCP
sip:alice@AtLanTa.CoM;Transport=tcp

sip:carol@chicago.com
sip:carol@chicago.com;newparam=5
sip:carol@chicago.com;security=on

sip:biloxi.com;transport=tcp;method=REGISTER?to=sip:bob%40biloxi.com
sip:biloxi.com;method=REGISTER;transport=tcp?to=sip:bob%40biloxi.com

sip:alice@atlanta.com?subject=project%20x&priority=urgent
sip:alice@atlanta.com?priority=urgent&subject=project%20x
```

The URIs within each of the following sets are not equivalent:

```
SIP:ALICE@AtLanTa.CoM;Transport=udp            (different usernames)
sip:alice@AtLanTa.CoM;Transport=UDP

sip:bob@biloxi.com                  (can resolve to different ports)
sip:bob@biloxi.com:5060

sip:bob@biloxi.com              (can resolve to different transports)
sip:bob@biloxi.com;transport=udp

sip:bob@biloxi.com     (can resolve to different port and transports)
sip:bob@biloxi.com:6000;transport=tcp

sip:carol@chicago.com                    (different header component)
sip:carol@chicago.com?Subject=next%20meeting

sip:bob@phone21.boxesbybob.com   (even though that's what
sip:bob@192.0.2.4                 phone21.boxesbybob.com resolves to)
```

Note that equality is not transitive:

- sip:carol@chicago.com and sip:carol@chicago.com;security=on are equivalent

- sip:carol@chicago.com and sip:carol@chicago.com;security=off are equivalent

- sip:carol@chicago.com;security=on and
  sip:carol@chicago.com;security=off are not equivalent

19.1.5 Forming Requests from a URI

An implementation needs to take care when forming requests directly from a URI. URIs from business cards, web pages, and even from sources inside the protocol such as registered contacts may contain inappropriate header fields or body parts.

An implementation MUST include any provided transport, maddr, ttl, or user parameter in the Request-URI of the formed request. If the URI contains a method parameter, its value MUST be used as the method of the request. The method parameter MUST NOT be placed in the Request-URI. Unknown URI parameters MUST be placed in the message's Request-URI.

An implementation SHOULD treat the presence of any headers or body parts in the URI as a desire to include them in the message, and choose to honor the request on a per-component basis.

An implementation SHOULD NOT honor these obviously dangerous header fields: From, Call-ID, CSeq, Via, and Record-Route.

An implementation SHOULD NOT honor any requested Route header field values in order to not be used as an unwitting agent in malicious attacks.

An implementation SHOULD NOT honor requests to include header fields that may cause it to falsely advertise its location or capabilities. These include: Accept, Accept-Encoding, Accept-Language, Allow, Contact (in its dialog usage), Organization, Supported, and User-Agent.

An implementation SHOULD verify the accuracy of any requested descriptive header fields, including: Content-Disposition, Content-Encoding, Content-Language, Content-Length, Content-Type, Date, Mime-Version, and Timestamp.

If the request formed from constructing a message from a given URI is not a valid SIP request, the URI is invalid. An implementation MUST NOT proceed with transmitting the request. It should instead pursue the course of action due an invalid URI in the context it occurs.

> The constructed request can be invalid in many ways. These include, but are not limited to, syntax error in header fields, invalid combinations of URI parameters, or an incorrect description of the message body.

Sending a request formed from a given URI may require capabilities unavailable to the implementation. The URI might indicate use of an unimplemented transport or extension, for example. An implementation SHOULD refuse to send these requests rather than modifying them to match their capabilities. An implementation MUST NOT send a request requiring an extension that it does not support.

For example, such a request can be formed through the presence of a Require header parameter or a method URI parameter with an unknown or explicitly unsupported value.

19.1.6 Relating SIP URIs and tel URLs

When a tel URL (RFC 2806 [9]) is converted to a SIP or SIPS URI, the entire telephone-subscriber portion of the tel URL, including any parameters, is placed into the userinfo part of the SIP or SIPS URI.

Thus, tel:+358-555-1234567;postd=pp22 becomes

sip:+358-555-1234567;postd=pp22@foo.com;user=phone

or

sips:+358-555-1234567;postd=pp22@foo.com;user=phone

not

sip:+358-555-1234567@foo.com;postd=pp22;user=phone

or

sips:+358-555-1234567@foo.com;postd=pp22;user=phone

In general, equivalent "tel" URLs converted to SIP or SIPS URIs in this fashion may not produce equivalent SIP or SIPS URIs. The userinfo of SIP and SIPS URIs are compared as a case-sensitive string. Variance in case-insensitive portions of tel URLs and reordering of tel URL parameters does not affect tel URL equivalence, but does affect the equivalence of SIP URIs formed from them.

For example,

tel:+358-555-1234567;postd=pp22
tel:+358-555-1234567;POSTD=PP22

are equivalent, while

sip:+358-555-1234567;postd=pp22@foo.com;user=phone
sip:+358-555-1234567;POSTD=PP22@foo.com;user=phone

are not.

Likewise,

tel:+358-555-1234567;postd=pp22;isub=1411
tel:+358-555-1234567;isub=1411;postd=pp22

are equivalent, while

sip:+358-555-1234567;postd=pp22;isub=1411@foo.com;user=phone
sip:+358-555-1234567;isub=1411;postd=pp22@foo.com;user=phone

are not.

To mitigate this problem, elements constructing telephone-subscriber fields to place in the userinfo part of a SIP or SIPS URI SHOULD fold any case-insensitive portion of telephone-subscriber to lower case, and order the telephone-subscriber parameters lexically by parameter name, excepting isdn-subaddress and post-dial, which occur first and in that order. (All components of a tel URL except for future-extension parameters are defined to be compared case-insensitive.)

Following this suggestion, both

tel:+358-555-1234567;postd=pp22
tel:+358-555-1234567;POSTD=PP22

become

sip:+358-555-1234567;postd=pp22@foo.com;user=phone

and both

tel:+358-555-1234567;tsp=a.b;phone-context=5
tel:+358-555-1234567;phone-context=5;tsp=a.b

become

sip:+358-555-1234567;phone-context=5;tsp=a.b@foo.com;user=phone

19.2 Option Tags

Option tags are unique identifiers used to designate new options (extensions) in SIP. These tags are used in Require (Section 20.32), Proxy-Require (Section 20.29), Supported (Section 20.37) and Unsupported (Section 20.40) header fields. Note that these options appear as parameters in those header fields in an option-tag = token form (see Section 25 for the definition of token).

Option tags are defined in standards track RFCs. This is a change from past practice, and is instituted to ensure continuing multi-vendor interoperability (see discussion in Section 20.32 and Section 20.37). An IANA registry of option tags is used to ensure easy reference.

19.3 Tags

The "tag" parameter is used in the To and From header fields of SIP messages. It serves as a general mechanism to identify a dialog, which is the combination of the Call-ID along with two tags, one from each participant in the dialog. When a UA sends a request outside of a dialog, it contains a From tag only, providing "half" of the dialog ID. The dialog is completed from the response(s), each of which contributes the second half in the To header field. The forking of SIP requests means that multiple dialogs can be established from a single request. This also explains the need for the two-sided dialog identifier; without a contribution from the recipients, the originator could not disambiguate the multiple dialogs established from a single request.

When a tag is generated by a UA for insertion into a request or response, it MUST be globally unique and cryptographically random with at least 32 bits of randomness. A property of this selection requirement is that a UA will place a different tag into the From header of an INVITE than it would place into the To header of the response to the same INVITE. This is needed in order for a UA to invite itself to a session, a common case for "hairpinning" of calls in PSTN gateways. Similarly, two INVITEs for different calls will have different From tags, and two responses for different calls will have different To tags.

Besides the requirement for global uniqueness, the algorithm for generating a tag is implementation-specific. Tags are helpful in fault tolerant systems, where a dialog is to be recovered on an alternate server after a failure. A UAS can select the tag in such a way that a backup can recognize a request as part of a dialog on the failed server, and therefore determine that it should attempt to recover the dialog and any other state associated with it.

20 Header Fields

The general syntax for header fields is covered in Section 7.3. This section lists the full set of header fields along with notes on syntax, meaning, and usage. Throughout this section, we use [HX.Y] to refer to Section X.Y of the current HTTP/1.1 specification RFC 2616 [8]. Examples of each header field are given.

Information about header fields in relation to methods and proxy processing is summarized in Tables 2 and 3.

The "where" column describes the request and response types in which the header field can be used. Values in this column are:

R: header field may only appear in requests;

r: header field may only appear in responses;

2xx, 4xx, etc.: A numerical value or range indicates response codes with which the header field can be used;

c: header field is copied from the request to the response.

An empty entry in the "where" column indicates that the header field may be present in all requests and responses.

The "proxy" column describes the operations a proxy may perform on a header field:

a: A proxy can add or concatenate the header field if not present.

m: A proxy can modify an existing header field value.

d: A proxy can delete a header field value.

r: A proxy must be able to read the header field, and thus this header field cannot be encrypted.

The next six columns relate to the presence of a header field in a method:

c: Conditional; requirements on the header field depend on the context of the message.

m: The header field is mandatory.

m*: The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.

o: The header field is optional.

t: The header field SHOULD be sent, but clients/servers need to be prepared to receive messages without that header field.

If a stream-based protocol (such as TCP) is used as a transport, then the header field MUST be sent.

*: The header field is required if the message body is not empty.
See Sections 20.14, 20.15 and 7.4 for details.

-: The header field is not applicable.

"Optional" means that an element MAY include the header field in a request or response, and a UA MAY ignore the header field if present in the request or response (The exception to this rule is the Require header field discussed in 20.32). A "mandatory" header field MUST be present in a request, and MUST be understood by the UAS receiving the request. A mandatory response header field MUST be present in the response, and the header field MUST be understood by the UAC processing the response. "Not applicable" means that the header field MUST NOT be present in a request. If one is placed in a request by mistake, it MUST be ignored by the UAS receiving the request. Similarly, a header field labeled "not applicable" for a response means that the UAS MUST NOT place the header field in the response, and the UAC MUST ignore the header field in the response.

A UA SHOULD ignore extension header parameters that are not understood.

A compact form of some common header field names is also defined for use when overall message size is an issue.

The Contact, From, and To header fields contain a URI. If the URI contains a comma, question mark or semicolon, the URI MUST be enclosed in angle brackets (< and >). Any URI parameters are contained within these brackets. If the URI is not enclosed in angle brackets, any semicolon-delimited parameters are header-parameters, not URI parameters.

20.1 Accept

The Accept header field follows the syntax defined in [H14.1]. The semantics are also identical, with the exception that if no Accept header field is present, the server SHOULD assume a default value of application/sdp.

An empty Accept header field means that no formats are acceptable.

Example:

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Accept	R		-	o	-	o	m*	o
Accept	2xx		-	-	-	o	m*	o
Accept	415		-	c	-	c	c	c
Accept-Encoding	R		-	o	-	o	o	o
Accept-Encoding	2xx		-	-	-	o	m*	o
Accept-Encoding	415		-	c	-	c	c	c
Accept-Language	R		-	o	-	o	o	o
Accept-Language	2xx		-	-	-	o	m*	o
Accept-Language	415		-	c	-	c	c	c
Alert-Info	R	ar	-	-	-	o	-	-
Alert-Info	180	ar	-	-	-	o	-	-
Allow	R		-	o	-	o	o	o
Allow	2xx		-	o	-	m*	m*	o
Allow	r		-	o	-	o	o	o
Allow	405		-	m	-	m	m	m
Authentication-Info	2xx		-	o	-	o	o	o
Authorization	R		o	o	o	o	o	o
Call-ID	c	r	m	m	m	m	m	m
Call-Info		ar	-	-	-	o	o	o
Contact	R		o	-	-	m	o	o
Contact	1xx		-	-	-	o	-	-
Contact	2xx		-	-	-	m	o	o
Contact	3xx	d	-	o	-	o	o	o
Contact	485		-	o	-	o	o	o
Content-Disposition			o	o	-	o	o	o
Content-Encoding			o	o	-	o	o	o
Content-Language			o	o	-	o	o	o
Content-Length		ar	t	t	t	t	t	t
Content-Type			*	*	-	*	*	*
CSeq	c	r	m	m	m	m	m	m
Date		a	o	o	o	o	o	o
Error-Info	300-699	a	-	o	o	o	o	o
Expires			-	-	-	o	-	o
From	c	r	m	m	m	m	m	m
In-Reply-To	R		-	-	-	o	-	-
Max-Forwards	R	amr	m	m	m	m	m	m
Min-Expires	423		-	-	-	-	-	m
MIME-Version			o	o	-	o	o	o
Organization		ar	-	-	-	o	o	o

Table 2: Summary of header fields, A--O

Header field	where	proxy	ACK	BYE	CAN	INV	OPT	REG
Priority	R	ar	-	-	-	o	-	-
Proxy-Authenticate	407	ar	-	m	-	m	m	m
Proxy-Authenticate	401	ar	-	o	o	o	o	o
Proxy-Authorization	R	dr	o	o	-	o	o	o
Proxy-Require	R	ar	-	o	-	o	o	o
Record-Route	R	ar	o	o	o	o	o	-
Record-Route	2xx,18x	mr	-	o	o	o	o	-
Reply-To			-	-	-	o	-	-
Require		ar	-	c	-	c	c	c
Retry-After	404,413,480,486		-	o	o	o	o	o
	500,503		-	o	o	o	o	o
	600,603		-	o	o	o	o	o
Route	R	adr	c	c	c	c	c	c
Server	r		-	o	o	o	o	o
Subject	R		-	-	-	o	-	-
Supported	R		-	o	o	m*	o	o
Supported	2xx		-	o	o	m*	m*	o
Timestamp			o	o	o	o	o	o
To	c(1)	r	m	m	m	m	m	m
Unsupported	420		-	m	-	m	m	m
User-Agent			o	o	o	o	o	o
Via	R	amr	m	m	m	m	m	m
Via	rc	dr	m	m	m	m	m	m
Warning	r		-	o	o	o	o	o
WWW-Authenticate	401	ar	-	m	-	m	m	m
WWW-Authenticate	407	ar	-	o	-	o	o	o

Table 3: Summary of header fields, P--Z; (1): copied with possible addition of tag

Accept: application/sdp;level=1, application/x-private, text/html

20.2 Accept-Encoding

The Accept-Encoding header field is similar to Accept, but restricts the content-codings [H3.5] that are acceptable in the response. See [H14.3]. The semantics in SIP are identical to those defined in [H14.3].

An empty Accept-Encoding header field is permissible. It is equivalent to Accept-Encoding: identity, that is, only the identity encoding, meaning no encoding, is permissible.

If no Accept-Encoding header field is present, the server SHOULD assume a default value of identity.

This differs slightly from the HTTP definition, which indicates that when not present, any encoding can be used, but the identity encoding is preferred.

Example:

```
Accept-Encoding: gzip
```

20.3 Accept-Language

The Accept-Language header field is used in requests to indicate the preferred languages for reason phrases, session descriptions, or status responses carried as message bodies in the response. If no Accept-Language header field is present, the server SHOULD assume all languages are acceptable to the client.

The Accept-Language header field follows the syntax defined in [H14.4]. The rules for ordering the languages based on the "q" parameter apply to SIP as well.

Example:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

20.4 Alert-Info

When present in an INVITE request, the Alert-Info header field specifies an alternative ring tone to the UAS. When present in a 180 (Ringing) response, the Alert-Info header field specifies an alternative ringback tone to the UAC. A typical usage is for a proxy to insert this header field to provide a distinctive ring feature.

The Alert-Info header field can introduce security risks. These risks and the ways to handle them are discussed in Section 20.9, which discusses the Call-Info header field since the risks are identical.

In addition, a user SHOULD be able to disable this feature selectively.

This helps prevent disruptions that could result from the use of this header field by untrusted elements.

Example:

```
Alert-Info: <http://www.example.com/sounds/moo.wav>
```

20.5 Allow

The Allow header field lists the set of methods supported by the UA generating the message.

All methods, including ACK and CANCEL, understood by the UA MUST be included in the list of methods in the Allow header field, when present. The absence of an Allow header field MUST NOT be interpreted to mean that the UA sending the message supports no methods. Rather, it implies that the UA is not providing any information on what methods it supports.

Supplying an Allow header field in responses to methods other than OPTIONS reduces the number of messages needed.

Example:

```
Allow: INVITE, ACK, OPTIONS, CANCEL, BYE
```

20.6 Authentication-Info

The Authentication-Info header field provides for mutual authentication with HTTP Digest. A UAS MAY include this header field in a 2xx response to a request that was successfully authenticated using digest based on the Authorization header field.

Syntax and semantics follow those specified in RFC 2617 [17].

Example:

```
Authentication-Info: nextnonce="47364c23432d2e131a5fb210812c"
```

20.7 Authorization

The Authorization header field contains authentication credentials of a UA. Section 22.2 overviews the use of the Authorization header field, and Section 22.4 describes the syntax and semantics when used with HTTP authentication.

This header field, along with Proxy-Authorization, breaks the general rules about multiple header field values. Although not a comma-separated list, this header field name may be present multiple times, and MUST NOT be combined into a single header line using the usual rules described in Section 7.3.

In the example below, there are no quotes around the Digest parameter:

```
Authorization: Digest username="Alice", realm="atlanta.com",
 nonce="84a4cc6f3082121f32b42a2187831a9e",
 response="7587245234b3434cc3412213e5f113a5432"
```

20.8 Call-ID

The Call-ID header field uniquely identifies a particular invitation or all registrations of a particular client. A single multimedia conference can give rise to several calls with different Call-IDs, for example, if a user invites a single individual several times to the same (long-running) conference. Call-IDs are case-sensitive and are simply compared byte-by-byte.

The compact form of the Call-ID header field is i.

Examples:

```
Call-ID: f81d4fae-7dec-11d0-a765-00a0c91e6bf6@biloxi.com
i:f81d4fae-7dec-11d0-a765-00a0c91e6bf6@192.0.2.4
```

20.9 Call-Info

The Call-Info header field provides additional information about the caller or callee, depending on whether it is found in a request or response. The purpose of the URI is described by the "purpose" parameter. The "icon" parameter designates an image suitable as an iconic representation of the caller or callee. The "info" parameter describes the caller or callee in general, for example, through a web page. The "card" parameter provides a business card, for example, in vCard [36] or LDIF [37] formats. Additional tokens can be registered using IANA and the procedures in Section 27.

Use of the Call-Info header field can pose a security risk. If a callee fetches the URIs provided by a malicious caller, the callee may be at risk for displaying inappropriate or offensive content, dangerous or illegal content, and so on. Therefore, it is RECOMMENDED that a UA only render the information in the Call-Info header field if it can verify the authenticity of the element that originated the header field and trusts that element. This need not be the peer UA; a proxy can insert this header field into requests.

Example:

```
Call-Info: <http://wwww.example.com/alice/photo.jpg> ;purpose=icon,
  <http://www.example.com/alice/> ;purpose=info
```

20.10 Contact

A Contact header field value provides a URI whose meaning depends on the type of request or response it is in.

A Contact header field value can contain a display name, a URI with URI parameters, and header parameters.

This document defines the Contact parameters "q" and "expires". These parameters are only used when the Contact is present in a REGISTER request or response, or in a 3xx response. Additional parameters may be defined in other specifications.

When the header field value contains a display name, the URI including all URI parameters is enclosed in "<" and ">". If no "<" and ">" are present, all parameters after the URI are header parameters, not URI parameters. The display name can be tokens, or a quoted string, if a larger character set is desired.

Even if the "display-name" is empty, the "name-addr" form MUST be used if the "addr-spec" contains a comma, semicolon, or question mark. There may or may not be LWS between the display-name and the "<".

These rules for parsing a display name, URI and URI parameters, and header parameters also apply for the header fields To and From.

> The Contact header field has a role similar to the Location header field in HTTP. However, the HTTP header field only allows one address, unquoted. Since URIs can contain commas and semicolons as reserved characters, they can be mistaken for header or parameter delimiters, respectively.

The compact form of the Contact header field is m (for "moved").

Examples:

```
Contact: "Mr. Watson" <sip:watson@worcester.bell-telephone.com>
   ;q=0.7; expires=3600,
   "Mr. Watson" <mailto:watson@bell-telephone.com> ;q=0.1
m: <sips:bob@192.0.2.4>;expires=60
```

20.11 Content-Disposition

The Content-Disposition header field describes how the message body or, for multipart messages, a message body part is to be interpreted by the UAC or UAS. This SIP header field extends the MIME Content-Type (RFC 2183 [18]).

Several new "disposition-types" of the Content-Disposition header are defined by SIP. The value "session" indicates that the body part describes a session, for either calls or early (pre-call) media. The value "render" indicates that the body part should be displayed or otherwise rendered to the user. Note that the value "render" is used rather than "inline" to avoid the connotation that the MIME body is displayed as a part of the rendering of the entire message (since the MIME bodies of SIP messages oftentimes are not displayed to users). For backward-compatibility, if the Content-Disposition header field is missing, the server SHOULD assume bodies of Content-Type application/sdp are the disposition "session", while other content types are "render".

The disposition type "icon" indicates that the body part contains an image suitable as an iconic representation of the caller or callee that could be rendered informationally by a user agent when a message has been received, or persistently while a dialog takes place. The value "alert" indicates that the body part contains information, such as an audio clip, that should be rendered by the user agent in an attempt to alert the user to the receipt of a request, generally a request that initiates a dialog; this alerting body could for example be rendered as a ring tone for a phone call after a 180 Ringing provisional response has been sent.

Any MIME body with a "disposition-type" that renders content to the user should only be processed when a message has been properly authenticated.

The handling parameter, handling-param, describes how the UAS should react if it receives a message body whose content type or disposition type it does not understand. The parameter has defined values of "optional" and "required". If the handling parameter is missing, the value "required" SHOULD be assumed. The handling parameter is described in RFC 3204 [19].

If this header field is missing, the MIME type determines the default content disposition. If there is none, "render" is assumed.

Example:

```
Content-Disposition: session
```

20.12 Content-Encoding

The Content-Encoding header field is used as a modifier to the "media-type". When present, its value indicates what additional content codings have been applied to the entity-body, and thus what decoding mechanisms MUST be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a body to be compressed without losing the identity of its underlying media type.

If multiple encodings have been applied to an entity-body, the content codings MUST be listed in the order in which they were applied.

All content-coding values are case-insensitive. IANA acts as a registry for content-coding value tokens. See [H3.5] for a definition of the syntax for content-coding.

Clients MAY apply content encodings to the body in requests. A server MAY apply content encodings to the bodies in responses. The server MUST only use encodings listed in the Accept-Encoding header field in the request.

The compact form of the Content-Encoding header field is e. Examples:

```
Content-Encoding: gzip
e: tar
```

20.13 Content-Language

See [H14.12]. Example:

```
Content-Language: fr
```

20.14 Content-Length

The Content-Length header field indicates the size of the message-body, in decimal number of octets, sent to the recipient. Applications SHOULD use this field to indicate the size of the message-body to be transferred, regardless of the media type of the entity. If a stream-based protocol (such as TCP) is used as transport, the header field MUST be used.

The size of the message-body does not include the CRLF separating header fields and body. Any Content-Length greater than or equal to zero is a valid value. If no body is present in a message, then the Content-Length header field value MUST be set to zero.

The ability to omit Content-Length simplifies the creation of cgi-like scripts that dynamically generate responses.

The compact form of the header field is l.

Examples:

```
Content-Length: 349
l: 173
```

20.15 Content-Type

The Content-Type header field indicates the media type of the message-body sent to the recipient. The "media-type" element is defined in [H3.7]. The Content-Type header field MUST be present if the body is not empty. If the body is empty, and a Content-Type header field is present, it indicates that the body of the specific type has zero length (for example, an empty audio file).

The compact form of the header field is c.

Examples:

```
Content-Type: application/sdp
c: text/html; charset=ISO-8859-4
```

20.16 CSeq

A CSeq header field in a request contains a single decimal sequence number and the request method. The sequence number MUST be expressible as a 32-bit unsigned integer. The method part of CSeq is case-sensitive. The CSeq header field serves to order transactions within a dialog, to provide a means to uniquely identify transactions, and to differentiate between new requests and request retransmissions. Two CSeq header fields are considered equal if the sequence number and the request method are identical. Example:

```
CSeq: 4711 INVITE
```

20.17 Date

The Date header field contains the date and time. Unlike HTTP/1.1, SIP only supports the most recent RFC 1123 [20] format for dates. As in [H3.3], SIP restricts the time zone in SIP-date to "GMT", while RFC 1123 allows any time zone. An RFC 1123 date is case-sensitive.

The Date header field reflects the time when the request or response is first sent.

The Date header field can be used by simple end systems without a battery-backed clock to acquire a notion of current time. However, in its GMT form, it requires clients to know their offset from GMT.

Example:

```
Date: Sat, 13 Nov 2010 23:29:00 GMT
```

20.18 Error-Info

The Error-Info header field provides a pointer to additional information about the error status response.

SIP UACs have user interface capabilities ranging from pop-up windows and audio on PC softclients to audio-only on "black" phones or endpoints connected via gateways. Rather than forcing a server generating an error to choose between sending an error status code with a detailed reason phrase and playing an audio recording, the Error-Info header field allows both to be sent. The UAC then has the choice of which error indicator to render to the caller.

A UAC MAY treat a SIP or SIPS URI in an Error-Info header field as if it were a Contact in a redirect and generate a new INVITE, resulting in a recorded announcement session being established. A non-SIP URI MAY be rendered to the user.

Examples:

```
SIP/2.0 404 The number you have dialed is not in service
Error-Info: <sip:not-in-service-recording@atlanta.com>
```

20.19 Expires

The Expires header field gives the relative time after which the message (or content) expires.

The precise meaning of this is method dependent.

The expiration time in an INVITE does not affect the duration of the actual session that may result from the invitation. Session description protocols may offer the ability to express time limits on the session duration, however.

The value of this field is an integral number of seconds (in decimal) between 0 and (2**32)-1, measured from the receipt of the request.

Example:

Expires: 5

20.20 From

The From header field indicates the initiator of the request. This may be different from the initiator of the dialog. Requests sent by the callee to the caller use the callee's address in the From header field.

The optional "display-name" is meant to be rendered by a human user interface. A system SHOULD use the display name "Anonymous" if the identity of the client is to remain hidden. Even if the "display-name" is empty, the "name-addr" form MUST be used if the "addr-spec" contains a comma, question mark, or semicolon. Syntax issues are discussed in Section 7.3.1.

Two From header fields are equivalent if their URIs match, and their parameters match. Extension parameters in one header field, not present in the other are ignored for the purposes of comparison. This means that the display name and presence or absence of angle brackets do not affect matching.

See Section 20.10 for the rules for parsing a display name, URI and URI parameters, and header field parameters.

The compact form of the From header field is f.

Examples:

```
From: "A. G. Bell" <sip:agb@bell-telephone.com> ;tag=a48s
From: sip:+12125551212@server.phone2net.com;tag=887s
f: Anonymous <sip:c8oqz84zk7z@privacy.org>;tag=hyh8
```

20.21 In-Reply-To

The In-Reply-To header field enumerates the Call-IDs that this call references or returns. These Call-IDs may have been cached by the client then included in this header field in a return call.

This allows automatic call distribution systems to route return calls to the originator of the first call. This also allows callees to filter calls, so that only return calls for calls they originated will be accepted. This field is not a substitute for request authentication.

Example:

```
In-Reply-To: 70710@saturn.bell-tel.com, 17320@saturn.bell-tel.com
```

20.22 Max-Forwards

The Max-Forwards header field must be used with any SIP method to limit the number of proxies or gateways that can forward the request to the next downstream server. This can also be useful when the client is attempting to trace a request chain that appears to be failing or looping in mid-chain.

The Max-Forwards value is an integer in the range 0-255 indicating the remaining number of times this request message is allowed to be forwarded. This count is decremented by each server that forwards the request. The recommended initial value is 70.

This header field should be inserted by elements that can not otherwise guarantee loop detection. For example, a B2BUA should insert a Max-Forwards header field.

Example:

```
Max-Forwards: 6
```

20.23 Min-Expires

The Min-Expires header field conveys the minimum refresh interval supported for soft-state elements managed by that server. This includes Contact header fields that are stored by a registrar. The header field contains a decimal integer number of seconds from 0 to (2**32)-1. The use of the header field in a 423 (Interval Too Brief) response is described in Sections 10.2.8, 10.3, and 21.4.17.

Example:

```
Min-Expires: 60
```

20.24 MIME-Version

See [H19.4.1].

Example:

```
MIME-Version: 1.0
```

20.25 Organization

The Organization header field conveys the name of the organization to which the SIP element issuing the request or response belongs.

The field MAY be used by client software to filter calls.

Example:

```
Organization: Boxes by Bob
```

20.26 Priority

The Priority header field indicates the urgency of the request as perceived by the client. The Priority header field describes the priority that the SIP request should have to the receiving human or its agent. For example, it may be factored into decisions about call routing and acceptance. For these decisions, a message containing no Priority header field SHOULD be treated as if it specified a Priority of "normal". The Priority header field does not influence the use of communications resources such as packet forwarding priority in routers or access to circuits in PSTN gateways. The header field can have the values "non-urgent", "normal", "urgent", and "emergency", but additional values can be defined elsewhere. It is RECOMMENDED that the value of "emergency" only be used when life, limb, or property are in imminent danger. Otherwise, there are no semantics defined for this header field.

These are the values of RFC 2076 [38], with the addition of "emergency".

Examples:

```
Subject: A tornado is heading our way!
Priority: emergency
```

or

```
Subject: Weekend plans
Priority: non-urgent
```

20.27 Proxy-Authenticate

A Proxy-Authenticate header field value contains an authentication challenge.

The use of this header field is defined in [H14.33]. See Section 22.3 for further details on its usage.

Example:

```
Proxy-Authenticate: Digest realm="atlanta.com",
 domain="sip:ss1.carrier.com", qop="auth",
 nonce="f84f1cec41e6cbe5aea9c8e88d359",
 opaque="", stale=FALSE, algorithm=MD5
```

20.28 Proxy-Authorization

The Proxy-Authorization header field allows the client to identify itself (or its user) to a proxy that requires authentication. A Proxy-Authorization field value consists of credentials containing the authentication information of the user agent for the proxy and/or realm of the resource being requested.

See Section 22.3 for a definition of the usage of this header field.

This header field, along with Authorization, breaks the general rules about multiple header field names. Although not a comma-separated list, this header field name may be present multiple times, and MUST NOT be combined into a single header line using the usual rules described in Section 7.3.1.

Example:

```
Proxy-Authorization: Digest username="Alice", realm="atlanta.com",
   nonce="c60f3082ee1212b402a21831ae",
   response="245f23415f11432b3434341c022"
```

20.29 Proxy-Require

The Proxy-Require header field is used to indicate proxy-sensitive features that must be supported by the proxy. See Section 20.32 for more details on the mechanics of this message and a usage example.

Example:

```
Proxy-Require: foo
```

20.30 Record-Route

The Record-Route header field is inserted by proxies in a request to force future requests in the dialog to be routed through the proxy.

Examples of its use with the Route header field are described in Sections 16.12.1.

Example:

```
Record-Route: <sip:server10.biloxi.com;lr>,
              <sip:bigbox3.site3.atlanta.com;lr>
```

20.31 Reply-To

The Reply-To header field contains a logical return URI that may be different from the From header field. For example, the URI MAY be used to return missed calls or unestablished sessions. If the user wished to remain anonymous, the header field SHOULD either be omitted from the request or populated in such a way that does not reveal any private information.

Even if the "display-name" is empty, the "name-addr" form MUST be used if the "addr-spec" contains a comma, question mark, or semicolon. Syntax issues are discussed in Section 7.3.1.

Example:

```
Reply-To: Bob <sip:bob@biloxi.com>
```

20.32 Require

The Require header field is used by UACs to tell UASs about options that the UAC expects the UAS to support in order to process the request. Although an optional header field, the Require MUST NOT be ignored if it is present.

The Require header field contains a list of option tags, described in Section 19.2. Each option tag defines a SIP extension that MUST be understood to process the request. Frequently, this is used to indicate that a specific set of extension header fields need to be understood. A UAC compliant to this specification MUST only include option tags corresponding to standards-track RFCs.

Example:

```
Require: 100rel
```

20.33 Retry-After

The Retry-After header field can be used with a 500 (Server Internal Error) or 503 (Service Unavailable) response to indicate how long the service is expected to be unavailable to the requesting client and with a 404 (Not Found), 413 (Request Entity Too Large), 480 (Temporarily Unavailable), 486 (Busy Here), 600 (Busy), or 603

(Decline) response to indicate when the called party anticipates being available again. The value of this field is a positive integer number of seconds (in decimal) after the time of the response.

An optional comment can be used to indicate additional information about the time of callback. An optional "duration" parameter indicates how long the called party will be reachable starting at the initial time of availability. If no duration parameter is given, the service is assumed to be available indefinitely.

Examples:

```
Retry-After: 18000;duration=3600
Retry-After: 120 (I'm in a meeting)
```

20.34 Route

The Route header field is used to force routing for a request through the listed set of proxies. Examples of the use of the Route header field are in Section 16.12.1.

Example:

```
Route: <sip:bigbox3.site3.atlanta.com;lr>,
       <sip:server10.biloxi.com;lr>
```

20.35 Server

The Server header field contains information about the software used by the UAS to handle the request.

Revealing the specific software version of the server might allow the server to become more vulnerable to attacks against software that is known to contain security holes. Implementers SHOULD make the Server header field a configurable option.

Example:

```
Server: HomeServer v2
```

20.36 Subject

The Subject header field provides a summary or indicates the nature of the call, allowing call filtering without having to parse the session description. The session description does not have to use the same subject indication as the invitation.

The compact form of the Subject header field is s.

Example:

```
Subject: Need more boxes
s: Tech Support
```

20.37 Supported

The Supported header field enumerates all the extensions supported by the UAC or UAS.

The Supported header field contains a list of option tags, described in Section 19.2, that are understood by the UAC or UAS. A UA compliant to this specification MUST only include option tags corresponding to standards-track RFCs. If empty, it means that no extensions are supported.

The compact form of the Supported header field is k.

Example:

```
Supported: 100rel
```

20.38 Timestamp

The Timestamp header field describes when the UAC sent the request to the UAS.

See Section 8.2.6 for details on how to generate a response to a request that contains the header field. Although there is no normative behavior defined here that makes use of the header, it allows for extensions or SIP applications to obtain RTT estimates.

Example:

```
Timestamp: 54
```

20.39 To

The To header field specifies the logical recipient of the request.

The optional "display-name" is meant to be rendered by a human-user interface. The "tag" parameter serves as a general mechanism for dialog identification.

See Section 19.3 for details of the "tag" parameter.

Comparison of To header fields for equality is identical to comparison of From header fields. See Section 20.10 for the rules for parsing a display name, URI and URI parameters, and header field parameters.

The compact form of the To header field is t.

The following are examples of valid To header fields:

```
To: The Operator <sip:operator@cs.columbia.edu>;tag=287447
t: sip:+12125551212@server.phone2net.com
```

20.40 Unsupported

The Unsupported header field lists the features not supported by the UAS. See Section 20.32 for motivation.

Example:

```
Unsupported: foo
```

20.41 User-Agent

The User-Agent header field contains information about the UAC originating the request. The semantics of this header field are defined in [H14.43].

Revealing the specific software version of the user agent might allow the user agent to become more vulnerable to attacks against software that is known to contain security holes. Implementers SHOULD make the User-Agent header field a configurable option.

Example:

```
User-Agent: Softphone Beta1.5
```

20.42 Via

The Via header field indicates the path taken by the request so far and indicates the path that should be followed in routing responses. The branch ID parameter in the Via header field values serves as a transaction identifier, and is used by proxies to detect loops.

A Via header field value contains the transport protocol used to send the message, the client's host name or network address, and possibly the port number at which it wishes to receive responses. A Via header field value can also contain parameters such as "maddr", "ttl", "received", and "branch", whose meaning and use are described

in other sections. For implementations compliant to this specification, the value of the branch parameter MUST start with the magic cookie "z9hG4bK", as discussed in Section 8.1.1.7.

Transport protocols defined here are "UDP", "TCP", "TLS", and "SCTP". "TLS" means TLS over TCP. When a request is sent to a SIPS URI, the protocol still indicates "SIP", and the transport protocol is TLS.

```
Via: SIP/2.0/UDP erlang.bell-telephone.com:5060;branch=z9hG4bK87asdks7
Via: SIP/2.0/UDP 192.0.2.1:5060 ;received=192.0.2.207
     ;branch=z9hG4bK77asjd
```

The compact form of the Via header field is v.

In this example, the message originated from a multi-homed host with two addresses, 192.0.2.1 and 192.0.2.207. The sender guessed wrong as to which network interface would be used. Erlang.bell-telephone.com noticed the mismatch and added a parameter to the previous hop's Via header field value, containing the address that the packet actually came from.

The host or network address and port number are not required to follow the SIP URI syntax. Specifically, LWS on either side of the ":" or "/" is allowed, as shown here:

```
   Via: SIP / 2.0 / UDP first.example.com: 4000;ttl=16
     ;maddr=224.2.0.1 ;branch=z9hG4bKa7c6a8dlze.1
```

Even though this specification mandates that the branch parameter be present in all requests, the BNF for the header field indicates that it is optional. This allows interoperation with RFC 2543 elements, which did not have to insert the branch parameter.

Two Via header fields are equal if their sent-protocol and sent-by fields are equal, both have the same set of parameters, and the values of all parameters are equal.

20.43 Warning

The Warning header field is used to carry additional information about the status of a response. Warning header field values are sent with responses and contain a three-digit warning code, host name, and warning text.

The "warn-text" should be in a natural language that is most likely to be intelligible to the human user receiving the response. This decision can be based on any available knowledge, such as the location of the user, the Accept-Language field in a request, or the

Content-Language field in a response. The default language is i-default [21].

The currently-defined "warn-code"s are listed below, with a recommended warn-text in English and a description of their meaning. These warnings describe failures induced by the session description. The first digit of warning codes beginning with "3" indicates warnings specific to SIP. Warnings 300 through 329 are reserved for indicating problems with keywords in the session description, 330 through 339 are warnings related to basic network services requested in the session description, 370 through 379 are warnings related to quantitative QoS parameters requested in the session description, and 390 through 399 are miscellaneous warnings that do not fall into one of the above categories.

300 Incompatible network protocol: One or more network protocols contained in the session description are not available.

301 Incompatible network address formats: One or more network address formats contained in the session description are not available.

302 Incompatible transport protocol: One or more transport protocols described in the session description are not available.

303 Incompatible bandwidth units: One or more bandwidth measurement units contained in the session description were not understood.

304 Media type not available: One or more media types contained in the session description are not available.

305 Incompatible media format: One or more media formats contained in the session description are not available.

306 Attribute not understood: One or more of the media attributes in the session description are not supported.

307 Session description parameter not understood: A parameter other than those listed above was not understood.

330 Multicast not available: The site where the user is located does not support multicast.

331 Unicast not available: The site where the user is located does not support unicast communication (usually due to the presence of a firewall).

370 Insufficient bandwidth: The bandwidth specified in the session description or defined by the media exceeds that known to be available.

399 Miscellaneous warning: The warning text can include arbitrary information to be presented to a human user or logged. A system receiving this warning MUST NOT take any automated action.

1xx and 2xx have been taken by HTTP/1.1.

Additional "warn-code"s can be defined through IANA, as defined in Section 27.2.

Examples:

```
Warning: 307 isi.edu "Session parameter 'foo' not understood"
Warning: 301 isi.edu "Incompatible network address type 'E.164'"
```

20.44 WWW-Authenticate

A WWW-Authenticate header field value contains an authentication challenge. See Section 22.2 for further details on its usage.

Example:

```
WWW-Authenticate: Digest realm="atlanta.com",
  domain="sip:boxesbybob.com", qop="auth",
  nonce="f84f1cec41e6cbe5aea9c8e88d359",
  opaque="", stale=FALSE, algorithm=MD5
```

21 Response Codes

The response codes are consistent with, and extend, HTTP/1.1 response codes. Not all HTTP/1.1 response codes are appropriate, and only those that are appropriate are given here. Other HTTP/1.1 response codes SHOULD NOT be used. Also, SIP defines a new class, 6xx.

21.1 Provisional 1xx

Provisional responses, also known as informational responses, indicate that the server contacted is performing some further action and does not yet have a definitive response. A server sends a 1xx response if it expects to take more than 200 ms to obtain a final response. Note that 1xx responses are not transmitted reliably. They never cause the client to send an ACK. Provisional (1xx) responses MAY contain message bodies, including session descriptions.

21.1.1 100 Trying

This response indicates that the request has been received by the next-hop server and that some unspecified action is being taken on behalf of this call (for example, a database is being consulted). This response, like all other provisional responses, stops retransmissions of an INVITE by a UAC. The 100 (Trying) response is different from other provisional responses, in that it is never forwarded upstream by a stateful proxy.

21.1.2 180 Ringing

The UA receiving the INVITE is trying to alert the user. This response MAY be used to initiate local ringback.

21.1.3 181 Call Is Being Forwarded

A server MAY use this status code to indicate that the call is being forwarded to a different set of destinations.

21.1.4 182 Queued

The called party is temporarily unavailable, but the server has decided to queue the call rather than reject it. When the callee becomes available, it will return the appropriate final status response. The reason phrase MAY give further details about the status of the call, for example, "5 calls queued; expected waiting time is 15 minutes". The server MAY issue several 182 (Queued) responses to update the caller about the status of the queued call.

21.1.5 183 Session Progress

The 183 (Session Progress) response is used to convey information about the progress of the call that is not otherwise classified. The Reason-Phrase, header fields, or message body MAY be used to convey more details about the call progress.

21.2 Successful 2xx

The request was successful.

21.2.1 200 OK

The request has succeeded. The information returned with the response depends on the method used in the request.

21.3 Redirection 3xx

3xx responses give information about the user's new location, or about alternative services that might be able to satisfy the call.

21.3.1 300 Multiple Choices

The address in the request resolved to several choices, each with its own specific location, and the user (or UA) can select a preferred communication end point and redirect its request to that location.

The response MAY include a message body containing a list of resource characteristics and location(s) from which the user or UA can choose the one most appropriate, if allowed by the Accept request header field. However, no MIME types have been defined for this message body.

The choices SHOULD also be listed as Contact fields (Section 20.10). Unlike HTTP, the SIP response MAY contain several Contact fields or a list of addresses in a Contact field. UAs MAY use the Contact header field value for automatic redirection or MAY ask the user to confirm a choice. However, this specification does not define any standard for such automatic selection.

This status response is appropriate if the callee can be reached at several different locations and the server cannot or prefers not to proxy the request.

21.3.2 301 Moved Permanently

The user can no longer be found at the address in the Request-URI, and the requesting client SHOULD retry at the new address given by the Contact header field (Section 20.10). The requestor SHOULD update any local directories, address books, and user location caches with this new value and redirect future requests to the address(es) listed.

21.3.3 302 Moved Temporarily

The requesting client SHOULD retry the request at the new address(es) given by the Contact header field (Section 20.10). The Request-URI of the new request uses the value of the Contact header field in the response.

The duration of the validity of the Contact URI can be indicated through an Expires (Section 20.19) header field or an expires parameter in the Contact header field. Both proxies and UAs MAY cache this URI for the duration of the expiration time. If there is no explicit expiration time, the address is only valid once for recursing, and MUST NOT be cached for future transactions.

If the URI cached from the Contact header field fails, the Request-URI from the redirected request MAY be tried again a single time.

> The temporary URI may have become out-of-date sooner than the expiration time, and a new temporary URI may be available.

21.3.4 305 Use Proxy

The requested resource MUST be accessed through the proxy given by the Contact field. The Contact field gives the URI of the proxy. The recipient is expected to repeat this single request via the proxy. 305 (Use Proxy) responses MUST only be generated by UASs.

21.3.5 380 Alternative Service

The call was not successful, but alternative services are possible.

The alternative services are described in the message body of the response. Formats for such bodies are not defined here, and may be the subject of future standardization.

21.4 Request Failure 4xx

4xx responses are definite failure responses from a particular server. The client SHOULD NOT retry the same request without modification (for example, adding appropriate authorization). However, the same request to a different server might be successful.

21.4.1 400 Bad Request

The request could not be understood due to malformed syntax. The Reason-Phrase SHOULD identify the syntax problem in more detail, for example, "Missing Call-ID header field".

21.4.2 401 Unauthorized

The request requires user authentication. This response is issued by UASs and registrars, while 407 (Proxy Authentication Required) is used by proxy servers.

21.4.3 402 Payment Required

Reserved for future use.

21.4.4 403 Forbidden

The server understood the request, but is refusing to fulfill it. Authorization will not help, and the request SHOULD NOT be repeated.

21.4.5 404 Not Found

The server has definitive information that the user does not exist at the domain specified in the Request-URI. This status is also returned if the domain in the Request-URI does not match any of the domains handled by the recipient of the request.

21.4.6 405 Method Not Allowed

The method specified in the Request-Line is understood, but not allowed for the address identified by the Request-URI.

The response MUST include an Allow header field containing a list of valid methods for the indicated address.

21.4.7 406 Not Acceptable

The resource identified by the request is only capable of generating response entities that have content characteristics not acceptable according to the Accept header field sent in the request.

21.4.8 407 Proxy Authentication Required

This code is similar to 401 (Unauthorized), but indicates that the client MUST first authenticate itself with the proxy. SIP access authentication is explained in Sections 26 and 22.3.

This status code can be used for applications where access to the communication channel (for example, a telephony gateway) rather than the callee requires authentication.

21.4.9 408 Request Timeout

The server could not produce a response within a suitable amount of time, for example, if it could not determine the location of the user in time. The client MAY repeat the request without modifications at any later time.

21.4.10 410 Gone

The requested resource is no longer available at the server and no forwarding address is known. This condition is expected to be considered permanent. If the server does not know, or has no facility to determine, whether or not the condition is permanent, the status code 404 (Not Found) SHOULD be used instead.

21.4.11 413 Request Entity Too Large

The server is refusing to process a request because the request entity-body is larger than the server is willing or able to process. The server MAY close the connection to prevent the client from continuing the request.

If the condition is temporary, the server SHOULD include a Retry-After header field to indicate that it is temporary and after what time the client MAY try again.

21.4.12 414 Request-URI Too Long

The server is refusing to service the request because the Request-URI is longer than the server is willing to interpret.

21.4.13 415 Unsupported Media Type

The server is refusing to service the request because the message body of the request is in a format not supported by the server for the requested method. The server MUST return a list of acceptable formats using the Accept, Accept-Encoding, or Accept-Language header field, depending on the specific problem with the content. UAC processing of this response is described in Section 8.1.3.5.

21.4.14 416 Unsupported URI Scheme

The server cannot process the request because the scheme of the URI in the Request-URI is unknown to the server. Client processing of this response is described in Section 8.1.3.5.

21.4.15 420 Bad Extension

The server did not understand the protocol extension specified in a Proxy-Require (Section 20.29) or Require (Section 20.32) header field. The server MUST include a list of the unsupported extensions in an Unsupported header field in the response. UAC processing of this response is described in Section 8.1.3.5.

21.4.16 421 Extension Required

The UAS needs a particular extension to process the request, but this extension is not listed in a Supported header field in the request. Responses with this status code MUST contain a Require header field listing the required extensions.

A UAS SHOULD NOT use this response unless it truly cannot provide any useful service to the client. Instead, if a desirable extension is not listed in the Supported header field, servers SHOULD process the request using baseline SIP capabilities and any extensions supported by the client.

21.4.17 423 Interval Too Brief

The server is rejecting the request because the expiration time of the resource refreshed by the request is too short. This response can be used by a registrar to reject a registration whose Contact header field expiration time was too small. The use of this response and the related Min-Expires header field are described in Sections 10.2.8, 10.3, and 20.23.

21.4.18 480 Temporarily Unavailable

The callee's end system was contacted successfully but the callee is currently unavailable (for example, is not logged in, logged in but in a state that precludes communication with the callee, or has activated the "do not disturb" feature). The response MAY indicate a better time to call in the Retry-After header field. The user could also be available elsewhere (unbeknownst to this server). The reason phrase SHOULD indicate a more precise cause as to why the callee is unavailable. This value SHOULD be settable by the UA. Status 486 (Busy Here) MAY be used to more precisely indicate a particular reason for the call failure.

This status is also returned by a redirect or proxy server that recognizes the user identified by the Request-URI, but does not currently have a valid forwarding location for that user.

21.4.19 481 Call/Transaction Does Not Exist

This status indicates that the UAS received a request that does not match any existing dialog or transaction.

21.4.20 482 Loop Detected

The server has detected a loop (Section 16.3 Item 4).

21.4.21 483 Too Many Hops

The server received a request that contains a Max-Forwards (Section 20.22) header field with the value zero.

21.4.22 484 Address Incomplete

The server received a request with a Request-URI that was incomplete. Additional information SHOULD be provided in the reason phrase.

This status code allows overlapped dialing. With overlapped dialing, the client does not know the length of the dialing string. It sends strings of increasing lengths, prompting the user for more input, until it no longer receives a 484 (Address Incomplete) status response.

21.4.23 485 Ambiguous

The Request-URI was ambiguous. The response MAY contain a listing of possible unambiguous addresses in Contact header fields. Revealing alternatives can infringe on privacy of the user or the organization. It MUST be possible to configure a server to respond with status 404 (Not Found) or to suppress the listing of possible choices for ambiguous Request-URIs.

Example response to a request with the Request-URI sip:lee@example.com:

```
SIP/2.0 485 Ambiguous
Contact: Carol Lee <sip:carol.lee@example.com>
Contact: Ping Lee <sip:p.lee@example.com>
Contact: Lee M. Foote <sips:lee.foote@example.com>
```

Some email and voice mail systems provide this functionality. A status code separate from 3xx is used since the semantics are different: for 300, it is assumed that the same person or service will be reached by the choices provided. While an automated choice or sequential search makes sense for a 3xx response, user intervention is required for a 485 (Ambiguous) response.

21.4.24 486 Busy Here

The callee's end system was contacted successfully, but the callee is currently not willing or able to take additional calls at this end system. The response MAY indicate a better time to call in the Retry-After header field. The user could also be available

elsewhere, such as through a voice mail service. Status 600 (Busy Everywhere) SHOULD be used if the client knows that no other end system will be able to accept this call.

21.4.25 487 Request Terminated

The request was terminated by a BYE or CANCEL request. This response is never returned for a CANCEL request itself.

21.4.26 488 Not Acceptable Here

The response has the same meaning as 606 (Not Acceptable), but only applies to the specific resource addressed by the Request-URI and the request may succeed elsewhere.

A message body containing a description of media capabilities MAY be present in the response, which is formatted according to the Accept header field in the INVITE (or application/sdp if not present), the same as a message body in a 200 (OK) response to an OPTIONS request.

21.4.27 491 Request Pending

The request was received by a UAS that had a pending request within the same dialog. Section 14.2 describes how such "glare" situations are resolved.

21.4.28 493 Undecipherable

The request was received by a UAS that contained an encrypted MIME body for which the recipient does not possess or will not provide an appropriate decryption key. This response MAY have a single body containing an appropriate public key that should be used to encrypt MIME bodies sent to this UA. Details of the usage of this response code can be found in Section 23.2.

21.5 Server Failure 5xx

5xx responses are failure responses given when a server itself has erred.

21.5.1 500 Server Internal Error

The server encountered an unexpected condition that prevented it from fulfilling the request. The client MAY display the specific error condition and MAY retry the request after several seconds.

If the condition is temporary, the server MAY indicate when the client may retry the request using the Retry-After header field.

21.5.2 501 Not Implemented

The server does not support the functionality required to fulfill the request. This is the appropriate response when a UAS does not recognize the request method and is not capable of supporting it for any user. (Proxies forward all requests regardless of method.)

Note that a 405 (Method Not Allowed) is sent when the server recognizes the request method, but that method is not allowed or supported.

21.5.3 502 Bad Gateway

The server, while acting as a gateway or proxy, received an invalid response from the downstream server it accessed in attempting to fulfill the request.

21.5.4 503 Service Unavailable

The server is temporarily unable to process the request due to a temporary overloading or maintenance of the server. The server MAY indicate when the client should retry the request in a Retry-After header field. If no Retry-After is given, the client MUST act as if it had received a 500 (Server Internal Error) response.

A client (proxy or UAC) receiving a 503 (Service Unavailable) SHOULD attempt to forward the request to an alternate server. It SHOULD NOT forward any other requests to that server for the duration specified in the Retry-After header field, if present.

Servers MAY refuse the connection or drop the request instead of responding with 503 (Service Unavailable).

21.5.5 504 Server Time-out

The server did not receive a timely response from an external server it accessed in attempting to process the request. 408 (Request Timeout) should be used instead if there was no response within the period specified in the Expires header field from the upstream server.

21.5.6 505 Version Not Supported

The server does not support, or refuses to support, the SIP protocol version that was used in the request. The server is indicating that it is unable or unwilling to complete the request using the same major version as the client, other than with this error message.

21.5.7 513 Message Too Large

The server was unable to process the request since the message length exceeded its capabilities.

21.6 Global Failures 6xx

6xx responses indicate that a server has definitive information about a particular user, not just the particular instance indicated in the Request-URI.

21.6.1 600 Busy Everywhere

The callee's end system was contacted successfully but the callee is busy and does not wish to take the call at this time. The response MAY indicate a better time to call in the Retry-After header field. If the callee does not wish to reveal the reason for declining the call, the callee uses status code 603 (Decline) instead. This status response is returned only if the client knows that no other end point (such as a voice mail system) will answer the request. Otherwise, 486 (Busy Here) should be returned.

21.6.2 603 Decline

The callee's machine was successfully contacted but the user explicitly does not wish to or cannot participate. The response MAY indicate a better time to call in the Retry-After header field. This status response is returned only if the client knows that no other end point will answer the request.

21.6.3 604 Does Not Exist Anywhere

The server has authoritative information that the user indicated in the Request-URI does not exist anywhere.

21.6.4 606 Not Acceptable

The user's agent was contacted successfully but some aspects of the session description such as the requested media, bandwidth, or addressing style were not acceptable.

A 606 (Not Acceptable) response means that the user wishes to communicate, but cannot adequately support the session described. The 606 (Not Acceptable) response MAY contain a list of reasons in a Warning header field describing why the session described cannot be supported. Warning reason codes are listed in Section 20.43.

A message body containing a description of media capabilities MAY be present in the response, which is formatted according to the Accept header field in the INVITE (or application/sdp if not present), the same as a message body in a 200 (OK) response to an OPTIONS request.

It is hoped that negotiation will not frequently be needed, and when a new user is being invited to join an already existing conference, negotiation may not be possible. It is up to the invitation initiator to decide whether or not to act on a 606 (Not Acceptable) response.

This status response is returned only if the client knows that no other end point will answer the request.

22 Usage of HTTP Authentication

SIP provides a stateless, challenge-based mechanism for authentication that is based on authentication in HTTP. Any time that a proxy server or UA receives a request (with the exceptions given in Section 22.1), it MAY challenge the initiator of the request to provide assurance of its identity. Once the originator has been identified, the recipient of the request SHOULD ascertain whether or not this user is authorized to make the request in question. No authorization systems are recommended or discussed in this document.

The "Digest" authentication mechanism described in this section provides message authentication and replay protection only, without message integrity or confidentiality. Protective measures above and beyond those provided by Digest need to be taken to prevent active attackers from modifying SIP requests and responses.

Note that due to its weak security, the usage of "Basic" authentication has been deprecated. Servers MUST NOT accept credentials using the "Basic" authorization scheme, and servers also MUST NOT challenge with "Basic". This is a change from RFC 2543.

22.1 Framework

The framework for SIP authentication closely parallels that of HTTP (RFC 2617 [17]). In particular, the BNF for auth-scheme, auth-param, challenge, realm, realm-value, and credentials is identical (although the usage of "Basic" as a scheme is not permitted). In SIP, a UAS uses the 401 (Unauthorized) response to challenge the identity of a UAC. Additionally, registrars and redirect servers MAY make use of 401 (Unauthorized) responses for authentication, but proxies MUST NOT, and instead MAY use the 407 (Proxy Authentication Required)

response. The requirements for inclusion of the Proxy-Authenticate, Proxy-Authorization, WWW-Authenticate, and Authorization in the various messages are identical to those described in RFC 2617 [17].

Since SIP does not have the concept of a canonical root URL, the notion of protection spaces is interpreted differently in SIP. The realm string alone defines the protection domain. This is a change from RFC 2543, in which the Request-URI and the realm together defined the protection domain.

> This previous definition of protection domain caused some amount of confusion since the Request-URI sent by the UAC and the Request-URI received by the challenging server might be different, and indeed the final form of the Request-URI might not be known to the UAC. Also, the previous definition depended on the presence of a SIP URI in the Request-URI and seemed to rule out alternative URI schemes (for example, the tel URL).

Operators of user agents or proxy servers that will authenticate received requests MUST adhere to the following guidelines for creation of a realm string for their server:

- Realm strings MUST be globally unique. It is RECOMMENDED that a realm string contain a hostname or domain name, following the recommendation in Section 3.2.1 of RFC 2617 [17].

- Realm strings SHOULD present a human-readable identifier that can be rendered to a user.

For example:

```
INVITE sip:bob@biloxi.com SIP/2.0
Authorization: Digest realm="biloxi.com", <...>
```

Generally, SIP authentication is meaningful for a specific realm, a protection domain. Thus, for Digest authentication, each such protection domain has its own set of usernames and passwords. If a server does not require authentication for a particular request, it MAY accept a default username, "anonymous", which has no password (password of ""). Similarly, UACs representing many users, such as PSTN gateways, MAY have their own device-specific username and password, rather than accounts for particular users, for their realm.

While a server can legitimately challenge most SIP requests, there are two requests defined by this document that require special handling for authentication: ACK and CANCEL.

Under an authentication scheme that uses responses to carry values used to compute nonces (such as Digest), some problems come up for any requests that take no response, including ACK. For this reason, any credentials in the INVITE that were accepted by a server MUST be accepted by that server for the ACK. UACs creating an ACK message will duplicate all of the Authorization and Proxy-Authorization header field values that appeared in the INVITE to which the ACK corresponds. Servers MUST NOT attempt to challenge an ACK.

Although the CANCEL method does take a response (a 2xx), servers MUST NOT attempt to challenge CANCEL requests since these requests cannot be resubmitted. Generally, a CANCEL request SHOULD be accepted by a server if it comes from the same hop that sent the request being canceled (provided that some sort of transport or network layer security association, as described in Section 26.2.1, is in place).

When a UAC receives a challenge, it SHOULD render to the user the contents of the "realm" parameter in the challenge (which appears in either a WWW-Authenticate header field or Proxy-Authenticate header field) if the UAC device does not already know of a credential for the realm in question. A service provider that pre-configures UAs with credentials for its realm should be aware that users will not have the opportunity to present their own credentials for this realm when challenged at a pre-configured device.

Finally, note that even if a UAC can locate credentials that are associated with the proper realm, the potential exists that these credentials may no longer be valid or that the challenging server will not accept these credentials for whatever reason (especially when "anonymous" with no password is submitted). In this instance a server may repeat its challenge, or it may respond with a 403 Forbidden. A UAC MUST NOT re-attempt requests with the credentials that have just been rejected (though the request may be retried if the nonce was stale).

22.2 User-to-User Authentication

When a UAS receives a request from a UAC, the UAS MAY authenticate the originator before the request is processed. If no credentials (in the Authorization header field) are provided in the request, the UAS can challenge the originator to provide credentials by rejecting the request with a 401 (Unauthorized) status code.

The WWW-Authenticate response-header field MUST be included in 401 (Unauthorized) response messages. The field value consists of at least one challenge that indicates the authentication scheme(s) and parameters applicable to the realm.

An example of the WWW-Authenticate header field in a 401 challenge is:

```
WWW-Authenticate: Digest
        realm="biloxi.com",
        qop="auth,auth-int",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

When the originating UAC receives the 401 (Unauthorized), it SHOULD, if it is able, re-originate the request with the proper credentials. The UAC may require input from the originating user before proceeding. Once authentication credentials have been supplied (either directly by the user, or discovered in an internal keyring), UAs SHOULD cache the credentials for a given value of the To header field and "realm" and attempt to re-use these values on the next request for that destination. UAs MAY cache credentials in any way they would like.

If no credentials for a realm can be located, UACs MAY attempt to retry the request with a username of "anonymous" and no password (a password of "").

Once credentials have been located, any UA that wishes to authenticate itself with a UAS or registrar -- usually, but not necessarily, after receiving a 401 (Unauthorized) response -- MAY do so by including an Authorization header field with the request. The Authorization field value consists of credentials containing the authentication information of the UA for the realm of the resource being requested as well as parameters required in support of authentication and replay protection.

An example of the Authorization header field is:

```
Authorization: Digest username="bob",
        realm="biloxi.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        uri="sip:bob@biloxi.com",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

When a UAC resubmits a request with its credentials after receiving a 401 (Unauthorized) or 407 (Proxy Authentication Required) response, it MUST increment the CSeq header field value as it would normally when sending an updated request.

22.3 Proxy-to-User Authentication

Similarly, when a UAC sends a request to a proxy server, the proxy server MAY authenticate the originator before the request is processed. If no credentials (in the Proxy-Authorization header field) are provided in the request, the proxy can challenge the originator to provide credentials by rejecting the request with a 407 (Proxy Authentication Required) status code. The proxy MUST populate the 407 (Proxy Authentication Required) message with a Proxy-Authenticate header field value applicable to the proxy for the requested resource.

The use of Proxy-Authenticate and Proxy-Authorization parallel that described in [17], with one difference. Proxies MUST NOT add values to the Proxy-Authorization header field. All 407 (Proxy Authentication Required) responses MUST be forwarded upstream toward the UAC following the procedures for any other response. It is the UAC's responsibility to add the Proxy-Authorization header field value containing credentials for the realm of the proxy that has asked for authentication.

> If a proxy were to resubmit a request adding a Proxy-Authorization header field value, it would need to increment the CSeq in the new request. However, this would cause the UAC that submitted the original request to discard a response from the UAS, as the CSeq value would be different.

When the originating UAC receives the 407 (Proxy Authentication Required) it SHOULD, if it is able, re-originate the request with the proper credentials. It should follow the same procedures for the display of the "realm" parameter that are given above for responding to 401.

If no credentials for a realm can be located, UACs MAY attempt to retry the request with a username of "anonymous" and no password (a password of "").

The UAC SHOULD also cache the credentials used in the re-originated request.

The following rule is RECOMMENDED for proxy credential caching:

If a UA receives a Proxy-Authenticate header field value in a 401/407 response to a request with a particular Call-ID, it should incorporate credentials for that realm in all subsequent requests that contain the same Call-ID. These credentials MUST NOT be cached across dialogs; however, if a UA is configured with the realm of its local outbound proxy, when one exists, then the UA MAY cache

credentials for that realm across dialogs. Note that this does mean a future request in a dialog could contain credentials that are not needed by any proxy along the Route header path.

Any UA that wishes to authenticate itself to a proxy server -- usually, but not necessarily, after receiving a 407 (Proxy Authentication Required) response -- MAY do so by including a Proxy-Authorization header field value with the request. The Proxy-Authorization request-header field allows the client to identify itself (or its user) to a proxy that requires authentication. The Proxy-Authorization header field value consists of credentials containing the authentication information of the UA for the proxy and/or realm of the resource being requested.

A Proxy-Authorization header field value applies only to the proxy whose realm is identified in the "realm" parameter (this proxy may previously have demanded authentication using the Proxy-Authenticate field). When multiple proxies are used in a chain, a Proxy-Authorization header field value MUST NOT be consumed by any proxy whose realm does not match the "realm" parameter specified in that value.

Note that if an authentication scheme that does not support realms is used in the Proxy-Authorization header field, a proxy server MUST attempt to parse all Proxy-Authorization header field values to determine whether one of them has what the proxy server considers to be valid credentials. Because this is potentially very time-consuming in large networks, proxy servers SHOULD use an authentication scheme that supports realms in the Proxy-Authorization header field.

If a request is forked (as described in Section 16.7), various proxy servers and/or UAs may wish to challenge the UAC. In this case, the forking proxy server is responsible for aggregating these challenges into a single response. Each WWW-Authenticate and Proxy-Authenticate value received in responses to the forked request MUST be placed into the single response that is sent by the forking proxy to the UA; the ordering of these header field values is not significant.

> When a proxy server issues a challenge in response to a request, it will not proxy the request until the UAC has retried the request with valid credentials. A forking proxy may forward a request simultaneously to multiple proxy servers that require authentication, each of which in turn will not forward the request until the originating UAC has authenticated itself in their respective realm. If the UAC does not provide credentials for

each challenge, the proxy servers that issued the challenges will not forward requests to the UA where the destination user might be located, and therefore, the virtues of forking are largely lost.

When resubmitting its request in response to a 401 (Unauthorized) or 407 (Proxy Authentication Required) that contains multiple challenges, a UAC MAY include an Authorization value for each WWW-Authenticate value and a Proxy-Authorization value for each Proxy-Authenticate value for which the UAC wishes to supply a credential. As noted above, multiple credentials in a request SHOULD be differentiated by the "realm" parameter.

It is possible for multiple challenges associated with the same realm to appear in the same 401 (Unauthorized) or 407 (Proxy Authentication Required). This can occur, for example, when multiple proxies within the same administrative domain, which use a common realm, are reached by a forking request. When it retries a request, a UAC MAY therefore supply multiple credentials in Authorization or Proxy-Authorization header fields with the same "realm" parameter value. The same credentials SHOULD be used for the same realm.

22.4 The Digest Authentication Scheme

This section describes the modifications and clarifications required to apply the HTTP Digest authentication scheme to SIP. The SIP scheme usage is almost completely identical to that for HTTP [17].

Since RFC 2543 is based on HTTP Digest as defined in RFC 2069 [39], SIP servers supporting RFC 2617 MUST ensure they are backwards compatible with RFC 2069. Procedures for this backwards compatibility are specified in RFC 2617. Note, however, that SIP servers MUST NOT accept or request Basic authentication.

The rules for Digest authentication follow those defined in [17], with "HTTP/1.1" replaced by "SIP/2.0" in addition to the following differences:

1. The URI included in the challenge has the following BNF:

   URI = SIP-URI / SIPS-URI

2. The BNF in RFC 2617 has an error in that the 'uri' parameter of the Authorization header field for HTTP Digest

authentication is not enclosed in quotation marks. (The example in Section 3.5 of RFC 2617 is correct.) For SIP, the 'uri' MUST be enclosed in quotation marks.

3. The BNF for digest-uri-value is:

   digest-uri-value = Request-URI ; as defined in Section 25

4. The example procedure for choosing a nonce based on Etag does not work for SIP.

5. The text in RFC 2617 [17] regarding cache operation does not apply to SIP.

6. RFC 2617 [17] requires that a server check that the URI in the request line and the URI included in the Authorization header field point to the same resource. In a SIP context, these two URIs may refer to different users, due to forwarding at some proxy. Therefore, in SIP, a server MAY check that the Request-URI in the Authorization header field value corresponds to a user for whom the server is willing to accept forwarded or direct requests, but it is not necessarily a failure if the two fields are not equivalent.

7. As a clarification to the calculation of the A2 value for message integrity assurance in the Digest authentication scheme, implementers should assume, when the entity-body is empty (that is, when SIP messages have no body) that the hash of the entity-body resolves to the MD5 hash of an empty string, or:

   H(entity-body) = MD5("") =
"d41d8cd98f00b204e9800998ecf8427e"

8. RFC 2617 notes that a cnonce value MUST NOT be sent in an Authorization (and by extension Proxy-Authorization) header field if no qop directive has been sent. Therefore, any algorithms that have a dependency on the cnonce (including "MD5-Sess") require that the qop directive be sent. Use of the "qop" parameter is optional in RFC 2617 for the purposes of backwards compatibility with RFC 2069; since RFC 2543 was based on RFC 2069, the "qop" parameter must unfortunately remain optional for clients and servers to receive. However, servers MUST always send a "qop" parameter in WWW-Authenticate and Proxy-Authenticate header field values. If a client receives a "qop" parameter in a challenge header field, it MUST send the "qop" parameter in any resulting authorization header field.

RFC 2543 did not allow usage of the Authentication-Info header field (it effectively used RFC 2069). However, we now allow usage of this header field, since it provides integrity checks over the bodies and provides mutual authentication. RFC 2617 [17] defines mechanisms for backwards compatibility using the qop attribute in the request. These mechanisms MUST be used by a server to determine if the client supports the new mechanisms in RFC 2617 that were not specified in RFC 2069.

23 S/MIME

SIP messages carry MIME bodies and the MIME standard includes mechanisms for securing MIME contents to ensure both integrity and confidentiality (including the 'multipart/signed' and 'application/pkcs7-mime' MIME types, see RFC 1847 [22], RFC 2630 [23] and RFC 2633 [24]). Implementers should note, however, that there may be rare network intermediaries (not typical proxy servers) that rely on viewing or modifying the bodies of SIP messages (especially SDP), and that secure MIME may prevent these sorts of intermediaries from functioning.

This applies particularly to certain types of firewalls.

The PGP mechanism for encrypting the header fields and bodies of SIP messages described in RFC 2543 has been deprecated.

23.1 S/MIME Certificates

The certificates that are used to identify an end-user for the purposes of S/MIME differ from those used by servers in one important respect - rather than asserting that the identity of the holder corresponds to a particular hostname, these certificates assert that the holder is identified by an end-user address. This address is composed of the concatenation of the "userinfo" "@" and "domainname" portions of a SIP or SIPS URI (in other words, an email address of the form "bob@biloxi.com"), most commonly corresponding to a user's address-of-record.

These certificates are also associated with keys that are used to sign or encrypt bodies of SIP messages. Bodies are signed with the private key of the sender (who may include their public key with the message as appropriate), but bodies are encrypted with the public key of the intended recipient. Obviously, senders must have foreknowledge of the public key of recipients in order to encrypt message bodies. Public keys can be stored within a UA on a virtual keyring.

Each user agent that supports S/MIME MUST contain a keyring specifically for end-users' certificates. This keyring should map between addresses of record and corresponding certificates. Over time, users SHOULD use the same certificate when they populate the originating URI of signaling (the From header field) with the same address-of-record.

Any mechanisms depending on the existence of end-user certificates are seriously limited in that there is virtually no consolidated authority today that provides certificates for end-user applications. However, users SHOULD acquire certificates from known public certificate authorities. As an alternative, users MAY create self-signed certificates. The implications of self-signed certificates are explored further in Section 26.4.2. Implementations may also use pre-configured certificates in deployments in which a previous trust relationship exists between all SIP entities.

Above and beyond the problem of acquiring an end-user certificate, there are few well-known centralized directories that distribute end-user certificates. However, the holder of a certificate SHOULD publish their certificate in any public directories as appropriate. Similarly, UACs SHOULD support a mechanism for importing (manually or automatically) certificates discovered in public directories corresponding to the target URIs of SIP requests.

23.2 S/MIME Key Exchange

SIP itself can also be used as a means to distribute public keys in the following manner.

Whenever the CMS SignedData message is used in S/MIME for SIP, it MUST contain the certificate bearing the public key necessary to verify the signature.

When a UAC sends a request containing an S/MIME body that initiates a dialog, or sends a non-INVITE request outside the context of a dialog, the UAC SHOULD structure the body as an S/MIME 'multipart/signed' CMS SignedData body. If the desired CMS service is EnvelopedData (and the public key of the target user is known), the UAC SHOULD send the EnvelopedData message encapsulated within a SignedData message.

When a UAS receives a request containing an S/MIME CMS body that includes a certificate, the UAS SHOULD first validate the certificate, if possible, with any available root certificates for certificate authorities. The UAS SHOULD also determine the subject of the certificate (for S/MIME, the SubjectAltName will contain the appropriate identity) and compare this value to the From header field

of the request. If the certificate cannot be verified, because it is self-signed, or signed by no known authority, or if it is verifiable but its subject does not correspond to the From header field of request, the UAS MUST notify its user of the status of the certificate (including the subject of the certificate, its signer, and any key fingerprint information) and request explicit permission before proceeding. If the certificate was successfully verified and the subject of the certificate corresponds to the From header field of the SIP request, or if the user (after notification) explicitly authorizes the use of the certificate, the UAS SHOULD add this certificate to a local keyring, indexed by the address-of-record of the holder of the certificate.

When a UAS sends a response containing an S/MIME body that answers the first request in a dialog, or a response to a non-INVITE request outside the context of a dialog, the UAS SHOULD structure the body as an S/MIME 'multipart/signed' CMS SignedData body. If the desired CMS service is EnvelopedData, the UAS SHOULD send the EnvelopedData message encapsulated within a SignedData message.

When a UAC receives a response containing an S/MIME CMS body that includes a certificate, the UAC SHOULD first validate the certificate, if possible, with any appropriate root certificate. The UAC SHOULD also determine the subject of the certificate and compare this value to the To field of the response; although the two may very well be different, and this is not necessarily indicative of a security breach. If the certificate cannot be verified because it is self-signed, or signed by no known authority, the UAC MUST notify its user of the status of the certificate (including the subject of the certificate, its signator, and any key fingerprint information) and request explicit permission before proceeding. If the certificate was successfully verified, and the subject of the certificate corresponds to the To header field in the response, or if the user (after notification) explicitly authorizes the use of the certificate, the UAC SHOULD add this certificate to a local keyring, indexed by the address-of-record of the holder of the certificate. If the UAC had not transmitted its own certificate to the UAS in any previous transaction, it SHOULD use a CMS SignedData body for its next request or response.

On future occasions, when the UA receives requests or responses that contain a From header field corresponding to a value in its keyring, the UA SHOULD compare the certificate offered in these messages with the existing certificate in its keyring. If there is a discrepancy, the UA MUST notify its user of a change of the certificate (preferably in terms that indicate that this is a potential security breach) and acquire the user's permission before continuing to

process the signaling. If the user authorizes this certificate, it SHOULD be added to the keyring alongside any previous value(s) for this address-of-record.

Note well however, that this key exchange mechanism does not guarantee the secure exchange of keys when self-signed certificates, or certificates signed by an obscure authority, are used - it is vulnerable to well-known attacks. In the opinion of the authors, however, the security it provides is proverbially better than nothing; it is in fact comparable to the widely used SSH application. These limitations are explored in greater detail in Section 26.4.2.

If a UA receives an S/MIME body that has been encrypted with a public key unknown to the recipient, it MUST reject the request with a 493 (Undecipherable) response. This response SHOULD contain a valid certificate for the respondent (corresponding, if possible, to any address of record given in the To header field of the rejected request) within a MIME body with a 'certs-only' "smime-type" parameter.

A 493 (Undecipherable) sent without any certificate indicates that the respondent cannot or will not utilize S/MIME encrypted messages, though they may still support S/MIME signatures.

Note that a user agent that receives a request containing an S/MIME body that is not optional (with a Content-Disposition header "handling" parameter of "required") MUST reject the request with a 415 Unsupported Media Type response if the MIME type is not understood. A user agent that receives such a response when S/MIME is sent SHOULD notify its user that the remote device does not support S/MIME, and it MAY subsequently resend the request without S/MIME, if appropriate; however, this 415 response may constitute a downgrade attack.

If a user agent sends an S/MIME body in a request, but receives a response that contains a MIME body that is not secured, the UAC SHOULD notify its user that the session could not be secured. However, if a user agent that supports S/MIME receives a request with an unsecured body, it SHOULD NOT respond with a secured body, but if it expects S/MIME from the sender (for example, because the sender's From header field value corresponds to an identity on its keychain), the UAS SHOULD notify its user that the session could not be secured.

A number of conditions that arise in the previous text call for the notification of the user when an anomalous certificate-management event occurs. Users might well ask what they should do under these circumstances. First and foremost, an unexpected change in a certificate, or an absence of security when security is expected, are

causes for caution but not necessarily indications that an attack is in progress. Users might abort any connection attempt or refuse a connection request they have received; in telephony parlance, they could hang up and call back. Users may wish to find an alternate means to contact the other party and confirm that their key has legitimately changed. Note that users are sometimes compelled to change their certificates, for example when they suspect that the secrecy of their private key has been compromised. When their private key is no longer private, users must legitimately generate a new key and re-establish trust with any users that held their old key.

Finally, if during the course of a dialog a UA receives a certificate in a CMS SignedData message that does not correspond with the certificates previously exchanged during a dialog, the UA MUST notify its user of the change, preferably in terms that indicate that this is a potential security breach.

23.3 Securing MIME bodies

There are two types of secure MIME bodies that are of interest to SIP: use of these bodies should follow the S/MIME specification [24] with a few variations.

- "multipart/signed" MUST be used only with CMS detached signatures.

  This allows backwards compatibility with non-S/MIME-compliant recipients.

- S/MIME bodies SHOULD have a Content-Disposition header field, and the value of the "handling" parameter SHOULD be "required."

- If a UAC has no certificate on its keyring associated with the address-of-record to which it wants to send a request, it cannot send an encrypted "application/pkcs7-mime" MIME message. UACs MAY send an initial request such as an OPTIONS message with a CMS detached signature in order to solicit the certificate of the remote side (the signature SHOULD be over a "message/sip" body of the type described in Section 23.4).

  Note that future standardization work on S/MIME may define non-certificate based keys.

- Senders of S/MIME bodies SHOULD use the "SMIMECapabilities" (see Section 2.5.2 of [24]) attribute to express their capabilities and preferences for further communications. Note especially that senders MAY use the "preferSignedData"

capability to encourage receivers to respond with CMS SignedData messages (for example, when sending an OPTIONS request as described above).

- o S/MIME implementations MUST at a minimum support SHA1 as a digital signature algorithm, and 3DES as an encryption algorithm. All other signature and encryption algorithms MAY be supported. Implementations can negotiate support for these algorithms with the "SMIMECapabilities" attribute.

- o Each S/MIME body in a SIP message SHOULD be signed with only one certificate. If a UA receives a message with multiple signatures, the outermost signature should be treated as the single certificate for this body. Parallel signatures SHOULD NOT be used.

The following is an example of an encrypted S/MIME SDP body within a SIP message:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Disposition: attachment; filename=smime.p7m
   handling=required

**************************************************
* Content-Type: application/sdp                  *
*                                                *
* v=0                                            *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com *
* s=-                                            *
* t=0 0                                          *
* c=IN IP4 pc33.atlanta.com                      *
* m=audio 3456 RTP/AVP 0 1 3 99                  *
* a=rtpmap:0 PCMU/8000                           *
**************************************************
```

23.4 SIP Header Privacy and Integrity using S/MIME: Tunneling SIP

As a means of providing some degree of end-to-end authentication, integrity or confidentiality for SIP header fields, S/MIME can encapsulate entire SIP messages within MIME bodies of type "message/sip" and then apply MIME security to these bodies in the same manner as typical SIP bodies. These encapsulated SIP requests and responses do not constitute a separate dialog or transaction, they are a copy of the "outer" message that is used to verify integrity or to supply additional information.

If a UAS receives a request that contains a tunneled "message/sip" S/MIME body, it SHOULD include a tunneled "message/sip" body in the response with the same smime-type.

Any traditional MIME bodies (such as SDP) SHOULD be attached to the "inner" message so that they can also benefit from S/MIME security. Note that "message/sip" bodies can be sent as a part of a MIME "multipart/mixed" body if any unsecured MIME types should also be transmitted in a request.

23.4.1 Integrity and Confidentiality Properties of SIP Headers

When the S/MIME integrity or confidentiality mechanisms are used, there may be discrepancies between the values in the "inner" message and values in the "outer" message. The rules for handling any such differences for all of the header fields described in this document are given in this section.

Note that for the purposes of loose timestamping, all SIP messages that tunnel "message/sip" SHOULD contain a Date header in both the "inner" and "outer" headers.

23.4.1.1 Integrity

Whenever integrity checks are performed, the integrity of a header field should be determined by matching the value of the header field in the signed body with that in the "outer" messages using the comparison rules of SIP as described in 20.

Header fields that can be legitimately modified by proxy servers are: Request-URI, Via, Record-Route, Route, Max-Forwards, and Proxy-Authorization. If these header fields are not intact end-to-end, implementations SHOULD NOT consider this a breach of security. Changes to any other header fields defined in this document constitute an integrity violation; users MUST be notified of a discrepancy.

23.4.1.2 Confidentiality

When messages are encrypted, header fields may be included in the encrypted body that are not present in the "outer" message.

Some header fields must always have a plaintext version because they are required header fields in requests and responses - these include:

To, From, Call-ID, CSeq, Contact. While it is probably not useful to provide an encrypted alternative for the Call-ID, CSeq, or Contact, providing an alternative to the information in the "outer" To or From is permitted. Note that the values in an encrypted body are not used for the purposes of identifying transactions or dialogs - they are merely informational. If the From header field in an encrypted body differs from the value in the "outer" message, the value within the encrypted body SHOULD be displayed to the user, but MUST NOT be used in the "outer" header fields of any future messages.

Primarily, a user agent will want to encrypt header fields that have an end-to-end semantic, including: Subject, Reply-To, Organization, Accept, Accept-Encoding, Accept-Language, Alert-Info, Error-Info, Authentication-Info, Expires, In-Reply-To, Require, Supported, Unsupported, Retry-After, User-Agent, Server, and Warning. If any of these header fields are present in an encrypted body, they should be used instead of any "outer" header fields, whether this entails displaying the header field values to users or setting internal states in the UA. They SHOULD NOT however be used in the "outer" headers of any future messages.

If present, the Date header field MUST always be the same in the "inner" and "outer" headers.

Since MIME bodies are attached to the "inner" message, implementations will usually encrypt MIME-specific header fields, including: MIME-Version, Content-Type, Content-Length, Content-Language, Content-Encoding and Content-Disposition. The "outer" message will have the proper MIME header fields for S/MIME bodies. These header fields (and any MIME bodies they preface) should be treated as normal MIME header fields and bodies received in a SIP message.

It is not particularly useful to encrypt the following header fields: Min-Expires, Timestamp, Authorization, Priority, and WWW-Authenticate. This category also includes those header fields that can be changed by proxy servers (described in the preceding section). UAs SHOULD never include these in an "inner" message if they are not

included in the "outer" message. UAs that receive any of these header fields in an encrypted body SHOULD ignore the encrypted values.

Note that extensions to SIP may define additional header fields; the authors of these extensions should describe the integrity and confidentiality properties of such header fields. If a SIP UA encounters an unknown header field with an integrity violation, it MUST ignore the header field.

23.4.2 Tunneling Integrity and Authentication

Tunneling SIP messages within S/MIME bodies can provide integrity for SIP header fields if the header fields that the sender wishes to secure are replicated in a "message/sip" MIME body signed with a CMS detached signature.

Provided that the "message/sip" body contains at least the fundamental dialog identifiers (To, From, Call-ID, CSeq), then a signed MIME body can provide limited authentication. At the very least, if the certificate used to sign the body is unknown to the recipient and cannot be verified, the signature can be used to ascertain that a later request in a dialog was transmitted by the same certificate-holder that initiated the dialog. If the recipient of the signed MIME body has some stronger incentive to trust the certificate (they were able to validate it, they acquired it from a trusted repository, or they have used it frequently) then the signature can be taken as a stronger assertion of the identity of the subject of the certificate.

In order to eliminate possible confusions about the addition or subtraction of entire header fields, senders SHOULD replicate all header fields from the request within the signed body. Any message bodies that require integrity protection MUST be attached to the "inner" message.

If a Date header is present in a message with a signed body, the recipient SHOULD compare the header field value with its own internal clock, if applicable. If a significant time discrepancy is detected (on the order of an hour or more), the user agent SHOULD alert the user to the anomaly, and note that it is a potential security breach.

If an integrity violation in a message is detected by its recipient, the message MAY be rejected with a 403 (Forbidden) response if it is a request, or any existing dialog MAY be terminated. UAs SHOULD notify users of this circumstance and request explicit guidance on how to proceed.

The following is an example of the use of a tunneled "message/sip" body:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
Content-Length: 568

--boundary42
Content-Type: message/sip

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <bob@biloxi.com>
From: Alice <alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 pc33.atlanta.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required
```

```
ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42-
```

23.4.3 Tunneling Encryption

It may also be desirable to use this mechanism to encrypt a "message/sip" MIME body within a CMS EnvelopedData message S/MIME body, but in practice, most header fields are of at least some use to the network; the general use of encryption with S/MIME is to secure message bodies like SDP rather than message headers. Some informational header fields, such as the Subject or Organization could perhaps warrant end-to-end security. Headers defined by future SIP applications might also require obfuscation.

Another possible application of encrypting header fields is selective anonymity. A request could be constructed with a From header field that contains no personal information (for example, sip:anonymous@anonymizer.invalid). However, a second From header field containing the genuine address-of-record of the originator could be encrypted within a "message/sip" MIME body where it will only be visible to the endpoints of a dialog.

> Note that if this mechanism is used for anonymity, the From header field will no longer be usable by the recipient of a message as an index to their certificate keychain for retrieving the proper S/MIME key to associated with the sender. The message must first be decrypted, and the "inner" From header field MUST be used as an index.

In order to provide end-to-end integrity, encrypted "message/sip" MIME bodies SHOULD be signed by the sender. This creates a "multipart/signed" MIME body that contains an encrypted body and a signature, both of type "application/pkcs7-mime".

In the following example, of an encrypted and signed message, the text boxed in asterisks ("*") is encrypted:

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Anonymous <sip:anonymous@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:pc33.atlanta.com>
Content-Type: multipart/signed;
  protocol="application/pkcs7-signature";
  micalg=sha1; boundary=boundary42
Content-Length: 568

--boundary42
Content-Type: application/pkcs7-mime; smime-type=enveloped-data;
     name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
   handling=required
Content-Length: 231

**********************************************************
* Content-Type: message/sip                              *
*                                                        *
* INVITE sip:bob@biloxi.com SIP/2.0                      *
* Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8 *
* To: Bob <bob@biloxi.com>                               *
* From: Alice <alice@atlanta.com>;tag=1928301774         *
* Call-ID: a84b4c76e66710                                *
* CSeq: 314159 INVITE                                    *
* Max-Forwards: 70                                       *
* Date: Thu, 21 Feb 2002 13:02:03 GMT                    *
* Contact: <sip:alice@pc33.atlanta.com>                  *
*                                                        *
* Content-Type: application/sdp                          *
*                                                        *
* v=0                                                    *
* o=alice 53655765 2353687637 IN IP4 pc33.atlanta.com    *
* s=Session SDP                                          *
* t=0 0                                                  *
* c=IN IP4 pc33.atlanta.com                              *
* m=audio 3456 RTP/AVP 0 1 3 99                          *
* a=rtpmap:0 PCMU/8000                                   *
**********************************************************
```

```
--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s;
   handling=required

ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
7GhIGfHfYT64VQbnj756

--boundary42-
```

24 Examples

In the following examples, we often omit the message body and the corresponding Content-Length and Content-Type header fields for brevity.

24.1 Registration

Bob registers on start-up. The message flow is shown in Figure 9. Note that the authentication usually required for registration is not shown for simplicity.

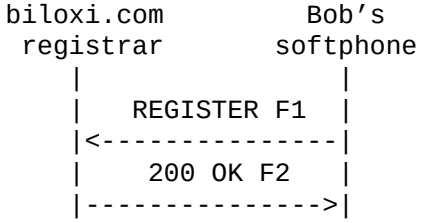


Figure 9: SIP Registration Example

F1 REGISTER Bob -> Registrar

```
REGISTER sip:registrar.biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 0
```

The registration expires after two hours. The registrar responds with a 200 OK:

F2 200 OK Registrar -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.4
To: Bob <sip:bob@biloxi.com>;tag=2493k59kd
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 0
```

24.2 Session Setup

This example contains the full details of the example session setup in Section 4. The message flow is shown in Figure 1. Note that these flows show the minimum required set of header fields - some other header fields such as Allow and Supported would normally be present.

F1 INVITE Alice -> atlanta.com proxy

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice’s SDP not shown)
```

```
F2 100 Trying atlanta.com proxy -> Alice

SIP/2.0 100 Trying
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Content-Length: 0

F3 INVITE atlanta.com proxy -> biloxi.com proxy

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 69
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)

F4 100 Trying biloxi.com proxy -> atlanta.com proxy

SIP/2.0 100 Trying
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Content-Length: 0
```

```
F5 INVITE biloxi.com proxy -> Bob

INVITE sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
Max-Forwards: 68
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

(Alice's SDP not shown)

F6 180 Ringing Bob -> biloxi.com proxy

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
 ;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0

F7 180 Ringing biloxi.com proxy -> atlanta.com proxy

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0
```

```
F8 180 Ringing atlanta.com proxy -> Alice

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
Contact: <sip:bob@192.0.2.4>
CSeq: 314159 INVITE
Content-Length: 0

F9 200 OK Bob -> biloxi.com proxy

SIP/2.0 200 OK
Via: SIP/2.0/UDP server10.biloxi.com;branch=z9hG4bK4b43c2ff8.1
 ;received=192.0.2.3
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)

F10 200 OK biloxi.com proxy -> atlanta.com proxy

SIP/2.0 200 OK
Via: SIP/2.0/UDP bigbox3.site3.atlanta.com;branch=z9hG4bK77ef4c2312983.1
 ;received=192.0.2.2
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)
```

```
F11 200 OK atlanta.com proxy -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
 ;received=192.0.2.1
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:bob@192.0.2.4>
Content-Type: application/sdp
Content-Length: 131

(Bob's SDP not shown)

F12 ACK Alice -> Bob

ACK sip:bob@192.0.2.4 SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds9
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>;tag=a6c85cf
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 ACK
Content-Length: 0
```

The media session between Alice and Bob is now established.

Bob hangs up first. Note that Bob's SIP phone maintains its own CSeq numbering space, which, in this example, begins with 231. Since Bob is making the request, the To and From URIs and tags have been swapped.

```
F13 BYE Bob -> Alice

BYE sip:alice@pc33.atlanta.com SIP/2.0
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
Max-Forwards: 70
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0
```

F14 200 OK Alice -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.4;branch=z9hG4bKnashds10
From: Bob <sip:bob@biloxi.com>;tag=a6c85cf
To: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 231 BYE
Content-Length: 0
```

The SIP Call Flows document [40] contains further examples of SIP messages.

25 Augmented BNF for the SIP Protocol

All of the mechanisms specified in this document are described in both prose and an augmented Backus-Naur Form (BNF) defined in RFC 2234 [10]. Section 6.1 of RFC 2234 defines a set of core rules that are used by this specification, and not repeated here. Implementers need to be familiar with the notation and content of RFC 2234 in order to understand this specification. Certain basic rules are in uppercase, such as SP, LWS, HTAB, CRLF, DIGIT, ALPHA, etc. Angle brackets are used within definitions to clarify the use of rule names.

The use of square brackets is redundant syntactically. It is used as a semantic hint that the specific parameter is optional to use.

25.1 Basic Rules

The following rules are used throughout this specification to describe basic parsing constructs. The US-ASCII coded character set is defined by ANSI X3.4-1986.

```
alphanum  =  ALPHA / DIGIT
```

Several rules are incorporated from RFC 2396 [5] but are updated to make them compliant with RFC 2234 [10]. These include:

```
reserved    =  ";" / "/" / "?" / ":" / "@" / "&" / "=" / "+"
               / "$" / ","
unreserved  =  alphanum / mark
mark        =  "-" / "_" / "." / "!" / "~" / "*" / "'"
               / "(" / ")"
escaped     =  "%" HEXDIG HEXDIG
```

SIP header field values can be folded onto multiple lines if the continuation line begins with a space or horizontal tab. All linear white space, including folding, has the same semantics as SP. A recipient MAY replace any linear white space with a single SP before interpreting the field value or forwarding the message downstream. This is intended to behave exactly as HTTP/1.1 as described in RFC 2616 [8]. The SWS construct is used when linear white space is optional, generally between tokens and separators.

```
LWS  =  [*WSP CRLF] 1*WSP ; linear whitespace
SWS  =  [LWS] ; sep whitespace
```

To separate the header name from the rest of value, a colon is used, which, by the above rule, allows whitespace before, but no line break, and whitespace after, including a linebreak. The HCOLON defines this construct.

```
HCOLON  =  *( SP / HTAB ) ":" SWS
```

The TEXT-UTF8 rule is only used for descriptive field contents and values that are not intended to be interpreted by the message parser. Words of *TEXT-UTF8 contain characters from the UTF-8 charset (RFC 2279 [7]). The TEXT-UTF8-TRIM rule is used for descriptive field contents that are n t quoted strings, where leading and trailing LWS is not meaningful. In this regard, SIP differs from HTTP, which uses the ISO 8859-1 character set.

```
TEXT-UTF8-TRIM  =  1*TEXT-UTF8char *(*LWS TEXT-UTF8char)
TEXT-UTF8char   =  %x21-7E / UTF8-NONASCII
UTF8-NONASCII   =  %xC0-DF 1UTF8-CONT
                /  %xE0-EF 2UTF8-CONT
                /  %xF0-F7 3UTF8-CONT
                /  %xF8-Fb 4UTF8-CONT
                /  %xFC-FD 5UTF8-CONT
UTF8-CONT       =  %x80-BF
```

A CRLF is allowed in the definition of TEXT-UTF8-TRIM only as part of a header field continuation. It is expected that the folding LWS will be replaced with a single SP before interpretation of the TEXT-UTF8-TRIM value.

Hexadecimal numeric characters are used in several protocol elements. Some elements (authentication) force hex alphas to be lower case.

```
LHEX  =  DIGIT / %x61-66 ;lowercase a-f
```

Many SIP header field values consist of words separated by LWS or special characters. Unless otherwise stated, tokens are case-insensitive. These special characters MUST be in a quoted string to be used within a parameter value. The word construct is used in Call-ID to allow most separators to be used.

```
token       =  1*(alphanum / "-" / "." / "!" / "%" / "*"
               / "_" / "+" / "`" / "'" / "~" )
separators  =  "(" / ")" / "<" / ">" / "@" /
               "," / ";" / ":" / "\" / DQUOTE /
               "/" / "[" / "]" / "?" / "=" /
               "{" / "}" / SP / HTAB
word        =  1*(alphanum / "-" / "." / "!" / "%" / "*" /
               "_" / "+" / "`" / "'" / "~" /
               "(" / ")" / "<" / ">" /
               ":" / "\" / DQUOTE /
               "/" / "[" / "]" / "?" /
               "{" / "}" )
```

When tokens are used or separators are used between elements, whitespace is often allowed before or after these characters:

```
STAR    =  SWS "*" SWS ; asterisk
SLASH   =  SWS "/" SWS ; slash
EQUAL   =  SWS "=" SWS ; equal
LPAREN  =  SWS "(" SWS ; left parenthesis
RPAREN  =  SWS ")" SWS ; right parenthesis
RAQUOT  =  ">" SWS ; right angle quote
LAQUOT  =  SWS "<"; left angle quote
COMMA   =  SWS "," SWS ; comma
SEMI    =  SWS ";" SWS ; semicolon
COLON   =  SWS ":" SWS ; colon
LDQUOT  =  SWS DQUOTE; open double quotation mark
RDQUOT  =  DQUOTE SWS ; close double quotation mark
```

Comments can be included in some SIP header fields by surrounding the comment text with parentheses. Comments are only allowed in fields containing "comment" as part of their field value definition. In all other fields, parentheses are considered part of the field value.

```
comment  =  LPAREN *(ctext / quoted-pair / comment) RPAREN
ctext    =  %x21-27 / %x2A-5B / %x5D-7E / UTF8-NONASCII
            / LWS
```

ctext includes all chars except left and right parens and backslash. A string of text is parsed as a single word if it is quoted using double-quote marks. In quoted strings, quotation marks (") and backslashes (\) need to be escaped.

```
quoted-string  =  SWS DQUOTE *(qdtext / quoted-pair ) DQUOTE
qdtext         =  LWS / %x21 / %x23-5B / %x5D-7E
                  / UTF8-NONASCII
```

The backslash character ("\") MAY be used as a single-character quoting mechanism only within quoted-string and comment constructs. Unlike HTTP/1.1, the characters CR and LF cannot be escaped by this mechanism to avoid conflict with line folding and header separation.

```
quoted-pair  =  "\" (%x00-09 / %x0B-0C
                / %x0E-7F)

SIP-URI          =  "sip:" [ userinfo ] hostport
                    uri-parameters [ headers ]
SIPS-URI         =  "sips:" [ userinfo ] hostport
                    uri-parameters [ headers ]
userinfo         =  ( user / telephone-subscriber ) [ ":" password ] "@"
user             =  1*( unreserved / escaped / user-unreserved )
user-unreserved  =  "&" / "=" / "+" / "$" / "," / ";" / "?" / "/"
password         =  *( unreserved / escaped /
                    "&" / "=" / "+" / "$" / "," )
hostport         =  host [ ":" port ]
host             =  hostname / IPv4address / IPv6reference
hostname         =  *( domainlabel "." ) toplabel [ "." ]
domainlabel      =  alphanum
                    / alphanum *( alphanum / "-" ) alphanum
toplabel         =  ALPHA / ALPHA *( alphanum / "-" ) alphanum
```

```
IPv4address    =  1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT "." 1*3DIGIT
IPv6reference  =  "[" IPv6address "]"
IPv6address    =  hexpart [ ":" IPv4address ]
hexpart        =  hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
hexseq         =  hex4 *( ":" hex4)
hex4           =  1*4HEXDIG
port           =  1*DIGIT
```

The BNF for telephone-subscriber can be found in RFC 2806 [9]. Note, however, that any characters allowed there that are not allowed in the user part of the SIP URI MUST be escaped.

```
uri-parameters    =  *( ";" uri-parameter)
uri-parameter     =  transport-param / user-param / method-param
                     / ttl-param / maddr-param / lr-param / other-param
transport-param   =  "transport="
                     ( "udp" / "tcp" / "sctp" / "tls"
                     / other-transport)
other-transport   =  token
user-param        =  "user=" ( "phone" / "ip" / other-user)
other-user        =  token
method-param      =  "method=" Method
ttl-param         =  "ttl=" ttl
maddr-param       =  "maddr=" host
lr-param          =  "lr"
other-param       =  pname [ "=" pvalue ]
pname             =  1*paramchar
pvalue            =  1*paramchar
paramchar         =  param-unreserved / unreserved / escaped
param-unreserved  =  "[" / "]" / "/" / ":" / "&" / "+" / "$"

headers         =  "?" header *( "&" header )
header          =  hname "=" hvalue
hname           =  1*( hnv-unreserved / unreserved / escaped )
hvalue          =  *( hnv-unreserved / unreserved / escaped )
hnv-unreserved  =  "[" / "]" / "/" / "?" / ":" / "+" / "$"

SIP-message    =  Request / Response
Request        =  Request-Line
                  *( message-header )
                  CRLF
                  [ message-body ]
Request-Line   =  Method SP Request-URI SP SIP-Version CRLF
Request-URI    =  SIP-URI / SIPS-URI / absoluteURI
absoluteURI    =  scheme ":" ( hier-part / opaque-part )
hier-part      =  ( net-path / abs-path ) [ "?" query ]
net-path       =  "//" authority [ abs-path ]
abs-path       =  "/" path-segments
```

```
opaque-part    =  uric-no-slash *uric
uric           =  reserved / unreserved / escaped
uric-no-slash  =  unreserved / escaped / ";" / "?" / ":" / "@"
                  / "&" / "=" / "+" / "$" / ","
path-segments  =  segment *( "/" segment )
segment        =  *pchar *( ";" param )
param          =  *pchar
pchar          =  unreserved / escaped /
                  ":" / "@" / "&" / "=" / "+" / "$" / ","
scheme         =  ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
authority      =  srvr / reg-name
srvr           =  [ [ userinfo "@" ] hostport ]
reg-name       =  1*( unreserved / escaped / "$" / ","
                  / ";" / ":" / "@" / "&" / "=" / "+" )
query          =  *uric
SIP-Version    =  "SIP" "/" 1*DIGIT "." 1*DIGIT

message-header  =  (Accept
                /  Accept-Encoding
                /  Accept-Language
                /  Alert-Info
                /  Allow
                /  Authentication-Info
                /  Authorization
                /  Call-ID
                /  Call-Info
                /  Contact
                /  Content-Disposition
                /  Content-Encoding
                /  Content-Language
                /  Content-Length
                /  Content-Type
                /  CSeq
                /  Date
                /  Error-Info
                /  Expires
                /  From
                /  In-Reply-To
                /  Max-Forwards
                /  MIME-Version
                /  Min-Expires
                /  Organization
                /  Priority
                /  Proxy-Authenticate
                /  Proxy-Authorization
                /  Proxy-Require
                /  Record-Route
                /  Reply-To
```

```
                / Require
                / Retry-After
                / Route
                / Server
                / Subject
                / Supported
                / Timestamp
                / To
                / Unsupported
                / User-Agent
                / Via
                / Warning
                / WWW-Authenticate
                / extension-header) CRLF

INVITEm           =  %x49.4E.56.49.54.45 ; INVITE in caps
ACKm              =  %x41.43.4B ; ACK in caps
OPTIONSm          =  %x4F.50.54.49.4F.4E.53 ; OPTIONS in caps
BYEm              =  %x42.59.45 ; BYE in caps
CANCELm           =  %x43.41.4E.43.45.4C ; CANCEL in caps
REGISTERm         =  %x52.45.47.49.53.54.45.52 ; REGISTER in caps
Method            =  INVITEm / ACKm / OPTIONSm / BYEm
                     / CANCELm / REGISTERm
                     / extension-method
extension-method  =  token
Response          =  Status-Line
                     *( message-header )
                     CRLF
                     [ message-body ]

Status-Line     =  SIP-Version SP Status-Code SP Reason-Phrase CRLF
Status-Code     =  Informational
                /  Redirection
                /  Success
                /  Client-Error
                /  Server-Error
                /  Global-Failure
                /  extension-code
extension-code  =  3DIGIT
Reason-Phrase   =  *(reserved / unreserved / escaped
                   / UTF8-NONASCII / UTF8-CONT / SP / HTAB)

Informational  =  "100"  ;  Trying
               /  "180"  ;  Ringing
               /  "181"  ;  Call Is Being Forwarded
               /  "182"  ;  Queued
               /  "183"  ;  Session Progress
```

```
Success  =  "200"  ;  OK

Redirection  =  "300"  ;  Multiple Choices
             /   "301"  ;  Moved Permanently
             /   "302"  ;  Moved Temporarily
             /   "305"  ;  Use Proxy
             /   "380"  ;  Alternative Service

Client-Error  =  "400"  ;  Bad Request
              /   "401"  ;  Unauthorized
              /   "402"  ;  Payment Required
              /   "403"  ;  Forbidden
              /   "404"  ;  Not Found
              /   "405"  ;  Method Not Allowed
              /   "406"  ;  Not Acceptable
              /   "407"  ;  Proxy Authentication Required
              /   "408"  ;  Request Timeout
              /   "410"  ;  Gone
              /   "413"  ;  Request Entity Too Large
              /   "414"  ;  Request-URI Too Large
              /   "415"  ;  Unsupported Media Type
              /   "416"  ;  Unsupported URI Scheme
              /   "420"  ;  Bad Extension
              /   "421"  ;  Extension Required
              /   "423"  ;  Interval Too Brief
              /   "480"  ;  Temporarily not available
              /   "481"  ;  Call Leg/Transaction Does Not Exist
              /   "482"  ;  Loop Detected
              /   "483"  ;  Too Many Hops
              /   "484"  ;  Address Incomplete
              /   "485"  ;  Ambiguous
              /   "486"  ;  Busy Here
              /   "487"  ;  Request Terminated
              /   "488"  ;  Not Acceptable Here
              /   "491"  ;  Request Pending
              /   "493"  ;  Undecipherable

Server-Error  =  "500"  ;  Internal Server Error
              /   "501"  ;  Not Implemented
              /   "502"  ;  Bad Gateway
              /   "503"  ;  Service Unavailable
              /   "504"  ;  Server Time-out
              /   "505"  ;  SIP Version not supported
              /   "513"  ;  Message Too Large
```

```
Global-Failure  =  "600"  ;  Busy Everywhere
               /   "603"  ;  Decline
               /   "604"  ;  Does not exist anywhere
               /   "606"  ;  Not Acceptable

Accept         =  "Accept" HCOLON
                   [ accept-range *(COMMA accept-range) ]
accept-range   =  media-range *(SEMI accept-param)
media-range    =  ( "*/*"
                  / ( m-type SLASH "*" )
                  / ( m-type SLASH m-subtype )
                  ) *( SEMI m-parameter )
accept-param   =  ("q" EQUAL qvalue) / generic-param
qvalue         =  ( "0" [ "." 0*3DIGIT ] )
                  / ( "1" [ "." 0*3("0") ] )
generic-param  =  token [ EQUAL gen-value ]
gen-value      =  token / host / quoted-string

Accept-Encoding  =  "Accept-Encoding" HCOLON
                    [ encoding *(COMMA encoding) ]
encoding         =  codings *(SEMI accept-param)
codings          =  content-coding / "*"
content-coding   =  token

Accept-Language  =  "Accept-Language" HCOLON
                    [ language *(COMMA language) ]
language         =  language-range *(SEMI accept-param)
language-range   =  ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) / "*" )

Alert-Info   =  "Alert-Info" HCOLON alert-param *(COMMA alert-param)
alert-param  =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Allow  =  "Allow" HCOLON [Method *(COMMA Method)]

Authorization     =  "Authorization" HCOLON credentials
credentials       =  ("Digest" LWS digest-response)
                     / other-response
digest-response   =  dig-resp *(COMMA dig-resp)
dig-resp          =  username / realm / nonce / digest-uri
                      / dresponse / algorithm / cnonce
                      / opaque / message-qop
                      / nonce-count / auth-param
username          =  "username" EQUAL username-value
username-value    =  quoted-string
digest-uri        =  "uri" EQUAL LDQUOT digest-uri-value RDQUOT
digest-uri-value  =  rquest-uri ; Equal to request-uri as specified
                     by HTTP/1.1
message-qop       =  "qop" EQUAL qop-value
```

```
cnonce            =  "cnonce" EQUAL cnonce-value
cnonce-value      =  nonce-value
nonce-count       =  "nc" EQUAL nc-value
nc-value          =  8LHEX
dresponse         =  "response" EQUAL request-digest
request-digest    =  LDQUOT 32LHEX RDQUOT
auth-param        =  auth-param-name EQUAL
                     ( token / quoted-string )
auth-param-name   =  token
other-response    =  auth-scheme LWS auth-param
                     *(COMMA auth-param)
auth-scheme       =  token

Authentication-Info  =  "Authentication-Info" HCOLON ainfo
                        *(COMMA ainfo)
ainfo                =  nextnonce / message-qop
                         / response-auth / cnonce
                         / nonce-count
nextnonce            =  "nextnonce" EQUAL nonce-value
response-auth        =  "rspauth" EQUAL response-digest
response-digest      =  LDQUOT *LHEX RDQUOT

Call-ID  =  ( "Call-ID" / "i" ) HCOLON callid
callid   =  word [ "@" word ]

Call-Info   =  "Call-Info" HCOLON info *(COMMA info)
info        =  LAQUOT absoluteURI RAQUOT *( SEMI info-param)
info-param  =  ( "purpose" EQUAL ( "icon" / "info"
               / "card" / token ) ) / generic-param

Contact        =  ("Contact" / "m" ) HCOLON
                  ( STAR / (contact-param *(COMMA contact-param)))
contact-param  =  (name-addr / addr-spec) *(SEMI contact-params)
name-addr      =  [ display-name ] LAQUOT addr-spec RAQUOT
addr-spec      =  SIP-URI / SIPS-URI / absoluteURI
display-name   =  *(token LWS)/ quoted-string

contact-params     =  c-p-q / c-p-expires
                      / contact-extension
c-p-q              =  "q" EQUAL qvalue
c-p-expires        =  "expires" EQUAL delta-seconds
contact-extension  =  generic-param
delta-seconds      =  1*DIGIT

Content-Disposition   =  "Content-Disposition" HCOLON
                         disp-type *( SEMI disp-param )
disp-type             =  "render" / "session" / "icon" / "alert"
                         / disp-extension-token
```

```
disp-param            =  handling-param / generic-param
handling-param        =  "handling" EQUAL
                         ( "optional" / "required"
                         / other-handling )
other-handling        =  token
disp-extension-token  =  token

Content-Encoding  =  ( "Content-Encoding" / "e" ) HCOLON
                     content-coding *(COMMA content-coding)

Content-Language  =  "Content-Language" HCOLON
                     language-tag *(COMMA language-tag)
language-tag      =  primary-tag *( "-" subtag )
primary-tag       =  1*8ALPHA
subtag            =  1*8ALPHA

Content-Length  =  ( "Content-Length" / "l" ) HCOLON 1*DIGIT
Content-Type     =  ( "Content-Type" / "c" ) HCOLON media-type
media-type       =  m-type SLASH m-subtype *(SEMI m-parameter)
m-type           =  discrete-type / composite-type
discrete-type    =  "text" / "image" / "audio" / "video"
                    / "application" / extension-token
composite-type   =  "message" / "multipart" / extension-token
extension-token  =  ietf-token / x-token
ietf-token       =  token
x-token          =  "x-" token
m-subtype        =  extension-token / iana-token
iana-token       =  token
m-parameter      =  m-attribute EQUAL m-value
m-attribute      =  token
m-value          =  token / quoted-string

CSeq  =  "CSeq" HCOLON 1*DIGIT LWS Method

Date          =  "Date" HCOLON SIP-date
SIP-date      =  rfc1123-date
rfc1123-date  =  wkday "," SP date1 SP time SP "GMT"
date1         =  2DIGIT SP month SP 4DIGIT
                 ; day month year (e.g., 02 Jun 1982)
time          =  2DIGIT ":" 2DIGIT ":" 2DIGIT
                 ; 00:00:00 - 23:59:59
wkday         =  "Mon" / "Tue" / "Wed"
                 / "Thu" / "Fri" / "Sat" / "Sun"
month         =  "Jan" / "Feb" / "Mar" / "Apr"
                 / "May" / "Jun" / "Jul" / "Aug"
                 / "Sep" / "Oct" / "Nov" / "Dec"

Error-Info  =  "Error-Info" HCOLON error-uri *(COMMA error-uri)
```

```
error-uri   =  LAQUOT absoluteURI RAQUOT *( SEMI generic-param )

Expires     =  "Expires" HCOLON delta-seconds
From        =  ( "From" / "f" ) HCOLON from-spec
from-spec   =  ( name-addr / addr-spec )
               *( SEMI from-param )
from-param  =  tag-param / generic-param
tag-param   =  "tag" EQUAL token

In-Reply-To  =  "In-Reply-To" HCOLON callid *(COMMA callid)

Max-Forwards  =  "Max-Forwards" HCOLON 1*DIGIT

MIME-Version  =  "MIME-Version" HCOLON 1*DIGIT "." 1*DIGIT

Min-Expires  =  "Min-Expires" HCOLON delta-seconds

Organization  =  "Organization" HCOLON [TEXT-UTF8-TRIM]

Priority        =  "Priority" HCOLON priority-value
priority-value  =  "emergency" / "urgent" / "normal"
                  / "non-urgent" / other-priority
other-priority  =  token

Proxy-Authenticate  =  "Proxy-Authenticate" HCOLON challenge
challenge           =  ("Digest" LWS digest-cln *(COMMA digest-cln))
                       / other-challenge
other-challenge     =  auth-scheme LWS auth-param
                       *(COMMA auth-param)
digest-cln          =  realm / domain / nonce
                        / opaque / stale / algorithm
                        / qop-options / auth-param
realm               =  "realm" EQUAL realm-value
realm-value         =  quoted-string
domain              =  "domain" EQUAL LDQUOT URI
                       *( 1*SP URI ) RDQUOT
URI                 =  absoluteURI / abs-path
nonce               =  "nonce" EQUAL nonce-value
nonce-value         =  quoted-string
opaque              =  "opaque" EQUAL quoted-string
stale               =  "stale" EQUAL ( "true" / "false" )
algorithm           =  "algorithm" EQUAL ( "MD5" / "MD5-sess"
                       / token )
qop-options         =  "qop" EQUAL LDQUOT qop-value
                       *("," qop-value) RDQUOT
qop-value           =  "auth" / "auth-int" / token

Proxy-Authorization  =  "Proxy-Authorization" HCOLON credentials
```

```
Proxy-Require  =  "Proxy-Require" HCOLON option-tag
                  *(COMMA option-tag)
option-tag     =  token

Record-Route  =  "Record-Route" HCOLON rec-route *(COMMA rec-route)
rec-route     =  name-addr *( SEMI rr-param )
rr-param      =  generic-param

Reply-To      =  "Reply-To" HCOLON rplyto-spec
rplyto-spec   =  ( name-addr / addr-spec )
                 *( SEMI rplyto-param )
rplyto-param  =  generic-param
Require       =  "Require" HCOLON option-tag *(COMMA option-tag)

Retry-After  =  "Retry-After" HCOLON delta-seconds
                [ comment ] *( SEMI retry-param )

retry-param  =  ("duration" EQUAL delta-seconds)
                / generic-param

Route        =  "Route" HCOLON route-param *(COMMA route-param)
route-param  =  name-addr *( SEMI rr-param )

Server           =  "Server" HCOLON server-val *(LWS server-val)
server-val       =  product / comment
product          =  token [SLASH product-version]
product-version  =  token

Subject  =  ( "Subject" / "s" ) HCOLON [TEXT-UTF8-TRIM]

Supported  =  ( "Supported" / "k" ) HCOLON
              [option-tag *(COMMA option-tag)]

Timestamp  =  "Timestamp" HCOLON 1*(DIGIT)
               [ "." *(DIGIT) ] [ LWS delay ]
delay      =  *(DIGIT) [ "." *(DIGIT) ]

To        =  ( "To" / "t" ) HCOLON ( name-addr
             / addr-spec ) *( SEMI to-param )
to-param  =  tag-param / generic-param

Unsupported  =  "Unsupported" HCOLON option-tag *(COMMA option-tag)
User-Agent   =  "User-Agent" HCOLON server-val *(LWS server-val)
```

```
Via               =  ( "Via" / "v" ) HCOLON via-parm *(COMMA via-parm)
via-parm          =  sent-protocol LWS sent-by *( SEMI via-params )
via-params        =  via-ttl / via-maddr
                     / via-received / via-branch
                     / via-extension
via-ttl           =  "ttl" EQUAL ttl
via-maddr         =  "maddr" EQUAL host
via-received      =  "received" EQUAL (IPv4address / IPv6address)
via-branch        =  "branch" EQUAL token
via-extension     =  generic-param
sent-protocol     =  protocol-name SLASH protocol-version
                     SLASH transport
protocol-name     =  "SIP" / token
protocol-version  =  token
transport         =  "UDP" / "TCP" / "TLS" / "SCTP"
                     / other-transport
sent-by           =  host [ COLON port ]
ttl               =  1*3DIGIT ; 0 to 255

Warning        =  "Warning" HCOLON warning-value *(COMMA warning-value)
warning-value  =  warn-code SP warn-agent SP warn-text
warn-code      =  3DIGIT
warn-agent     =  hostport / pseudonym
                  ;  the name or pseudonym of the server adding
                  ;  the Warning header, for use in debugging
warn-text      =  quoted-string
pseudonym      =  token

WWW-Authenticate  =  "WWW-Authenticate" HCOLON challenge

extension-header  =  header-name HCOLON header-value
header-name       =  token
header-value      =  *(TEXT-UTF8char / UTF8-CONT / LWS)
message-body  =  *OCTET
```

26 Security Considerations: Threat Model and Security Usage Recommendations

SIP is not an easy protocol to secure. Its use of intermediaries, its multi-faceted trust relationships, its expected usage between elements with no trust at all, and its user-to-user operation make security far from trivial. Security solutions are needed that are deployable today, without extensive coordination, in a wide variety of environments and usages. In order to meet these diverse needs, several distinct mechanisms applicable to different aspects and usages of SIP will be required.

Note that the security of SIP signaling itself has no bearing on the security of protocols used in concert with SIP such as RTP, or with the security implications of any specific bodies SIP might carry (although MIME security plays a substantial role in securing SIP). Any media associated with a session can be encrypted end-to-end independently of any associated SIP signaling. Media encryption is outside the scope of this document.

The considerations that follow first examine a set of classic threat models that broadly identify the security needs of SIP. The set of security services required to address these threats is then detailed, followed by an explanation of several security mechanisms that can be used to provide these services. Next, the requirements for implementers of SIP are enumerated, along with exemplary deployments in which these security mechanisms could be used to improve the security of SIP. Some notes on privacy conclude this section.

26.1 Attacks and Threat Models

This section details some threats that should be common to most deployments of SIP. These threats have been chosen specifically to illustrate each of the security services that SIP requires.

The following examples by no means provide an exhaustive list of the threats against SIP; rather, these are "classic" threats that demonstrate the need for particular security services that can potentially prevent whole categories of threats.

These attacks assume an environment in which attackers can potentially read any packet on the network - it is anticipated that SIP will frequently be used on the public Internet. Attackers on the network may be able to modify packets (perhaps at some compromised intermediary). Attackers may wish to steal services, eavesdrop on communications, or disrupt sessions.

26.1.1 Registration Hijacking

The SIP registration mechanism allows a user agent to identify itself to a registrar as a device at which a user (designated by an address of record) is located. A registrar assesses the identity asserted in the From header field of a REGISTER message to determine whether this request can modify the contact addresses associated with the address-of-record in the To header field. While these two fields are frequently the same, there are many valid deployments in which a third-party may register contacts on a user’s behalf.

The From header field of a SIP request, however, can be modified arbitrarily by the owner of a UA, and this opens the door to malicious registrations. An attacker that successfully impersonates a party authorized to change contacts associated with an address-of-record could, for example, de-register all existing contacts for a URI and then register their own device as the appropriate contact address, thereby directing all requests for the affected user to the attacker's device.

This threat belongs to a family of threats that rely on the absence of cryptographic assurance of a request's originator. Any SIP UAS that represents a valuable service (a gateway that interworks SIP requests with traditional telephone calls, for example) might want to control access to its resources by authenticating requests that it receives. Even end-user UAs, for example SIP phones, have an interest in ascertaining the identities of originators of requests.

This threat demonstrates the need for security services that enable SIP entities to authenticate the originators of requests.

26.1.2 Impersonating a Server

The domain to which a request is destined is generally specified in the Request-URI. UAs commonly contact a server in this domain directly in order to deliver a request. However, there is always a possibility that an attacker could impersonate the remote server, and that the UA's request could be intercepted by some other party.

For example, consider a case in which a redirect server at one domain, chicago.com, impersonates a redirect server at another domain, biloxi.com. A user agent sends a request to biloxi.com, but the redirect server at chicago.com answers with a forged response that has appropriate SIP header fields for a response from biloxi.com. The forged contact addresses in the redirection response could direct the originating UA to inappropriate or insecure resources, or simply prevent requests for biloxi.com from succeeding.

This family of threats has a vast membership, many of which are critical. As a converse to the registration hijacking threat, consider the case in which a registration sent to biloxi.com is intercepted by chicago.com, which replies to the intercepted registration with a forged 301 (Moved Permanently) response. This response might seem to come from biloxi.com yet designate chicago.com as the appropriate registrar. All future REGISTER requests from the originating UA would then go to chicago.com.

Prevention of this threat requires a means by which UAs can authenticate the servers to whom they send requests.

26.1.3 Tampering with Message Bodies

As a matter of course, SIP UAs route requests through trusted proxy servers. Regardless of how that trust is established (authentication of proxies is discussed elsewhere in this section), a UA may trust a proxy server to route a request, but not to inspect or possibly modify the bodies contained in that request.

Consider a UA that is using SIP message bodies to communicate session encryption keys for a media session. Although it trusts the proxy server of the domain it is contacting to deliver signaling properly, it may not want the administrators of that domain to be capable of decrypting any subsequent media session. Worse yet, if the proxy server were actively malicious, it could modify the session key, either acting as a man-in-the-middle, or perhaps changing the security characteristics requested by the originating UA.

This family of threats applies not only to session keys, but to most conceivable forms of content carried end-to-end in SIP. These might include MIME bodies that should be rendered to the user, SDP, or encapsulated telephony signals, among others. Attackers might attempt to modify SDP bodies, for example, in order to point RTP media streams to a wiretapping device in order to eavesdrop on subsequent voice communications.

Also note that some header fields in SIP are meaningful end-to-end, for example, Subject. UAs might be protective of these header fields as well as bodies (a malicious intermediary changing the Subject header field might make an important request appear to be spam, for example). However, since many header fields are legitimately inspected or altered by proxy servers as a request is routed, not all header fields should be secured end-to-end.

For these reasons, the UA might want to secure SIP message bodies, and in some limited cases header fields, end-to-end. The security services required for bodies include confidentiality, integrity, and authentication. These end-to-end services should be independent of the means used to secure interactions with intermediaries such as proxy servers.

26.1.4 Tearing Down Sessions

Once a dialog has been established by initial messaging, subsequent requests can be sent that modify the state of the dialog and/or session. It is critical that principals in a session can be certain that such requests are not forged by attackers.

Consider a case in which a third-party attacker captures some initial messages in a dialog shared by two parties in order to learn the parameters of the session (To tag, From tag, and so forth) and then inserts a BYE request into the session. The attacker could opt to forge the request such that it seemed to come from either participant. Once the BYE is received by its target, the session will be torn down prematurely.

Similar mid-session threats include the transmission of forged re-INVITEs that alter the session (possibly to reduce session security or redirect media streams as part of a wiretapping attack).

The most effective countermeasure to this threat is the authentication of the sender of the BYE. In this instance, the recipient needs only know that the BYE came from the same party with whom the corresponding dialog was established (as opposed to ascertaining the absolute identity of the sender). Also, if the attacker is unable to learn the parameters of the session due to confidentiality, it would not be possible to forge the BYE. However, some intermediaries (like proxy servers) will need to inspect those parameters as the session is established.

26.1.5 Denial of Service and Amplification

Denial-of-service attacks focus on rendering a particular network element unavailable, usually by directing an excessive amount of network traffic at its interfaces. A distributed denial-of-service attack allows one network user to cause multiple network hosts to flood a target host with a large amount of network traffic.

In many architectures, SIP proxy servers face the public Internet in order to accept requests from worldwide IP endpoints. SIP creates a number of potential opportunities for distributed denial-of-service attacks that must be recognized and addressed by the implementers and operators of SIP systems.

Attackers can create bogus requests that contain a falsified source IP address and a corresponding Via header field that identify a targeted host as the originator of the request and then send this request to a large number of SIP network elements, thereby using hapless SIP UAs or proxies to generate denial-of-service traffic aimed at the target.

Similarly, attackers might use falsified Route header field values in a request that identify the target host and then send such messages to forking proxies that will amplify messaging sent to the target.

Record-Route could be used to similar effect when the attacker is certain that the SIP dialog initiated by the request will result in numerous transactions originating in the backwards direction.

A number of denial-of-service attacks open up if REGISTER requests are not properly authenticated and authorized by registrars. Attackers could de-register some or all users in an administrative domain, thereby preventing these users from being invited to new sessions. An attacker could also register a large number of contacts designating the same host for a given address-of-record in order to use the registrar and any associated proxy servers as amplifiers in a denial-of-service attack. Attackers might also attempt to deplete available memory and disk resources of a registrar by registering huge numbers of bindings.

The use of multicast to transmit SIP requests can greatly increase the potential for denial-of-service attacks.

These problems demonstrate a general need to define architectures that minimize the risks of denial-of-service, and the need to be mindful in recommendations for security mechanisms of this class of attacks.

26.2 Security Mechanisms

From the threats described above, we gather that the fundamental security services required for the SIP protocol are: preserving the confidentiality and integrity of messaging, preventing replay attacks or message spoofing, providing for the authentication and privacy of the participants in a session, and preventing denial-of-service attacks. Bodies within SIP messages separately require the security services of confidentiality, integrity, and authentication.

Rather than defining new security mechanisms specific to SIP, SIP reuses wherever possible existing security models derived from the HTTP and SMTP space.

Full encryption of messages provides the best means to preserve the confidentiality of signaling - it can also guarantee that messages are not modified by any malicious intermediaries. However, SIP requests and responses cannot be naively encrypted end-to-end in their entirety because message fields such as the Request-URI, Route, and Via need to be visible to proxies in most network architectures so that SIP requests are routed correctly. Note that proxy servers need to modify some features of messages as well (such as adding Via header field values) in order for SIP to function. Proxy servers must therefore be trusted, to some degree, by SIP UAs. To this purpose, low-layer security mechanisms for SIP are recommended, which

encrypt the entire SIP requests or responses on the wire on a hop-by-hop basis, and that allow endpoints to verify the identity of proxy servers to whom they send requests.

SIP entities also have a need to identify one another in a secure fashion. When a SIP endpoint asserts the identity of its user to a peer UA or to a proxy server, that identity should in some way be verifiable. A cryptographic authentication mechanism is provided in SIP to address this requirement.

An independent security mechanism for SIP message bodies supplies an alternative means of end-to-end mutual authentication, as well as providing a limit on the degree to which user agents must trust intermediaries.

26.2.1 Transport and Network Layer Security

Transport or network layer security encrypts signaling traffic, guaranteeing message confidentiality and integrity.

Oftentimes, certificates are used in the establishment of lower-layer security, and these certificates can also be used to provide a means of authentication in many architectures.

Two popular alternatives for providing security at the transport and network layer are, respectively, TLS [25] and IPSec [26].

IPSec is a set of network-layer protocol tools that collectively can be used as a secure replacement for traditional IP (Internet Protocol). IPSec is most commonly used in architectures in which a set of hosts or administrative domains have an existing trust relationship with one another. IPSec is usually implemented at the operating system level in a host, or on a security gateway that provides confidentiality and integrity for all traffic it receives from a particular interface (as in a VPN architecture). IPSec can also be used on a hop-by-hop basis.

In many architectures IPSec does not require integration with SIP applications; IPSec is perhaps best suited to deployments in which adding security directly to SIP hosts would be arduous. UAs that have a pre-shared keying relationship with their first-hop proxy server are also good candidates to use IPSec. Any deployment of IPSec for SIP would require an IPSec profile describing the protocol tools that would be required to secure SIP. No such profile is given in this document.

TLS provides transport-layer security over connection-oriented protocols (for the purposes of this document, TCP); "tls" (signifying TLS over TCP) can be specified as the desired transport protocol within a Via header field value or a SIP-URI. TLS is most suited to architectures in which hop-by-hop security is required between hosts with no pre-existing trust association. For example, Alice trusts her local proxy server, which after a certificate exchange decides to trust Bob's local proxy server, which Bob trusts, hence Bob and Alice can communicate securely.

TLS must be tightly coupled with a SIP application. Note that transport mechanisms are specified on a hop-by-hop basis in SIP, thus a UA that sends requests over TLS to a proxy server has no assurance that TLS will be used end-to-end.

The TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite [6] MUST be supported at a minimum by implementers when TLS is used in a SIP application. For purposes of backwards compatibility, proxy servers, redirect servers, and registrars SHOULD support TLS_RSA_WITH_3DES_EDE_CBC_SHA. Implementers MAY also support any other ciphersuite.

26.2.2 SIPS URI Scheme

The SIPS URI scheme adheres to the syntax of the SIP URI (described in 19), although the scheme string is "sips" rather than "sip". The semantics of SIPS are very different from the SIP URI, however. SIPS allows resources to specify that they should be reached securely.

A SIPS URI can be used as an address-of-record for a particular user - the URI by which the user is canonically known (on their business cards, in the From header field of their requests, in the To header field of REGISTER requests). When used as the Request-URI of a request, the SIPS scheme signifies that each hop over which the request is forwarded, until the request reaches the SIP entity responsible for the domain portion of the Request-URI, must be secured with TLS; once it reaches the domain in question it is handled in accordance with local security and routing policy, quite possibly using TLS for any last hop to a UAS. When used by the originator of a request (as would be the case if they employed a SIPS URI as the address-of-record of the target), SIPS dictates that the entire request path to the target domain be so secured.

The SIPS scheme is applicable to many of the other ways in which SIP URIs are used in SIP today in addition to the Request-URI, including in addresses-of-record, contact addresses (the contents of Contact headers, including those of REGISTER methods), and Route headers. In each instance, the SIPS URI scheme allows these existing fields to

designate secure resources. The manner in which a SIPS URI is dereferenced in any of these contexts has its own security properties which are detailed in [4].

The use of SIPS in particular entails that mutual TLS authentication SHOULD be employed, as SHOULD the ciphersuite TLS_RSA_WITH_AES_128_CBC_SHA. Certificates received in the authentication process SHOULD be validated with root certificates held by the client; failure to validate a certificate SHOULD result in the failure of the request.

> Note that in the SIPS URI scheme, transport is independent of TLS, and thus "sips:alice@atlanta.com;transport=tcp" and "sips:alice@atlanta.com;transport=sctp" are both valid (although note that UDP is not a valid transport for SIPS). The use of "transport=tls" has consequently been deprecated, partly because it was specific to a single hop of the request. This is a change since RFC 2543.

Users that distribute a SIPS URI as an address-of-record may elect to operate devices that refuse requests over insecure transports.

26.2.3 HTTP Authentication

SIP provides a challenge capability, based on HTTP authentication, that relies on the 401 and 407 response codes as well as header fields for carrying challenges and credentials. Without significant modification, the reuse of the HTTP Digest authentication scheme in SIP allows for replay protection and one-way authentication.

The usage of Digest authentication in SIP is detailed in Section 22.

26.2.4 S/MIME

As is discussed above, encrypting entire SIP messages end-to-end for the purpose of confidentiality is not appropriate because network intermediaries (like proxy servers) need to view certain header fields in order to route messages correctly, and if these intermediaries are excluded from security associations, then SIP messages will essentially be non-routable.

However, S/MIME allows SIP UAs to encrypt MIME bodies within SIP, securing these bodies end-to-end without affecting message headers. S/MIME can provide end-to-end confidentiality and integrity for message bodies, as well as mutual authentication. It is also possible to use S/MIME to provide a form of integrity and confidentiality for SIP header fields through SIP message tunneling.

The usage of S/MIME in SIP is detailed in Section 23.

26.3 Implementing Security Mechanisms

26.3.1 Requirements for Implementers of SIP

Proxy servers, redirect servers, and registrars MUST implement TLS, and MUST support both mutual and one-way authentication. It is strongly RECOMMENDED that UAs be capable initiating TLS; UAs MAY also be capable of acting as a TLS server. Proxy servers, redirect servers, and registrars SHOULD possess a site certificate whose subject corresponds to their canonical hostname. UAs MAY have certificates of their own for mutual authentication with TLS, but no provisions are set forth in this document for their use. All SIP elements that support TLS MUST have a mechanism for validating certificates received during TLS negotiation; this entails possession of one or more root certificates issued by certificate authorities (preferably well-known distributors of site certificates comparable to those that issue root certificates for web browsers).

All SIP elements that support TLS MUST also support the SIPS URI scheme.

Proxy servers, redirect servers, registrars, and UAs MAY also implement IPSec or other lower-layer security protocols.

When a UA attempts to contact a proxy server, redirect server, or registrar, the UAC SHOULD initiate a TLS connection over which it will send SIP messages. In some architectures, UASs MAY receive requests over such TLS connections as well.

Proxy servers, redirect servers, registrars, and UAs MUST implement Digest Authorization, encompassing all of the aspects required in 22. Proxy servers, redirect servers, and registrars SHOULD be configured with at least one Digest realm, and at least one "realm" string supported by a given server SHOULD correspond to the server's hostname or domainname.

UAs MAY support the signing and encrypting of MIME bodies, and transference of credentials with S/MIME as described in Section 23. If a UA holds one or more root certificates of certificate authorities in order to validate certificates for TLS or IPSec, it SHOULD be capable of reusing these to verify S/MIME certificates, as appropriate. A UA MAY hold root certificates specifically for validating S/MIME certificates.

Note that is it anticipated that future security extensions may upgrade the normative strength associated with S/MIME as S/MIME implementations appear and the problem space becomes better understood.

26.3.2 Security Solutions

The operation of these security mechanisms in concert can follow the existing web and email security models to some degree. At a high level, UAs authenticate themselves to servers (proxy servers, redirect servers, and registrars) with a Digest username and password; servers authenticate themselves to UAs one hop away, or to another server one hop away (and vice versa), with a site certificate delivered by TLS.

On a peer-to-peer level, UAs trust the network to authenticate one another ordinarily; however, S/MIME can also be used to provide direct authentication when the network does not, or if the network itself is not trusted.

The following is an illustrative example in which these security mechanisms are used by various UAs and servers to prevent the sorts of threats described in Section 26.1. While implementers and network administrators MAY follow the normative guidelines given in the remainder of this section, these are provided only as example implementations.

26.3.2.1 Registration

When a UA comes online and registers with its local administrative domain, it SHOULD establish a TLS connection with its registrar (Section 10 describes how the UA reaches its registrar). The registrar SHOULD offer a certificate to the UA, and the site identified by the certificate MUST correspond with the domain in which the UA intends to register; for example, if the UA intends to register the address-of-record 'alice@atlanta.com', the site certificate must identify a host within the atlanta.com domain (such as sip.atlanta.com). When it receives the TLS Certificate message, the UA SHOULD verify the certificate and inspect the site identified by the certificate. If the certificate is invalid, revoked, or if it does not identify the appropriate party, the UA MUST NOT send the REGISTER message and otherwise proceed with the registration.

When a valid certificate has been provided by the registrar, the UA knows that the registrar is not an attacker who might redirect the UA, steal passwords, or attempt any similar attacks.

The UA then creates a REGISTER request that SHOULD be addressed to a Request-URI corresponding to the site certificate received from the registrar. When the UA sends the REGISTER request over the existing TLS connection, the registrar SHOULD challenge the request with a 401 (Proxy Authentication Required) response. The "realm" parameter within the Proxy-Authenticate header field of the response SHOULD correspond to the domain previously given by the site certificate. When the UAC receives the challenge, it SHOULD either prompt the user for credentials or take an appropriate credential from a keyring corresponding to the "realm" parameter in the challenge. The username of this credential SHOULD correspond with the "userinfo" portion of the URI in the To header field of the REGISTER request. Once the Digest credentials have been inserted into an appropriate Proxy-Authorization header field, the REGISTER should be resubmitted to the registrar.

> Since the registrar requires the user agent to authenticate itself, it would be difficult for an attacker to forge REGISTER requests for the user's address-of-record. Also note that since the REGISTER is sent over a confidential TLS connection, attackers will not be able to intercept the REGISTER to record credentials for any possible replay attack.

Once the registration has been accepted by the registrar, the UA SHOULD leave this TLS connection open provided that the registrar also acts as the proxy server to which requests are sent for users in this administrative domain. The existing TLS connection will be reused to deliver incoming requests to the UA that has just completed registration.

> Because the UA has already authenticated the server on the other side of the TLS connection, all requests that come over this connection are known to have passed through the proxy server - attackers cannot create spoofed requests that appear to have been sent through that proxy server.

26.3.2.2 Interdomain Requests

Now let's say that Alice's UA would like to initiate a session with a user in a remote administrative domain, namely "bob@biloxi.com". We will also say that the local administrative domain (atlanta.com) has a local outbound proxy.

The proxy server that handles inbound requests for an administrative domain MAY also act as a local outbound proxy; for simplicity's sake we'll assume this to be the case for atlanta.com (otherwise the user agent would initiate a new TLS connection to a separate server at this point). Assuming that the client has completed the registration

process described in the preceding section, it SHOULD reuse the TLS connection to the local proxy server when it sends an INVITE request to another user. The UA SHOULD reuse cached credentials in the INVITE to avoid prompting the user unnecessarily.

When the local outbound proxy server has validated the credentials presented by the UA in the INVITE, it SHOULD inspect the Request-URI to determine how the message should be routed (see [4]). If the "domainname" portion of the Request-URI had corresponded to the local domain (atlanta.com) rather than biloxi.com, then the proxy server would have consulted its location service to determine how best to reach the requested user.

> Had "alice@atlanta.com" been attempting to contact, say, "alex@atlanta.com", the local proxy would have proxied to the request to the TLS connection Alex had established with the registrar when he registered. Since Alex would receive this request over his authenticated channel, he would be assured that Alice's request had been authorized by the proxy server of the local administrative domain.

However, in this instance the Request-URI designates a remote domain. The local outbound proxy server at atlanta.com SHOULD therefore establish a TLS connection with the remote proxy server at biloxi.com. Since both of the participants in this TLS connection are servers that possess site certificates, mutual TLS authentication SHOULD occur. Each side of the connection SHOULD verify and inspect the certificate of the other, noting the domain name that appears in the certificate for comparison with the header fields of SIP messages. The atlanta.com proxy server, for example, SHOULD verify at this stage that the certificate received from the remote side corresponds with the biloxi.com domain. Once it has done so, and TLS negotiation has completed, resulting in a secure channel between the two proxies, the atlanta.com proxy can forward the INVITE request to biloxi.com.

The proxy server at biloxi.com SHOULD inspect the certificate of the proxy server at atlanta.com in turn and compare the domain asserted by the certificate with the "domainname" portion of the From header field in the INVITE request. The biloxi proxy MAY have a strict security policy that requires it to reject requests that do not match the administrative domain from which they have been proxied.

> Such security policies could be instituted to prevent the SIP equivalent of SMTP 'open relays' that are frequently exploited to generate spam.

This policy, however, only guarantees that the request came from the domain it ascribes to itself; it does not allow biloxi.com to ascertain how atlanta.com authenticated Alice. Only if biloxi.com has some other way of knowing atlanta.com's authentication policies could it possibly ascertain how Alice proved her identity. biloxi.com might then institute an even stricter policy that forbids requests that come from domains that are not known administratively to share a common authentication policy with biloxi.com.

Once the INVITE has been approved by the biloxi proxy, the proxy server SHOULD identify the existing TLS channel, if any, associated with the user targeted by this request (in this case "bob@biloxi.com"). The INVITE should be proxied through this channel to Bob. Since the request is received over a TLS connection that had previously been authenticated as the biloxi proxy, Bob knows that the From header field was not tampered with and that atlanta.com has validated Alice, although not necessarily whether or not to trust Alice's identity.

Before they forward the request, both proxy servers SHOULD add a Record-Route header field to the request so that all future requests in this dialog will pass through the proxy servers. The proxy servers can thereby continue to provide security services for the lifetime of this dialog. If the proxy servers do not add themselves to the Record-Route, future messages will pass directly end-to-end between Alice and Bob without any security services (unless the two parties agree on some independent end-to-end security such as S/MIME). In this respect the SIP trapezoid model can provide a nice structure where conventions of agreement between the site proxies can provide a reasonably secure channel between Alice and Bob.

> An attacker preying on this architecture would, for example, be unable to forge a BYE request and insert it into the signaling stream between Bob and Alice because the attacker has no way of ascertaining the parameters of the session and also because the integrity mechanism transitively protects the traffic between Alice and Bob.

26.3.2.3 Peer-to-Peer Requests

Alternatively, consider a UA asserting the identity "carol@chicago.com" that has no local outbound proxy. When Carol wishes to send an INVITE to "bob@biloxi.com", her UA SHOULD initiate a TLS connection with the biloxi proxy directly (using the mechanism described in [4] to determine how to best to reach the given Request-URI). When her UA receives a certificate from the biloxi proxy, it SHOULD be verified normally before she passes her INVITE across the TLS connection. However, Carol has no means of proving

her identity to the biloxi proxy, but she does have a CMS-detached signature over a "message/sip" body in the INVITE. It is unlikely in this instance that Carol would have any credentials in the biloxi.com realm, since she has no formal association with biloxi.com. The biloxi proxy MAY also have a strict policy that precludes it from even bothering to challenge requests that do not have biloxi.com in the "domainname" portion of the From header field - it treats these users as unauthenticated.

The biloxi proxy has a policy for Bob that all non-authenticated requests should be redirected to the appropriate contact address registered against 'bob@biloxi.com', namely <sip:bob@192.0.2.4>. Carol receives the redirection response over the TLS connection she established with the biloxi proxy, so she trusts the veracity of the contact address.

Carol SHOULD then establish a TCP connection with the designated address and send a new INVITE with a Request-URI containing the received contact address (recomputing the signature in the body as the request is readied). Bob receives this INVITE on an insecure interface, but his UA inspects and, in this instance, recognizes the From header field of the request and subsequently matches a locally cached certificate with the one presented in the signature of the body of the INVITE. He replies in similar fashion, authenticating himself to Carol, and a secure dialog begins.

> Sometimes firewalls or NATs in an administrative domain could preclude the establishment of a direct TCP connection to a UA. In these cases, proxy servers could also potentially relay requests to UAs in a way that has no trust implications (for example, forgoing an existing TLS connection and forwarding the request over cleartext TCP) as local policy dictates.

26.3.2.4 DoS Protection

In order to minimize the risk of a denial-of-service attack against architectures using these security solutions, implementers should take note of the following guidelines.

When the host on which a SIP proxy server is operating is routable from the public Internet, it SHOULD be deployed in an administrative domain with defensive operational policies (blocking source-routed traffic, preferably filtering ping traffic). Both TLS and IPSec can also make use of bastion hosts at the edges of administrative domains that participate in the security associations to aggregate secure tunnels and sockets. These bastion hosts can also take the brunt of denial-of-service attacks, ensuring that SIP hosts within the administrative domain are not encumbered with superfluous messaging.

No matter what security solutions are deployed, floods of messages directed at proxy servers can lock up proxy server resources and prevent desirable traffic from reaching its destination. There is a computational expense associated with processing a SIP transaction at a proxy server, and that expense is greater for stateful proxy servers than it is for stateless proxy servers. Therefore, stateful proxies are more susceptible to flooding than stateless proxy servers.

UAs and proxy servers SHOULD challenge questionable requests with only a single 401 (Unauthorized) or 407 (Proxy Authentication Required), forgoing the normal response retransmission algorithm, and thus behaving statelessly towards unauthenticated requests.

> Retransmitting the 401 (Unauthorized) or 407 (Proxy Authentication Required) status response amplifies the problem of an attacker using a falsified header field value (such as Via) to direct traffic to a third party.

In summary, the mutual authentication of proxy servers through mechanisms such as TLS significantly reduces the potential for rogue intermediaries to introduce falsified requests or responses that can deny service. This commensurately makes it harder for attackers to make innocent SIP nodes into agents of amplification.

26.4 Limitations

Although these security mechanisms, when applied in a judicious manner, can thwart many threats, there are limitations in the scope of the mechanisms that must be understood by implementers and network operators.

26.4.1 HTTP Digest

One of the primary limitations of using HTTP Digest in SIP is that the integrity mechanisms in Digest do not work very well for SIP. Specifically, they offer protection of the Request-URI and the method of a message, but not for any of the header fields that UAs would most likely wish to secure.

The existing replay protection mechanisms described in RFC 2617 also have some limitations for SIP. The next-nonce mechanism, for example, does not support pipelined requests. The nonce-count mechanism should be used for replay protection.

Another limitation of HTTP Digest is the scope of realms. Digest is valuable when a user wants to authenticate themselves to a resource with which they have a pre-existing association, like a service

provider of which the user is a customer (which is quite a common scenario and thus Digest provides an extremely useful function). By way of contrast, the scope of TLS is interdomain or multirealm, since certificates are often globally verifiable, so that the UA can authenticate the server with no pre-existing association.

26.4.2 S/MIME

The largest outstanding defect with the S/MIME mechanism is the lack of a prevalent public key infrastructure for end users. If self-signed certificates (or certificates that cannot be verified by one of the participants in a dialog) are used, the SIP-based key exchange mechanism described in Section 23.2 is susceptible to a man-in-the-middle attack with which an attacker can potentially inspect and modify S/MIME bodies. The attacker needs to intercept the first exchange of keys between the two parties in a dialog, remove the existing CMS-detached signatures from the request and response, and insert a different CMS-detached signature containing a certificate supplied by the attacker (but which seems to be a certificate for the proper address-of-record). Each party will think they have exchanged keys with the other, when in fact each has the public key of the attacker.

It is important to note that the attacker can only leverage this vulnerability on the first exchange of keys between two parties - on subsequent occasions, the alteration of the key would be noticeable to the UAs. It would also be difficult for the attacker to remain in the path of all future dialogs between the two parties over time (as potentially days, weeks, or years pass).

SSH is susceptible to the same man-in-the-middle attack on the first exchange of keys; however, it is widely acknowledged that while SSH is not perfect, it does improve the security of connections. The use of key fingerprints could provide some assistance to SIP, just as it does for SSH. For example, if two parties use SIP to establish a voice communications session, each could read off the fingerprint of the key they received from the other, which could be compared against the original. It would certainly be more difficult for the man-in-the-middle to emulate the voices of the participants than their signaling (a practice that was used with the Clipper chip-based secure telephone).

The S/MIME mechanism allows UAs to send encrypted requests without preamble if they possess a certificate for the destination address-of-record on their keyring. However, it is possible that any particular device registered for an address-of-record will not hold the certificate that has been previously employed by the device's current user, and that it will therefore be unable to process an

encrypted request properly, which could lead to some avoidable error signaling. This is especially likely when an encrypted request is forked.

The keys associated with S/MIME are most useful when associated with a particular user (an address-of-record) rather than a device (a UA). When users move between devices, it may be difficult to transport private keys securely between UAs; how such keys might be acquired by a device is outside the scope of this document.

Another, more prosaic difficulty with the S/MIME mechanism is that it can result in very large messages, especially when the SIP tunneling mechanism described in Section 23.4 is used. For that reason, it is RECOMMENDED that TCP should be used as a transport protocol when S/MIME tunneling is employed.

26.4.3 TLS

The most commonly voiced concern about TLS is that it cannot run over UDP; TLS requires a connection-oriented underlying transport protocol, which for the purposes of this document means TCP.

It may also be arduous for a local outbound proxy server and/or registrar to maintain many simultaneous long-lived TLS connections with numerous UAs. This introduces some valid scalability concerns, especially for intensive ciphersuites. Maintaining redundancy of long-lived TLS connections, especially when a UA is solely responsible for their establishment, could also be cumbersome.

TLS only allows SIP entities to authenticate servers to which they are adjacent; TLS offers strictly hop-by-hop security. Neither TLS, nor any other mechanism specified in this document, allows clients to authenticate proxy servers to whom they cannot form a direct TCP connection.

26.4.4 SIPS URIs

Actually using TLS on every segment of a request path entails that the terminating UAS must be reachable over TLS (perhaps registering with a SIPS URI as a contact address). This is the preferred use of SIPS. Many valid architectures, however, use TLS to secure part of the request path, but rely on some other mechanism for the final hop to a UAS, for example. Thus SIPS cannot guarantee that TLS usage will be truly end-to-end. Note that since many UAs will not accept incoming TLS connections, even those UAs that do support TLS may be required to maintain persistent TLS connections as described in the TLS limitations section above in order to receive requests over TLS as a UAS.

Location services are not required to provide a SIPS binding for a SIPS Request-URI. Although location services are commonly populated by user registrations (as described in Section 10.2.1), various other protocols and interfaces could conceivably supply contact addresses for an AOR, and these tools are free to map SIPS URIs to SIP URIs as appropriate. When queried for bindings, a location service returns its contact addresses without regard for whether it received a request with a SIPS Request-URI. If a redirect server is accessing the location service, it is up to the entity that processes the Contact header field of a redirection to determine the propriety of the contact addresses.

Ensuring that TLS will be used for all of the request segments up to the target domain is somewhat complex. It is possible that cryptographically authenticated proxy servers along the way that are non-compliant or compromised may choose to disregard the forwarding rules associated with SIPS (and the general forwarding rules in Section 16.6). Such malicious intermediaries could, for example, retarget a request from a SIPS URI to a SIP URI in an attempt to downgrade security.

Alternatively, an intermediary might legitimately retarget a request from a SIP to a SIPS URI. Recipients of a request whose Request-URI uses the SIPS URI scheme thus cannot assume on the basis of the Request-URI alone that SIPS was used for the entire request path (from the client onwards).

To address these concerns, it is RECOMMENDED that recipients of a request whose Request-URI contains a SIP or SIPS URI inspect the To header field value to see if it contains a SIPS URI (though note that it does not constitute a breach of security if this URI has the same scheme but is not equivalent to the URI in the To header field). Although clients may choose to populate the Request-URI and To header field of a request differently, when SIPS is used this disparity could be interpreted as a possible security violation, and the request could consequently be rejected by its recipient. Recipients MAY also inspect the Via header chain in order to double-check whether or not TLS was used for the entire request path until the local administrative domain was reached. S/MIME may also be used by the originating UAC to help ensure that the original form of the To header field is carried end-to-end.

If the UAS has reason to believe that the scheme of the Request-URI has been improperly modified in transit, the UA SHOULD notify its user of a potential security breach.

As a further measure to prevent downgrade attacks, entities that accept only SIPS requests MAY also refuse connections on insecure ports.

End users will undoubtedly discern the difference between SIPS and SIP URIs, and they may manually edit them in response to stimuli. This can either benefit or degrade security. For example, if an attacker corrupts a DNS cache, inserting a fake record set that effectively removes all SIPS records for a proxy server, then any SIPS requests that traverse this proxy server may fail. When a user, however, sees that repeated calls to a SIPS AOR are failing, they could on some devices manually convert the scheme from SIPS to SIP and retry. Of course, there are some safeguards against this (if the destination UA is truly paranoid it could refuse all non-SIPS requests), but it is a limitation worth noting. On the bright side, users might also divine that 'SIPS' would be valid even when they are presented only with a SIP URI.

26.5 Privacy

SIP messages frequently contain sensitive information about their senders - not just what they have to say, but with whom they communicate, when they communicate and for how long, and from where they participate in sessions. Many applications and their users require that this sort of private information be hidden from any parties that do not need to know it.

Note that there are also less direct ways in which private information can be divulged. If a user or service chooses to be reachable at an address that is guessable from the person's name and organizational affiliation (which describes most addresses-of-record), the traditional method of ensuring privacy by having an unlisted "phone number" is compromised. A user location service can infringe on the privacy of the recipient of a session invitation by divulging their specific whereabouts to the caller; an implementation consequently SHOULD be able to restrict, on a per-user basis, what kind of location and availability information is given out to certain classes of callers. This is a whole class of problem that is expected to be studied further in ongoing SIP work.

In some cases, users may want to conceal personal information in header fields that convey identity. This can apply not only to the From and related headers representing the originator of the request, but also the To - it may not be appropriate to convey to the final destination a speed-dialing nickname, or an unexpanded identifier for a group of targets, either of which would be removed from the Request-URI as the request is routed, but not changed in the To

header field if the two were initially identical. Thus it MAY be desirable for privacy reasons to create a To header field that differs from the Request-URI.

27 IANA Considerations

All method names, header field names, status codes, and option tags used in SIP applications are registered with IANA through instructions in an IANA Considerations section in an RFC.

The specification instructs the IANA to create four new sub-registries under http://www.iana.org/assignments/sip-parameters: Option Tags, Warning Codes (warn-codes), Methods and Response Codes, added to the sub-registry of Header Fields that is already present there.

27.1 Option Tags

This specification establishes the Option Tags sub-registry under http://www.iana.org/assignments/sip-parameters.

Option tags are used in header fields such as Require, Supported, Proxy-Require, and Unsupported in support of SIP compatibility mechanisms for extensions (Section 19.2). The option tag itself is a string that is associated with a particular SIP option (that is, an extension). It identifies the option to SIP endpoints.

Option tags are registered by the IANA when they are published in standards track RFCs. The IANA Considerations section of the RFC must include the following information, which appears in the IANA registry along with the RFC number of the publication.

- Name of the option tag. The name MAY be of any length, but SHOULD be no more than twenty characters long. The name MUST consist of alphanum (Section 25) characters only.

- Descriptive text that describes the extension.

27.2 Warn-Codes

This specification establishes the Warn-codes sub-registry under http://www.iana.org/assignments/sip-parameters and initiates its population with the warn-codes listed in Section 20.43. Additional warn-codes are registered by RFC publication.

The descriptive text for the table of warn-codes is:

Warning codes provide information supplemental to the status code in SIP response messages when the failure of the transaction results from a Session Description Protocol (SDP) (RFC 2327 [1]) problem.

The "warn-code" consists of three digits. A first digit of "3" indicates warnings specific to SIP. Until a future specification describes uses of warn-codes other than 3xx, only 3xx warn-codes may be registered.

Warnings 300 through 329 are reserved for indicating problems with keywords in the session description, 330 through 339 are warnings related to basic network services requested in the session description, 370 through 379 are warnings related to quantitative QoS parameters requested in the session description, and 390 through 399 are miscellaneous warnings that do not fall into one of the above categories.

27.3 Header Field Names

This obsoletes the IANA instructions about the header sub-registry under http://www.iana.org/assignments/sip-parameters.

The following information needs to be provided in an RFC publication in order to register a new header field name:

- The RFC number in which the header is registered;
- the name of the header field being registered;
- a compact form version for that header field, if one is defined;

Some common and widely used header fields MAY be assigned one-letter compact forms (Section 7.3.3). Compact forms can only be assigned after SIP working group review, followed by RFC publication.

27.4 Method and Response Codes

This specification establishes the Method and Response-Code sub-registries under http://www.iana.org/assignments/sip-parameters and initiates their population as follows. The initial Methods table is:

INVITE	[RFC3261]
ACK	[RFC3261]
BYE	[RFC3261]
CANCEL	[RFC3261]
REGISTER	[RFC3261]
OPTIONS	[RFC3261]
INFO	[RFC2976]

The response code table is initially populated from Section 21, the portions labeled Informational, Success, Redirection, Client-Error, Server-Error, and Global-Failure. The table has the following format:

```
Type (e.g., Informational)
      Number    Default Reason Phrase          [RFC3261]
```

The following information needs to be provided in an RFC publication in order to register a new response code or method:

- The RFC number in which the method or response code is registered;
- the number of the response code or name of the method being registered;
- the default reason phrase for that response code, if applicable;

27.5 The "message/sip" MIME type.

This document registers the "message/sip" MIME media type in order to allow SIP messages to be tunneled as bodies within SIP, primarily for end-to-end security purposes. This media type is defined by the following information:

```
Media type name: message
Media subtype name: sip
Required parameters: none

Optional parameters: version
   version: The SIP-Version number of the enclosed message (e.g.,
   "2.0"). If not present, the version defaults to "2.0".
Encoding scheme: SIP messages consist of an 8-bit header
   optionally followed by a binary MIME data object. As such, SIP
   messages must be treated as binary. Under normal circumstances
   SIP messages are transported over binary-capable transports, no
   special encodings are needed.
```

Security considerations: see below
Motivation and examples of this usage as a security mechanism in concert with S/MIME are given in 23.4.

27.6 New Content-Disposition Parameter Registrations

This document also registers four new Content-Disposition header "disposition-types": alert, icon, session and render. The authors request that these values be recorded in the IANA registry for Content-Dispositions.

Descriptions of these "disposition-types", including motivation and examples, are given in Section 20.11.

Short descriptions suitable for the IANA registry are:

alert	the body is a custom ring tone to alert the user
icon	the body is displayed as an icon to the user
render	the body should be displayed to the user
session	the body describes a communications session, for example, as RFC 2327 SDP body

28 Changes From RFC 2543

This RFC revises RFC 2543. It is mostly backwards compatible with RFC 2543. The changes described here fix many errors discovered in RFC 2543 and provide information on scenarios not detailed in RFC 2543. The protocol has been presented in a more cleanly layered model here.

We break the differences into functional behavior that is a substantial change from RFC 2543, which has impact on interoperability or correct operation in some cases, and functional behavior that is different from RFC 2543 but not a potential source of interoperability problems. There have been countless clarifications as well, which are not documented here.

28.1 Major Functional Changes

- When a UAC wishes to terminate a call before it has been answered, it sends CANCEL. If the original INVITE still returns a 2xx, the UAC then sends BYE. BYE can only be sent on an existing call leg (now called a dialog in this RFC), whereas it could be sent at any time in RFC 2543.

- The SIP BNF was converted to be RFC 2234 compliant.

- SIP URL BNF was made more general, allowing a greater set of characters in the user part. Furthermore, comparison rules were simplified to be primarily case-insensitive, and detailed handling of comparison in the presence of parameters was described. The most substantial change is that a URI with a parameter with the default value does not match a URI without that parameter.

- Removed Via hiding. It had serious trust issues, since it relied on the next hop to perform the obfuscation process. Instead, Via hiding can be done as a local implementation choice in stateful proxies, and thus is no longer documented.

- In RFC 2543, CANCEL and INVITE transactions were intermingled. They are separated now. When a user sends an INVITE and then a CANCEL, the INVITE transaction still terminates normally. A UAS needs to respond to the original INVITE request with a 487 response.

- Similarly, CANCEL and BYE transactions were intermingled; RFC 2543 allowed the UAS not to send a response to INVITE when a BYE was received. That is disallowed here. The original INVITE needs a response.

- In RFC 2543, UAs needed to support only UDP. In this RFC, UAs need to support both UDP and TCP.

- In RFC 2543, a forking proxy only passed up one challenge from downstream elements in the event of multiple challenges. In this RFC, proxies are supposed to collect all challenges and place them into the forwarded response.

- In Digest credentials, the URI needs to be quoted; this is unclear from RFC 2617 and RFC 2069 which are both inconsistent on it.

- SDP processing has been split off into a separate specification [13], and more fully specified as a formal offer/answer exchange process that is effectively tunneled through SIP. SDP is allowed in INVITE/200 or 200/ACK for baseline SIP implementations; RFC 2543 alluded to the ability to use it in INVITE, 200, and ACK in a single transaction, but this was not well specified. More complex SDP usages are allowed in extensions.

- Added full support for IPv6 in URIs and in the Via header field. Support for IPv6 in Via has required that its header field parameters allow the square bracket and colon characters. These characters were previously not permitted. In theory, this could cause interop problems with older implementations. However, we have observed that most implementations accept any non-control ASCII character in these parameters.

- DNS SRV procedure is now documented in a separate specification [4]. This procedure uses both SRV and NAPTR resource records and no longer combines data from across SRV records as described in RFC 2543.

- Loop detection has been made optional, supplanted by a mandatory usage of Max-Forwards. The loop detection procedure in RFC 2543 had a serious bug which would report "spirals" as an error condition when it was not. The optional loop detection procedure is more fully and correctly specified here.

- Usage of tags is now mandatory (they were optional in RFC 2543), as they are now the fundamental building blocks of dialog identification.

- Added the Supported header field, allowing for clients to indicate what extensions are supported to a server, which can apply those extensions to the response, and indicate their usage with a Require in the response.

- Extension parameters were missing from the BNF for several header fields, and they have been added.

- Handling of Route and Record-Route construction was very underspecified in RFC 2543, and also not the right approach. It has been substantially reworked in this specification (and made vastly simpler), and this is arguably the largest change. Backwards compatibility is still provided for deployments that do not use "pre-loaded routes", where the initial request has a set of Route header field values obtained in some way outside of Record-Route. In those situations, the new mechanism is not interoperable.

- In RFC 2543, lines in a message could be terminated with CR, LF, or CRLF. This specification only allows CRLF.

- Usage of Route in CANCEL and ACK was not well defined in RFC 2543. It is now well specified; if a request had a Route header field, its CANCEL or ACK for a non-2xx response to the request need to carry the same Route header field values. ACKs for 2xx responses use the Route values learned from the Record-Route of the 2xx responses.

- RFC 2543 allowed multiple requests in a single UDP packet. This usage has been removed.

- Usage of absolute time in the Expires header field and parameter has been removed. It caused interoperability problems in elements that were not time synchronized, a common occurrence. Relative times are used instead.

- The branch parameter of the Via header field value is now mandatory for all elements to use. It now plays the role of a unique transaction identifier. This avoids the complex and bug-laden transaction identification rules from RFC 2543. A magic cookie is used in the parameter value to determine if the previous hop has made the parameter globally unique, and comparison falls back to the old rules when it is not present. Thus, interoperability is assured.

- In RFC 2543, closure of a TCP connection was made equivalent to a CANCEL. This was nearly impossible to implement (and wrong) for TCP connections between proxies. This has been eliminated, so that there is no coupling between TCP connection state and SIP processing.

- RFC 2543 was silent on whether a UA could initiate a new transaction to a peer while another was in progress. That is now specified here. It is allowed for non-INVITE requests, disallowed for INVITE.

- PGP was removed. It was not sufficiently specified, and not compatible with the more complete PGP MIME. It was replaced with S/MIME.

- Added the "sips" URI scheme for end-to-end TLS. This scheme is not backwards compatible with RFC 2543. Existing elements that receive a request with a SIPS URI scheme in the Request-URI will likely reject the request. This is actually a feature; it ensures that a call to a SIPS URI is only delivered if all path hops can be secured.

- Additional security features were added with TLS, and these are described in a much larger and complete security considerations section.

- In RFC 2543, a proxy was not required to forward provisional responses from 101 to 199 upstream. This was changed to MUST. This is important, since many subsequent features depend on delivery of all provisional responses from 101 to 199.

- Little was said about the 503 response code in RFC 2543. It has since found substantial use in indicating failure or overload conditions in proxies. This requires somewhat special treatment. Specifically, receipt of a 503 should trigger an attempt to contact the next element in the result of a DNS SRV lookup. Also, 503 response is only forwarded upstream by a proxy under certain conditions.

- RFC 2543 defined, but did no sufficiently specify, a mechanism for UA authentication of a server. That has been removed. Instead, the mutual authentication procedures of RFC 2617 are allowed.

- A UA cannot send a BYE for a call until it has received an ACK for the initial INVITE. This was allowed in RFC 2543 but leads to a potential race condition.

- A UA or proxy cannot send CANCEL for a transaction until it gets a provisional response for the request. This was allowed in RFC 2543 but leads to potential race conditions.

- The action parameter in registrations has been deprecated. It was insufficient for any useful services, and caused conflicts when application processing was applied in proxies.

- RFC 2543 had a number of special cases for multicast. For example, certain responses were suppressed, timers were adjusted, and so on. Multicast now plays a more limited role, and the protocol operation is unaffected by usage of multicast as opposed to unicast. The limitations as a result of that are documented.

- Basic authentication has been removed entirely and its usage forbidden.

- Proxies no longer forward a 6xx immediately on receiving it. Instead, they CANCEL pending branches immediately. This avoids a potential race condition that would result in a UAC getting a 6xx followed by a 2xx. In all cases except this race condition, the result will be the same - the 6xx is forwarded upstream.

- RFC 2543 did not address the problem of request merging. This occurs when a request forks at a proxy and later rejoins at an element. Handling of merging is done only at a UA, and procedures are defined for rejecting all but the first request.

28.2 Minor Functional Changes

- Added the Alert-Info, Error-Info, and Call-Info header fields for optional content presentation to users.

- Added the Content-Language, Content-Disposition and MIME-Version header fields.

- Added a "glare handling" mechanism to deal with the case where both parties send each other a re-INVITE simultaneously. It uses the new 491 (Request Pending) error code.

- Added the In-Reply-To and Reply-To header fields for supporting the return of missed calls or messages at a later time.

- Added TLS and SCTP as valid SIP transports.

- There were a variety of mechanisms described for handling failures at any time during a call; those are now generally unified. BYE is sent to terminate.

- RFC 2543 mandated retransmission of INVITE responses over TCP, but noted it was really only needed for 2xx. That was an artifact of insufficient protocol layering. With a more coherent transaction layer defined here, that is no longer needed. Only 2xx responses to INVITEs are retransmitted over TCP.

- Client and server transaction machines are now driven based on timeouts rather than retransmit counts. This allows the state machines to be properly specified for TCP and UDP.

- The Date header field is used in REGISTER responses to provide a simple means for auto-configuration of dates in user agents.

- Allowed a registrar to reject registrations with expirations that are too short in duration. Defined the 423 response code and the Min-Expires for this purpose.

29 Normative References

[1] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[4] Rosenberg, J. and H. Schulzrinne, "SIP: Locating SIP Servers", RFC 3263, June 2002.

[5] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[6] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[7] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[9] Vaha-Sipila, A., "URLs for Telephone Calls", RFC 2806, April 2000.

[10] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[11] Freed, F. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, November 1996.

[12] Eastlake, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.

[13] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", RFC 3264, June 2002.

[14] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[15] Postel, J., "DoD Standard Transmission Control Protocol", RFC 761, January 1980.

[16] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[17] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[18] Troost, R., Dorner, S. and K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, August 1997.

[19] Zimmerer, E., Peterson, J., Vemuri, A., Ong, L., Audet, F., Watson, M. and M. Zonoun, "MIME media types for ISUP and QSIG Objects", RFC 3204, December 2001.

[20] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[21] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

[22] Galvin, J., Murphy, S., Crocker, S. and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, October 1995.

[23] Housley, R., "Cryptographic Message Syntax", RFC 2630, June 1999.

[24] Ramsdell B., "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

[25] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[26] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

30 Informative References

[27] R. Pandya, "Emerging mobile and personal communication systems," IEEE Communications Magazine, Vol. 33, pp. 44--52, June 1995.

[28] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[29] Schulzrinne, H., Rao, R. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[30] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. and J. Segers, "Megaco Protocol Version 1.0", RFC 3015, November 2000.

[31] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[32] Hoffman, P., Masinter, L. and J. Zawinski, "The mailto URL scheme", RFC 2368, July 1998.

[33] E. M. Schooler, "A multicast user directory service for synchronous rendezvous," Master's Thesis CS-TR-96-18, Department of Computer Science, California Institute of Technology, Pasadena, California, Aug. 1996.

[34] Donovan, S., "The SIP INFO Method", RFC 2976, October 2000.

[35] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[36] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, September 1998.

[37] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

[38] Palme, J., "Common Internet Message Headers", RFC 2076, February 1997.

[39] Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E. and L. Stewart, "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997.

[40] Johnston, A., Donovan, S., Sparks, R., Cunningham, C., Willis, D., Rosenberg, J., Summers, K. and H. Schulzrinne, "SIP Call Flow Examples", Work in Progress.

[41] E. M. Schooler, "Case study: multimedia conference control in a packet-switched teleconferencing system," Journal of Internetworking: Research and Experience, Vol. 4, pp. 99--120, June 1993. ISI reprint series ISI/RS-93-359.

[42] H. Schulzrinne, "Personal mobility for multimedia services in the Internet," in European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), (Berlin, Germany), Mar. 1996.

[43] Floyd, S., "Congestion Control Principles", RFC 2914, September 2000.

A Table of Timer Values

Table 4 summarizes the meaning and defaults of the various timers used by this specification.

Timer	Value	Section	Meaning
T1	500ms default	Section 17.1.1.1	RTT Estimate
T2	4s	Section 17.1.2.2	The maximum retransmit interval for non-INVITE requests and INVITE responses
T4	5s	Section 17.1.2.2	Maximum duration a message will remain in the network
Timer A	initially T1	Section 17.1.1.2	INVITE request retransmit interval, for UDP only
Timer B	64*T1	Section 17.1.1.2	INVITE transaction timeout timer
Timer C	> 3min	Section 16.6 bullet 11	proxy INVITE transaction timeout
Timer D	> 32s for UDP 0s for TCP/SCTP	Section 17.1.1.2	Wait time for response retransmits
Timer E	initially T1	Section 17.1.2.2	non-INVITE request retransmit interval, UDP only
Timer F	64*T1	Section 17.1.2.2	non-INVITE transaction timeout timer
Timer G	initially T1	Section 17.2.1	INVITE response retransmit interval
Timer H	64*T1	Section 17.2.1	Wait time for ACK receipt
Timer I	T4 for UDP 0s for TCP/SCTP	Section 17.2.1	Wait time for ACK retransmits
Timer J	64*T1 for UDP 0s for TCP/SCTP	Section 17.2.2	Wait time for non-INVITE request retransmits
Timer K	T4 for UDP 0s for TCP/SCTP	Section 17.1.2.2	Wait time for response retransmits

Table 4: Summary of timers

Acknowledgments

We wish to thank the members of the IETF MMUSIC and SIP WGs for their comments and suggestions. Detailed comments were provided by Ofir Arkin, Brian Bidulock, Jim Buller, Neil Deason, Dave Devanathan, Keith Drage, Bill Fenner, Cedric Fluckiger, Yaron Goland, John Hearty, Bernie Hoeneisen, Jo Hornsby, Phil Hoffer, Christian Huitema, Hisham Khartabil, Jean Jervis, Gadi Karmi, Peter Kjellerstedt, Anders Kristensen, Jonathan Lennox, Gethin Liddell, Allison Mankin, William Marshall, Rohan Mahy, Keith Moore, Vern Paxson, Bob Penfield, Moshe J. Sambol, Chip Sharp, Igor Slepchin, Eric Tremblay, and Rick Workman.

Brian Rosen provided the compiled BNF.

Jean Mahoney provided technical writing assistance.

This work is based, inter alia, on [41,42].

Authors' Addresses

Authors addresses are listed alphabetically for the editors, the writers, and then the original authors of RFC 2543. All listed authors actively contributed large amounts of text to this document.

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Ave
East Hanover, NJ 07936
USA

EMail: jdrosen@dynamicsoft.com

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA

EMail: schulzrinne@cs.columbia.edu

Gonzalo Camarillo
Ericsson
Advanced Signalling Research Lab.
FIN-02420 Jorvas
Finland

EMail: Gonzalo.Camarillo@ericsson.com

Alan Johnston
WorldCom
100 South 4th Street
St. Louis, MO 63102
USA

EMail: alan.johnston@wcom.com

Jon Peterson
NeuStar, Inc
1800 Sutter Street, Suite 570
Concord, CA 94520
USA

EMail: jon.peterson@neustar.com

Robert Sparks
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: rsparks@dynamicsoft.com

Mark Handley
International Computer Science Institute
1947 Center St, Suite 600
Berkeley, CA 94704
USA

EMail: mjh@icir.org

Eve Schooler
AT&T Labs-Research
75 Willow Road
Menlo Park, CA 94025
USA

EMail: schooler@research.att.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Request for Comments: 3666
BCP: 76
Category: Best Current Practice

A. Johnston
MCI
S. Donovan
R. Sparks
C. Cunningham
dynamicsoft
K. Summers
Sonus

December 2003

Session Initiation Protocol (SIP) Public Switched Telephone Network (PSTN) Call Flows

Status of this Memo

Copyright Notice

Abstract

This document contains best current practice examples of Session Initiation Protocol (SIP) call flows showing interworking with the Public Switched Telephone Network (PSTN). Elements in these call flows include SIP User Agents, SIP Proxy Servers, and PSTN Gateways. Scenarios include SIP to PSTN, PSTN to SIP, and PSTN to PSTN via SIP. PSTN telephony protocols are illustrated using ISDN (Integrated Services Digital Network), ISUP (ISDN User Part), and FGB (Feature Group B) circuit associated signaling. PSTN calls are illustrated using global telephone numbers from the PSTN and private extensions served on by a PBX (Private Branch Exchange). Call flow diagrams and message details are shown.

Table of Contents

1. Overview

The call flows shown in this document were developed in the design of a SIP IP communications network. They represent an example of a minimum set of functionality.

It is the hope of the authors that this document will be useful for SIP implementers, designers, and protocol researchers alike and will help further the goal of a standard implementation of RFC 3261 [2]. These flows represent carefully checked and working group reviewed scenarios of the most common SIP/PSTN interworking examples as a companion to the specifications.

These call flows are based on the current version 2.0 of SIP in RFC 3261 [2] with SDP usage described in RFC 3264 [3]. Other RFCs also comprise the SIP standard but are not used in this set of basic call flows. The SIP/ISUP mapping is based on RFC 3398 [4].

Various PSTN signaling protocols are illustrated in this document: ISDN (Integrated Services Digital Network), ISUP (ISDN User Part) and FGB (Feature Group B) circuit associated signaling. This document shows mainly ANSI ISUP due to its practical origins. However, as used in this document, the usage is virtually identical to the ITU-T International ISUP used as the reference in [4].

Basic SIP call flow examples are contained in a companion document, RFC 3665 [10].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [1].

1.1. General Assumptions

A number of architecture, network, and protocol assumptions underlie the call flows in this document. Note that these assumptions are not requirements. They are outlined in this section so that they may be taken into consideration and to aid in the understanding of the call flow examples.

The authentication of SIP User Agents in these example call flows is performed using HTTP Digest as defined in [3] and [5].

Some Proxy Servers in these call flows insert Record-Route headers into requests to ensure that they are in the signaling path for future message exchanges.

These flows show TLS, TCP, and UDP for transport. SCTP could also be used. See the discussion in RFC 3261 [2] for details on the transport issues for SIP.

The SIP Proxy Server has access to a Location Service and other databases. Information present in the Request-URI and the context (From header) is sufficient to determine to which proxy or gateway the message should be routed. In most cases, a primary and secondary route will be determined in case of a Proxy or Gateway failure downstream.

Gateways provide tones (ringing, busy, etc) and announcements to the PSTN side based on SIP response messages, or pass along audio in-band tones (ringing, busy tone, etc.) in an early media stream to the SIP side.

The interactions between the Proxy and Gateway can be summarized as follows:

- The SIP Proxy Server performs digit analysis and lookup and locates the correct gateway.

- The SIP Proxy Server performs gateway location based on primary and secondary routing.

Telephone numbers are usually represented as SIP URIs. Note that an alternative is the use of the tel URI [6].

This document shows typical examples of SIP/ISUP interworking. Although in the spirit of the SIP-T framework [7], these examples do not represent a complete implementation of the framework. The examples here represent more of a minimal set of examples for very basic SIP to ISUP interworking, rather than the more complex goal of ISUP transparency. In particular, there are NO examples of encapsulated ISUP in this document. If present, these messages would show S/MIME encryption due to the sensitive nature of this information, as discussed in the SIP-T Framework security considerations section. (Note - RFC 3204 [8] contains an example of an INVITE with encapsulated ISUP.) See the Security Considerations section for a more detailed discussion on the security of these call flows.

In ISUP, the Calling Party Number is abbreviated as CgPN and the Called Party Number is abbreviated as CdPN. Other abbreviations include Numbering Plan Indicator (NPI) and Nature of Address (NOA).

1.2. Legend for Message Flows

Dashed lines (---) represent signaling messages that are mandatory to the call scenario. These messages can be SIP or PSTN signaling. The arrow indicates the direction of message flow.

Double dashed lines (===) represent media paths between network elements.

Messages with parentheses around their name represent optional messages.

Messages are identified in the Figures as F1, F2, etc. This references the message details in the list that follows the Figure. Comments in the message details are shown in the following form:

```
/* Comments. */
```

1.3. SIP Protocol Assumptions

This document does not prescribe the flows precisely as they are shown, but rather the flows illustrate the principles for best practice. They are best practices usages (orderings, syntax, selection of features for the purpose, handling of error) of SIP methods, headers and parameters. IMPORTANT: The exact flows here must not be copied as is by an implementer due to specific incorrect characteristics that were introduced into the document for convenience and are listed below. To sum up, the SIP/PSTN call flows represent well-reviewed examples of SIP usage, which are best common practice according to IETF consensus.

For simplicity in reading and editing the document, there are a number of differences between some of the examples and actual SIP messages. For example, the SIP Digest responses are not actual MD5 encodings. Call-IDs are often repeated, and CSeq counts often begin at 1. Header fields are usually shown in the same order. Usually only the minimum required header field set is shown, others that would normally be present, such as Accept, Supported, Allow, etc. are not shown.

Actors:

Element	Display Name	URI	IP Address
User Agent	Alice	sip:alice@a.example.com	192.0.2.101
User Agent	Bob	sip:bob@b.example.com	192.0.2.200
Proxy Server		sip:ss1.a.example.com	192.0.2.111
User Agent (Gateway)		sip:gw1.a.example.com	192.0.2.201
User Agent (Gateway)		sip:gw2.a.example.com	192.0.2.202
User Agent (Gateway)		sip:gw3.a.example.com	192.0.2.203
User Agent (Gateway)		sip:ngw1.a.example.com	192.0.2.103
User Agent (Gateway)		sip:ngw2.a.example.com	192.0.2.102

Note that NGW 1 and NGW 2 also have device URIs (Contacts) of sip:ngw1@a.example.com and sip:ngw2@a.example.com which resolve to the Proxy Server sip:ss1.wcom.com using DNS SRV records.

2. SIP to PSTN Dialing

In the following scenarios, Alice (sip:alice@a.example.com) is a SIP phone or other SIP-enabled device. Bob is reachable via the PSTN at global telephone number +19725552222. Alice places a call to Bob through a Proxy Server, Proxy 1, and a Network Gateway. In other scenarios, Alice places calls to Carol, who is served via a PBX (Private Branch Exchange) and is identified by a private extension 444-3333, or global number +1-918-555-3333. Note that Alice uses his/her global telephone number +1-314-555-1111 in the From header in the INVITE messages. This then gives the Gateway the option of using this header to populate the calling party identification field in subsequent signaling. Left open is the issue of how the Gateway can determine the accuracy of the telephone number which is necessary before passing it as a valid calling party number in the PSTN.

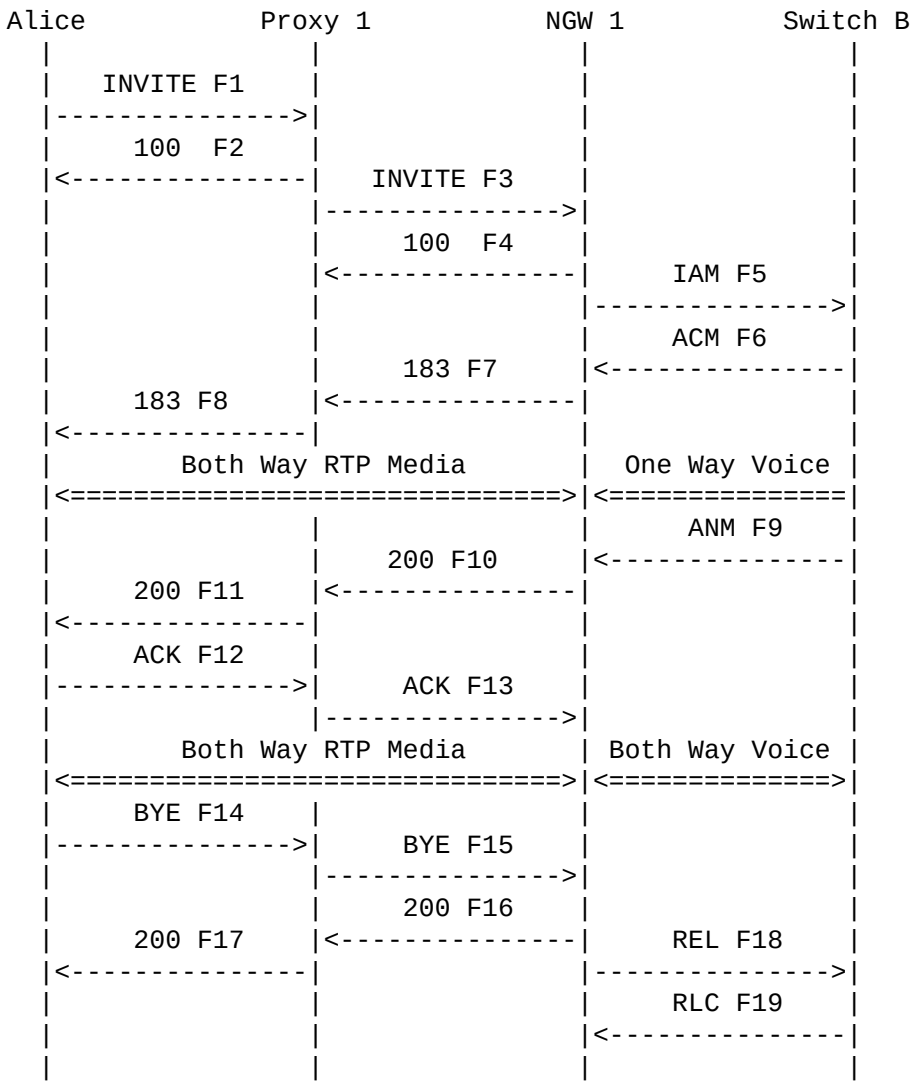
In these scenarios, Alice is a SIP phone or other SIP-enabled device. Alice places a call to Bob in the PSTN or Carol on a PBX through a Proxy Server and a Gateway.

In the failure scenarios, the call does not complete. In some cases however, a media stream is still setup. This is due to the fact that some failures in dialing to the PSTN result in in-band tones (busy, reorder tones or announcements - "The number you have dialed has changed. The new number is..."). The 183 Session Progress response containing SDP media information is used to setup this early media path so that the caller Alice knows the final disposition of the call.

The media stream is either terminated by the caller after the tone or announcement has been heard and understood, or by the Gateway after a timer expires.

In other failure scenarios, a SS7 Release with Cause Code is mapped to a SIP response. In these scenarios, the early media path is not used, but the actual failure code is conveyed to the caller by the SIP User Agent Client.

2.1. Successful SIP to ISUP PSTN call



Alice dials the globalized E.164 number +19725552222 to reach Bob. Note that A might have only dialed the last 7 digits, or some other dialing plan. It is assumed that the SIP User Agent Client converts the digits into a global number and puts them into a SIP URI. Note that tel URIs could be used instead of SIP URIs.

Alice could use either their SIP address (sip:alice@a.example.com) or SIP telephone number (sip:+13145551111@ss1.a.example.com;user=phone) in the From header. In this example, the telephone number is included, and it is shown as being passed as calling party identification through the Network Gateway (NGW 1) to Bob (F5). Note

that for this number to be passed into the SS7 network, it would have to be somehow verified for accuracy.

In this scenario, Bob answers the call, then Alice disconnects the call. Signaling between NGW 1 and Bob's telephone switch is ANSI ISUP. For the details of SIP to ISUP mapping, refer to [4].

In this flow, notice that the Contact returned by NGW 1 in messages F7-11 is sip:ngw1@a.example.com. This is because NGW 1 only accepts SIP messages that come through Proxy 1 - any direct signaling will be ignored. Since this Contact URI may be used outside of this dialog and must be routable (Section 8.1.1.8 in RFC 3261 [2]) the Contact URI for NGW 1 must resolve to Proxy 1. This Contact URI resolves via DNS to Proxy 1 (sip:ss1.a.example.com) which then resolves it to sip:ngw1.a.example.com which is the address of NGW 1.

This flow shows TCP transport.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com;transport=tcp>
Proxy-Authorization: Digest username="alice", realm="a.example.com",
 nonce="dc3a5ab25302aa931904ba7d88fa1cf5", opaque="",
 uri="sip:+19725552222@ss1.a.example.com;user=phone",
 response="ccdca50cb091d587421457305d097458c"
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
   F2 100 Trying Proxy 1 -> Alice

   SIP/2.0 100 Trying
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Content-Length: 0

   /* Proxy 1 uses a Location Service function to determine the gateway
   for terminating this call.  The call is forwarded to NGW 1.  Client
   for A prepares to receive data on port 49172 from the
   network.*/


   F3 INVITE Proxy 1 -> NGW 1

   INVITE sip:+19725552222@ngw1.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 154

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.a.example.com
   s=-
   c=IN IP4 client.a.example.com
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F4 100 Trying NGW 1 -> Proxy 1

   SIP/2.0 100 Trying
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
```

```
    ;received=192.0.2.111
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F5 IAM NGW 1 -> Bob

   IAM
   CdPN=972-555-2222,NPI=E.164,NOA=National
   CgPN=314-555-1111,NPI=E.164,NOA=National


   F6 ACM Bob -> NGW 1

   ACM


   F7 183 Session Progress NGW 1 -> Proxy 1

   SIP/2.0 183 Session Progress
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 ngw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* NGW 1 sends PSTN audio (ringing) in the RTP path to A */
```

```
   F8 183 Session Progress Proxy 1 -> Alice

   SIP/2.0 183 Session Progress
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 ngw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F9 ANM Bob -> NGW 1

   ANM


   F10 200 OK NGW 1 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
```

```
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 gw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F11 200 OK Proxy 1 -> Alice

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 ngw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F12 ACK Alice -> Proxy 1

   ACK sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 ACK
```

```
   Content-Length: 0


   F13 ACK Proxy 1 -> NGW 1

   ACK sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 ACK
   Content-Length: 0

   /* Alice Hangs Up with Bob. */


   F14 BYE Alice -> Proxy 1

   BYE sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 BYE
   Content-Length: 0


   F15 BYE Proxy 1 -> NGW 1

   BYE sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
```

```
CSeq: 2 BYE
Content-Length: 0
```

F16 200 OK NGW 1 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 BYE
Content-Length: 0
```

F17 200 OK Proxy 1 -> A

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 BYE
Content-Length: 0
```

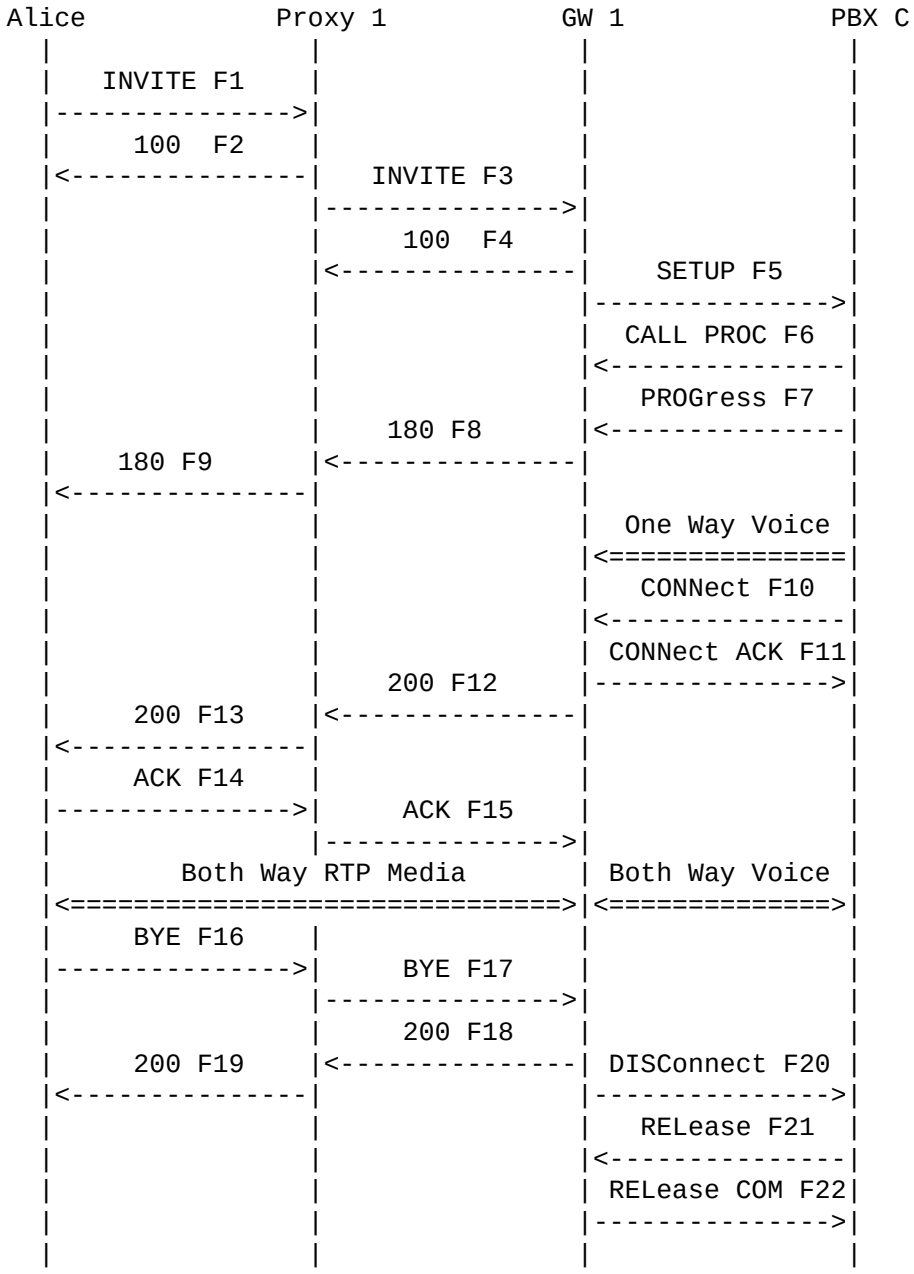
F18 REL NGW 1 -> B

```
REL
CauseCode=16 Normal
```

F19 RLC B -> NGW 1

```
RLC
```

2.2. Successful SIP to ISDN PBX call



Alice is a SIP device while Carol is connected via a Gateway (GW 1) to a PBX. The PBX connection is via a ISDN trunk group. Alice dials Carol's telephone number (918-555-3333) which is globalized and put into a SIP URI.

The host portion of the Request-URI in the INVITE F3 is used to identify the context (customer, trunk group, or line) in which the private number 444-3333 is valid. Otherwise, this INVITE message could get forwarded by GW 1 and the context of the digits could become lost and the call unroutable.

Proxy 1 looks up the telephone number and locates the gateway that serves Carol. Carol is identified by its extension (444-3333) in the Request-URI sent to GW 1.

Note that the Contact URI for GW 1, as used in messages F8, F9, F12, and F13, is sips:4443333@gw1.a.example.com, which resolves directly to the gateway.

This flow shows the use of Secure SIP (sips) URIs.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sips:+19185553333@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sips:alice@client.a.example.com>
Proxy-Authorization: Digest username="alice",
 realm="a.example.com", nonce="qo0dc3a5ab22aa931904badfa1cf5j9h",
 opaque="", uri="sips:+19185553333@ss1.a.example.com;user=phone",
 response="6c792f5c9fa360358b93c7fb826bf550"
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F2 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
```

```
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Content-Length: 0


F3 INVITE Proxy 1 -> GW 1

INVITE sips:4443333@gw1.a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sips:ss1.a.example.com;lr>
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sips:alice@client.a.example.com>
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 100 Trying GW -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Content-Length: 0
```

```
F5 SETUP GW 1 -> Carol

Protocol discriminator=Q.931
Message type=SETUP
Bearer capability: Information transfer capability=0 (Speech) or 16
(3.1 kHz audio)
Channel identification=Preferred or exclusive B-channel
Progress indicator=1 (Call is not end-to-end ISDN;further call
progress information may be available inband)
Called party number:
Type of number unknown
Digits=444-3333


F6 CALL PROCeeding Carol-> GW 1

Protocol discriminator=Q.931
Message type=CALL PROC
Channel identification=Exclusive B-channel


F7 PROGress Carol-> GW 1

Protocol discriminator=Q.931
Message type=PROG
Progress indicator=1 (Call is not end-to-end ISDN;further call
progress information may be available inband)


F8 180 Ringing GW 1 -> Proxy 1

SIP/2.0 180 Ringing
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sips:ss1.a.example.com;lr>
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sips:4443333@gw1.a.example.com>
Content-Length: 0
```

```
   F9 180 Ringing Proxy 1 -> Alice

   SIP/2.0 180 Ringing
   Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sips:ss1.a.example.com;lr>
   From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 INVITE
   Contact: <sips:4443333@gw1.a.example.com>
   Content-Length: 0


   F10 CONNect Carol-> GW 1

   Protocol discriminator=Q.931
   Message type=CONN


   F11 CONNect ACK GW 1 -> Carol

   Protocol discriminator=Q.931
   Message type=CONN ACK


   F12 200 OK GW 1 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sips:ss1.a.example.com;lr>
   From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 INVITE
   Contact: <sips:4443333@gw1.a.example.com>
   Content-Type: application/sdp
   Content-Length: 144

   v=0
   o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
```

```
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F13 200 OK Proxy 1 -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sips:ss1.a.example.com;lr>
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sips:4443333@gw1.a.example.com>
Content-Type: application/sdp
Content-Length: 144

v=0
o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F14 ACK Alice -> Proxy 1

ACK sips:4443333@gw1.a.example.com SIP/2.0
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sips:ss1.a.example.com;lr>
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 ACK
Content-Length: 0
```

F15 ACK Proxy 1 -> GW 1

```
ACK sips:4443333@gw1.a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 ACK
Content-Length: 0
```

/* Alice Hangs Up with Bob. */

F16 BYE Alice -> Proxy 1

```
BYE sips:4443333@gw1.a.example.com SIP/2.0
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sips:ss1.a.example.com;lr>
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 3 BYE
Content-Length: 0
```

F17 BYE Proxy 1 -> GW 1

```
BYE sips:4443333@gw1.a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 3 BYE
Content-Length: 0
```

F18 200 OK GW 1 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 3 BYE
Content-Length: 0
```

F19 200 OK Proxy 1 -> A

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS client.a.example.com:5061;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sips:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Carol <sips:+19185553333@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 3 BYE
Content-Length: 0
```

F20 DISConnect GW 1 -> Carol

```
Protocol discriminator=Q.931
Message type=DISC
Cause=16 (Normal clearing)
```

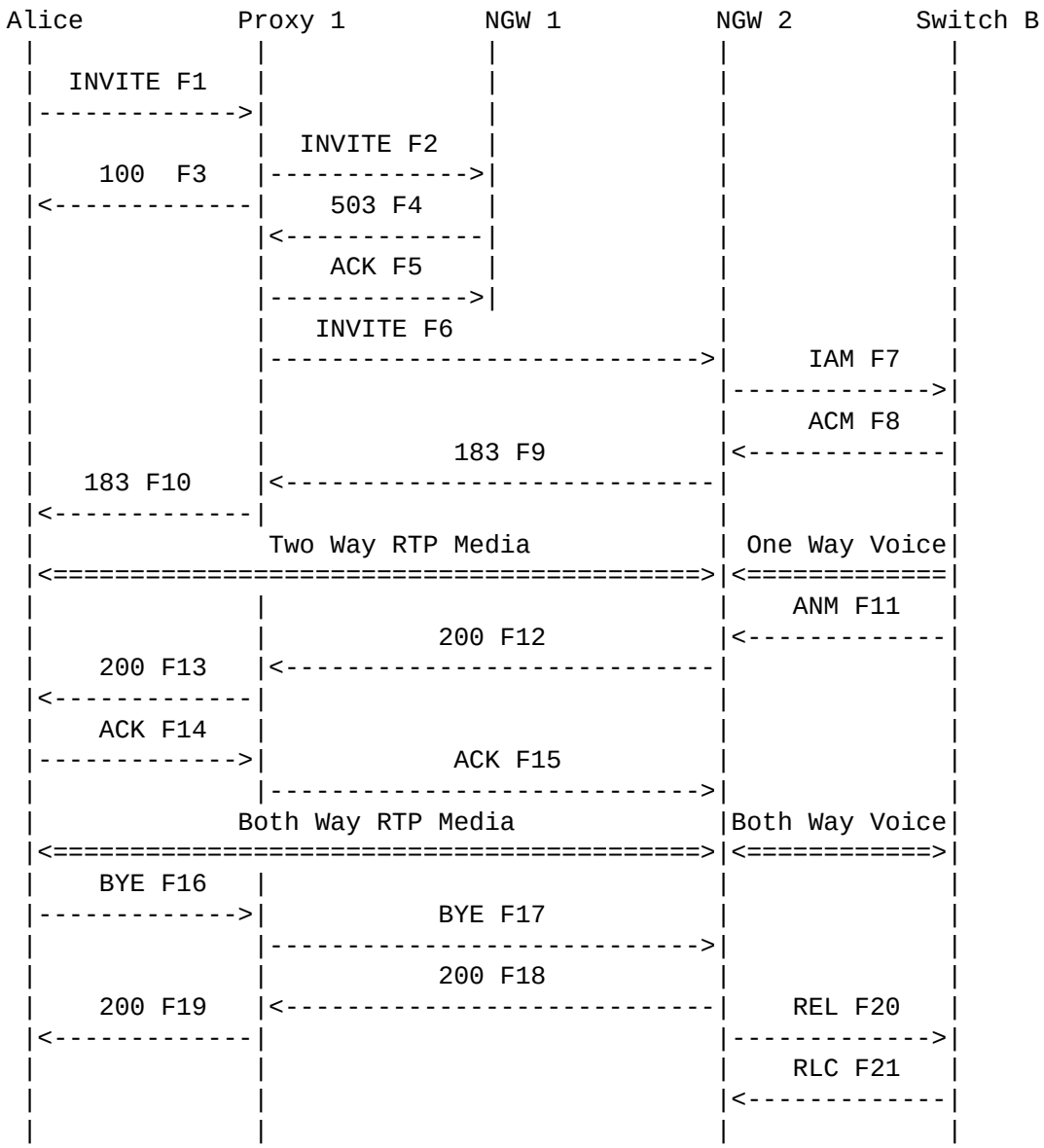
F21 RELease Carol-> GW 1

```
Protocol discriminator=Q.931
Message type=REL
```

F22 RELease COMplete GW 1 -> Carol

```
Protocol discriminator=Q.931
Message type=REL COM
```

2.3. Successful SIP to ISUP PSTN call with overflow



Alice calls Bob through Proxy 1. Proxy 1 tries to route to a Network Gateway NGW 1. NGW 1 is not available and responds with a 503 Service Unavailable (F4). The call is then routed to Network Gateway NGW 2. Bob answers the call. The call is terminated when Alice disconnects the call. NGW 2 and Bob's telephone switch use ANSI ISUP signaling.

NGW 2 also only accepts SIP messages that come through Proxy 1, so the Contact URI sip:ngw2@a.example.com is used in this flow.

This flow shows UDP transport.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com>
Proxy-Authorization: Digest username="alice",
 realm="a.example.com", nonce="b59311c3ba05b401cf80b2a2c5ac51b0",
 opaque="", uri="sip:+19725552222@ss1.a.example.com;user=phone",
 response="ba6ab44923fa2614b28e3e3957789ab0"
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine where B is
located.  Proxy 1 receives a primary route NGW 1 and a secondary
route NGW 2.  NGW 1 is tried first */
```

F2 INVITE Proxy 1 -> NGW 1

```
INVITE sip:+19725552222@ngw1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
```

```
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com>
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F3 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F4 503 Service Unavailable NGW 1 -> Proxy 1

```
SIP/2.0 503 Service Unavailable
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=123456789
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

```
   F5 ACK Proxy 1 -> NGW 1

   ACK sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Max-Forwards: 70
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com>;user=phone>
    ;tag=123456789
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 ACK
   Content-Length: 0

   /* Proxy 1 now tries secondary route to NGW 2 */


   F6 INVITE Proxy 1 -> NGW 2

   INVITE sip:+19725552222@ngw2.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.a.example.com>
   Content-Type: application/sdp
   Content-Length: 154

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.a.example.com
   s=-
   c=IN IP4 client.a.example.com
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F7 IAM NGW 2 -> Bob

   IAM
   CdPN=972-555-2222,NPI=E.164,NOA=National
   CgPN=314-555-1111,NPI=E.164,NOA=National
```

```
   F8 ACM Bob -> NGW 2

   ACM


   F9 183 Session Progress NGW 2 -> Proxy 1

   SIP/2.0 183 Session Progress
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw2@a.example.com>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw2.a.example.com
   s=-
   c=IN IP4 ngw2.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* RTP packets are sent by GW to A for audio (e.g. ring tone) */


   F10 183 Session Progress Proxy 1 -> Alice

   SIP/2.0 183 Session Progress
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw2@a.example.com>
   Content-Type: application/sdp
```

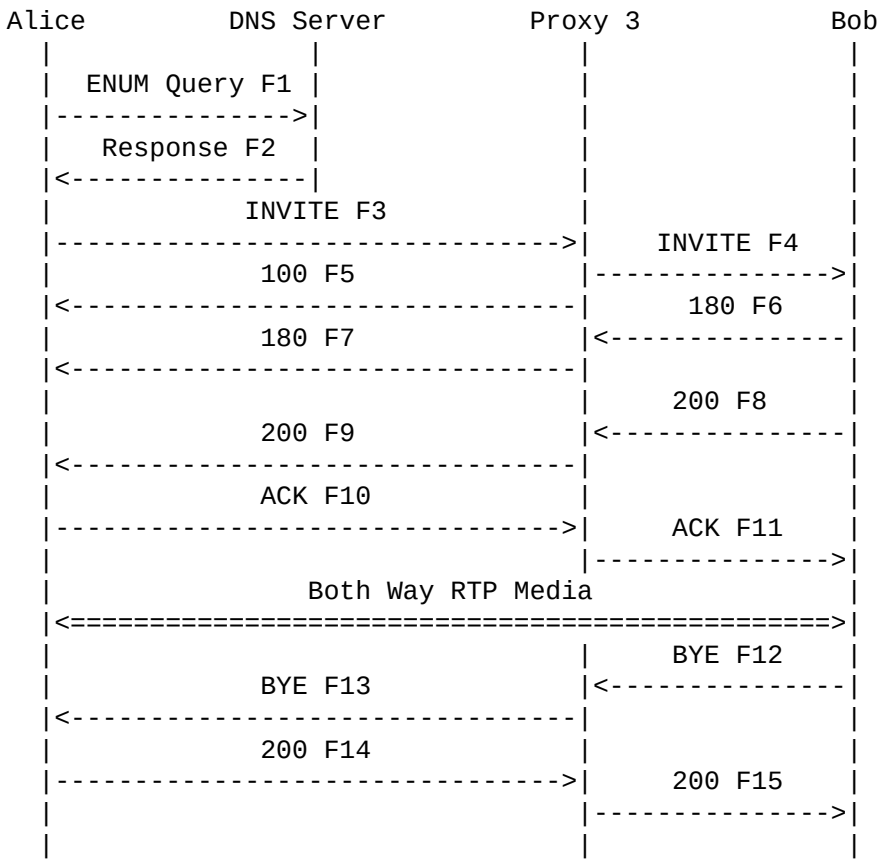
```
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw2.a.example.com
   s=-
   c=IN IP4 ngw2.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F11 ANM Bob -> NGW 2

   ANM


   F12 200 OK NGW 2 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw2@a.example.com>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw2.a.example.com
   s=-
   c=IN IP4 ngw2.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F13 200 OK Proxy 1 -> Alice

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
```

```
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw2@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw2.a.example.com
s=-
c=IN IP4 ngw2.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F14 ACK Alice -> Proxy 1

ACK sip:ngw2@a.example.com SIP/2.0
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0


F15 ACK Proxy 1 -> NGW 2

ACK sip:ngw2@a.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
```

```
   Content-Length: 0

   /* RTP streams are established between A and B(via the GW) */

   /* Alice Hangs Up with Bob. */


   F16 BYE Alice -> Proxy 1

   BYE sip:ngw2@a.example.com SIP/2.0
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   Route: <ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 BYE
   Content-Length: 0


   F17 BYE Proxy 1 -> NGW 2

   BYE sip:ngw2@a.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 BYE
   Content-Length: 0


   F18 200 OK NGW 2 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
```

```
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 BYE
Content-Length: 0


F19 200 OK Proxy 1 -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 BYE
Content-Length: 0


F20 REL NGW 2 -> B

REL
CauseCode=16 Normal


F21 RLC B -> NGW 2

RLC
```

2.4. Successful SIP to SIP using ENUM Query



In this scenario, Alice places a call to Bob by dialing Bob's telephone number (9725552222). Alice's UA converts the phone number to an E.164 number (+19725552222), and performs an ENUM query [9] on the E.164 number (2.2.2.2.5.5.5.2.7.9.1.e164.arpa), which returns a NAPTR record containing a SIP AOR URI for Bob (sip:+19725552222@b.example.com). As a result, Alice's UA sends an INVITE and the call completes over IP bypassing the PSTN.

The call is terminated when Bob sends a BYE message.

Message Details

F1 ENUM Query Alice -> DNS Server

2.2.2.2.5.5.5.2.7.9.1.e164.arpa

```
F2 ENUM NAPTR Set DNS Server -> Alice

$ORIGIN 2.2.2.2.5.5.5.2.7.9.1.e164.arpa.
      IN NAPTR 100 10 "u" "sip+E2U"
             "!^.*$!sip:+19725552222@b.example.com!".


F3 INVITE Alice -> Proxy 3

INVITE sip:+19725552222@b.example.com SIP/2.0
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+13145551111@client.a.example.com>
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 INVITE Proxy 3 -> Bob

INVITE sip:+19725552222@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss3.b.example.com;lr>
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+13145551111@client.a.example.com>
Content-Type: application/sdp
Content-Length: 154

v=0
o=UserA 2890844526 2890844526 IN IP4 client.a.example.com
s=-
```

```
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F5 100 Trying Proxy 3 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Content-Length: 0
```

F6 180 Ringing B -> Proxy 3

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
 ;received=192.0.2.233
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.b.example.com;lr>
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+19725552222@client.b.example.com>
Content-Length: 0
```

F7 180 Ringing Proxy 3 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.b.example.com;lr>
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+19725552222@client.b.example.com>
Content-Length: 0
```

F8 200 OK Bob -> Proxy 3

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
 ;received=192.0.2.233
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.b.example.com;lr>
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+19725552222@client.b.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 client.b.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F9 200 OK Proxy -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.b.example.com;lr>
From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
To: <tel:+19725552222>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 2 INVITE
Contact: <sip:+19725552222@client.b.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 192.0.2.100
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
   F10 ACK Alice -> Proxy 3

   ACK sip:+19725552222@client.b.example.com SIP/2.0
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bq9
   Max-Forwards: 70
   Route: <sip:ss3.b.example.com;lr>
   From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
   To: <tel:+19725552222>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 ACK
   Content-Length: 0


   F11 ACK Proxy 3 -> Bob

   ACK sip:+19725552222@client.b.example.com SIP/2.0
   Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bq9
    ;received=192.0.2.101
   Max-Forwards: 69
   From: <sip:+13145551111@a.example.com>;tag=9fxced76sl
   To: <tel:+19725552222>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 2 ACK
   Content-Type: application/sdp
   Content-Length: 0

   /* RTP streams are established between A and B*/

   /* User B Hangs Up with User A. */


   F12 BYE Bob -> Proxy 3

   BYE sip:+13145551111@client.a.example.com SIP/2.0
   Via: SIP/2.0/UDP client.b.example.com:5060;branch=z9hG4bKfgaw2
   Max-Forwards: 70
   Route: <sip:ss3.b.example.com;lr>
   From: <tel:+19725552222>;tag=314159
   To: <sip:+13145551111@a.example.com>;tag=9fxced76sl
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 BYE
   Content-Length: 0


   F13 BYE Proxy 3 -> Alice

   BYE sip:+13145551111@client.a.example.com SIP/2.0
```

```
Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
 ;received=192.0.2.100
Via: SIP/2.0/UDP client.b.example.com:5060;branch=z9hG4bKfgaw2
Max-Forwards: 69
From: <tel:+19725552222>;tag=314159
To: <sip:+13145551111@a.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 BYE
Content-Length: 0
```

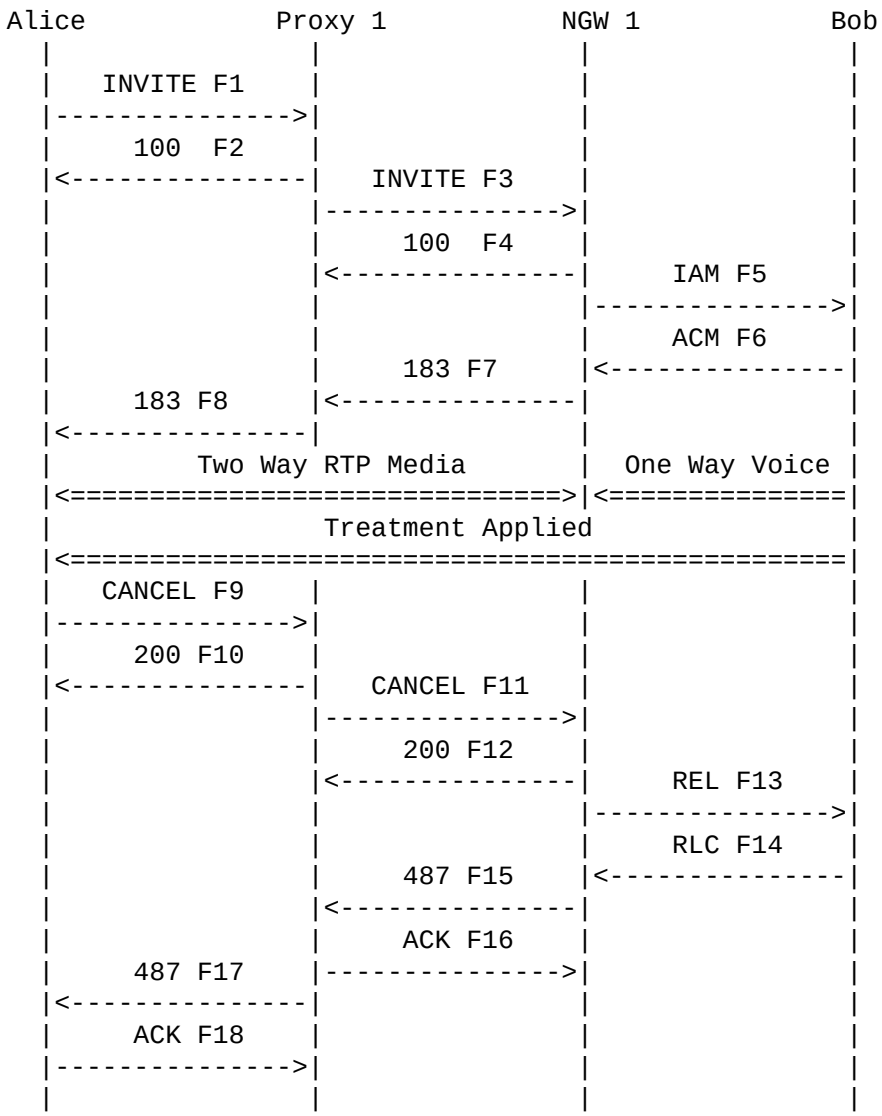
F14 200 OK Alice -> Proxy 3

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss3.b.example.com:5060;branch=z9hG4bK721e418c4.1
 ;received=192.0.2.233
Via: SIP/2.0/UDP client.b.example.com:5060;branch=z9hG4bKfgaw2
 ;received=192.0.2.100
From: <tel:+19725552222>;tag=314159
To: <sip:+13145551111@a.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 BYE
Content-Length: 0
```

F15 200 OK Proxy 3 -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.b.example.com:5060;branch=z9hG4bKfgaw2
 ;received=192.0.2.100
From: <tel:+19725552222>;tag=314159
To: <sip:+13145551111@a.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 BYE
Content-Length: 0
```

2.5. Unsuccessful SIP to PSTN call: Treatment from PSTN



Alice calls Bob in the PSTN through a proxy server Proxy 1 and a Network Gateway NGW 1. The call is rejected by the PSTN with an in-band treatment (tone or recording) played. Alice hears the treatment and then hangs up, which results in a CANCEL (F9) being sent to terminate the call. (A BYE is not sent since no final response was ever received by Alice.)

```
   Message Details

   F1 INVITE Alice -> Proxy 1

   INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.a.example.com>
   Proxy-Authorization: Digest username="alice",
    realm="a.example.com", nonce="01cf8311c3b0b2a2c5ac51bb59a05b40",
    opaque="", uri="sip:+19725552222@ss1.a.example.com;user=phone",
    response="e178fbe430e6680a1690261af8831f40"
   Content-Type: application/sdp
   Content-Length: 154

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.a.example.com
   s=-
   c=IN IP4 client.a.example.com
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F2 100 Trying Proxy 1 -> A

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Content-Length: 0

   /* Proxy 1 uses a Location Service function to determine where B is
   located.  Based upon location analysis the call is forwarded to NGW
   1.  Client for A prepares to receive data on port 49172 from the
   network. */
```

```
F3 INVITE Proxy 1 -> NGW 1

INVITE sip:+19725552222@ngw1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com>
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 100 Trying NGW 1 -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0


F5 IAM NGW 1 -> Bob

IAM
CdPN=972-555-2222,NPI=E.164,NOA=National
CgPN=314-555-1111,NPI=E.164,NOA=National
```

F6 ACM Bob -> NGW 1

ACM

F7 183 Session Progress NGW 1 -> Proxy 1

```
SIP/2.0 183 Session Progress
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F8 183 Session Progress Proxy 1 -> Alice

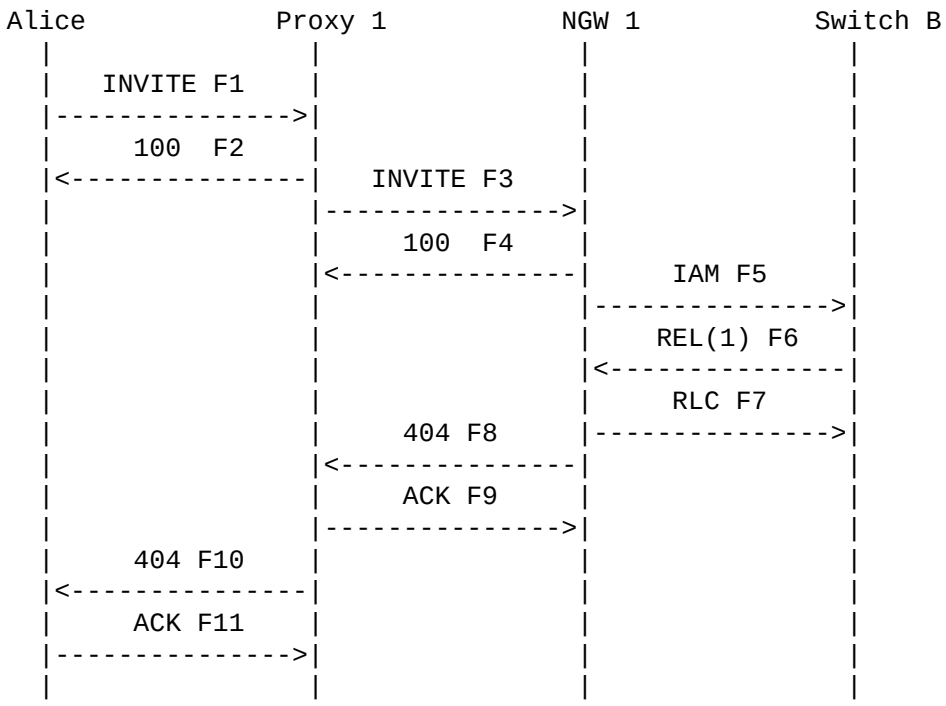
```
SIP/2.0 183 Session Progress
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146
```

```
v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Caller hears the recorded announcement, then hangs up */


F9 CANCEL Alice -> Proxy 1

CANCEL sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 CANCEL
Content-Length: 0


F10 200 OK Proxy 1 -> A

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 CANCEL
Content-Length: 0


F11 CANCEL Proxy 1 -> NGW 1

CANCEL sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

```
   F12 200 OK NGW 1 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 CANCEL
   Content-Length: 0


   F13 REL NGW 1 -> B

   REL
   CauseCode=18 No user responding


   F14 RLC B -> NGW 1

   RLC


   F15 487 Request Terminated NGW 1 -> Proxy 1

   SIP/2.0 487 Request Terminated
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
    ;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F16 ACK Proxy 1 -> NGW 1

   ACK sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Max-Forwards: 70
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
```

```
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0


F17 487 Request Terminated Proxy 1 -> A

SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0


F18 ACK Alice -> Proxy 1

ACK sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0
```

2.6. Unsuccessful SIP to PSTN: REL w/Cause from PSTN



Alice calls PSTN Bob through a Proxy Server Proxy 1 and a Network Gateway NGW 1. The call is rejected by the PSTN with a ANSI ISUP Release message REL containing a specific Cause code. This cause value (1) is mapped by the Gateway to a SIP 404 Address Incomplete response which is proxied back to Alice. For more details of ISUP cause value to SIP response mapping, refer to [4].

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:+44-1234@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com;transport=tcp>
Proxy-Authorization: Digest username="alice",
 realm="a.example.com", nonce="j1c3b0b01cf832da2c5ac51bb59a05b40",
 opaque="", uri="sip:+44-1234@ss1.a.example.com;user=phone",
```

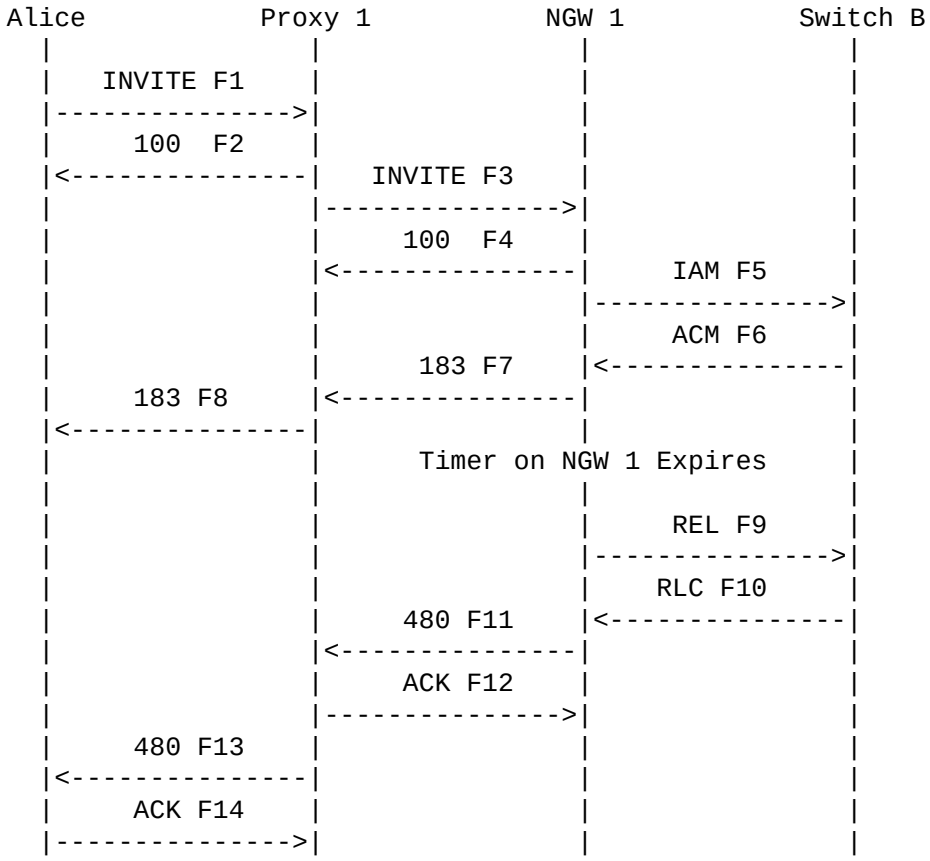
```
    response="a451358d46b55512863efe1dccaa2f42"
   Content-Type: application/sdp
   Content-Length: 154

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.a.example.com
   s=-
   c=IN IP4 client.a.example.com
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F2 100 Trying Proxy 1 -> A

   SIP/2.0 100 Trying
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Content-Length: 0

   /* Proxy 1 uses a Location Service function to determine where B is
   located.  Based upon location analysis the call is forwarded to NGW1.
   Client for A prepares to receive data on port 49172 from the network.
   */


   F3 INVITE Proxy 1 -> NGW 1

   INVITE sip:+44-1234@ngw1.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss1.a.example.com;lr>
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 154
```

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 100 Trying NGW 1 -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0


F5 IAM NGW 1 -> Bob

IAM
CdPN=44-1234,NPI=E.164,NOA=International
CgPN=314-555-1111,NPI=E.164,NOA=National


F6 REL Bob -> NGW 1

REL
CauseValue=1 Unallocated number


F7 RLC NGW 1 -> Bob

RLC

/* Network Gateway maps CauseValue=1 to the SIP message 404 Not
   Found */
```

```
   F8 404 Not Found NGW 1 -> Proxy 1

   SIP/2.0 404 Not Found
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Error-Info: <sip:not-found-ann@ann.a.example.com>
   Content-Length: 0


   F9 ACK Proxy 1 -> NGW 1

   ACK sip:+44-1234@ngw1.a.example.com;user=phone SIP/2.0
   Max-Forwards: 70
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F10 404 Not Found Proxy 1 -> Alice

   SIP/2.0 404 Not Found
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
    ;tag=9fxced76sl
   To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@a.example.com
   CSeq: 1 INVITE
   Error-Info: <sip:not-found-ann@ann.a.example.com>
   Content-Length: 0


   F11 ACK Alice -> Proxy 1

   ACK sip:+44-1234@ss1.a.example.com;user=phone SIP/2.0
   Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
```

```
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+44-1234@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0
```

2.7. Unsuccessful SIP to PSTN: ANM Timeout



Alice calls Bob in the PSTN through a proxy server Proxy 1 and Network Gateway NGW 1. The call is released by the Gateway after a timer expires due to no ANswer Message (ANM) being received. The Gateway sends an ISUP Release REL message to the PSTN and a 480 Temporarily Unavailable response to Alice in the SIP network.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com;transport=tcp>
Proxy-Authorization: Digest username="alice",
 realm="a.example.com", nonce="da2c5ac51bb59a05j1c3b0b01cf832b40",
 opaque="", uri="sip:+19725552222@ss1.a.example.com;user=phone",
 response="579cb9db184cdc25bf816f37cbc03c7d"
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine where B is
located.  Based upon location analysis the call is forwarded to NGW
1.  Client for A prepares to receive data on port 49172 from the
network.*/
```

F2 100 Trying Proxy 1 -> A

```
SIP/2.0  100 Trying
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

```
F3 INVITE Proxy 1 -> NGW 1

INVITE sip:+19725552222@ngw1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 154

v=0
o=alice 2890844526 2890844526 IN IP4 client.a.example.com
s=-
c=IN IP4 client.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 100 Trying NGW 1 -> Proxy 1

SIP/2.0  100 Trying
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Content-Length: 0


F5 IAM NGW 1 -> Bob

IAM
CdPN=972-555-2222,NPI=E.164,NOA=National
CgPN=314-555-1111,NPI=E.164,NOA=National
```

F6 ACM Bob -> NGW 1

ACM

F7 183 Session Progress NGW 1 -> Proxy 1

```
SIP/2.0 183 Session Progress
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F8 183 Session Progress Proxy 1 -> Alice

```
SIP/2.0 183 Session Progress
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss1.a.example.com;lr>
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146
```

```
v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* After NGW 1's timer expires, Network Gateway sends REL to ISUP
network and 480 to SIP network */


F9 REL NGW 1 -> Bob

REL

CauseCode=18 No user responding


F10 RLC Bob -> NGW 1

RLC


F11 480 Temporarily Unavailable NGW 1 -> Proxy 1

SIP/2.0 480 Temporarily Unavailable
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Error-Info: <sip:temp-unavail-ann@ann.a.example.com>
Content-Length: 0


F12 ACK Proxy 1 -> NGW 1

ACK sip:ngw1@a.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
```

```
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0
```

```
F13 480 Temporarily Unavailable F13 Proxy 1 -> Alice

SIP/2.0 480 Temporarily Unavailable
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 INVITE
Error-Info: <sip:temp-unavail-ann@ann.a.example.com>
Content-Length: 0
```

```
F14 ACK Alice -> Proxy 1

ACK sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Max-Forwards: 70
Via: SIP/2.0/TCP client.a.example.com:5060;branch=z9hG4bK74bf9
From: Alice <sip:+13145551111@ss1.a.example.com;user=phone>
 ;tag=9fxced76sl
To: Bob <sip:+19725552222@ss1.a.example.com;user=phone>
 ;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@a.example.com
CSeq: 1 ACK
Content-Length: 0
```

3. PSTN to SIP Dialing

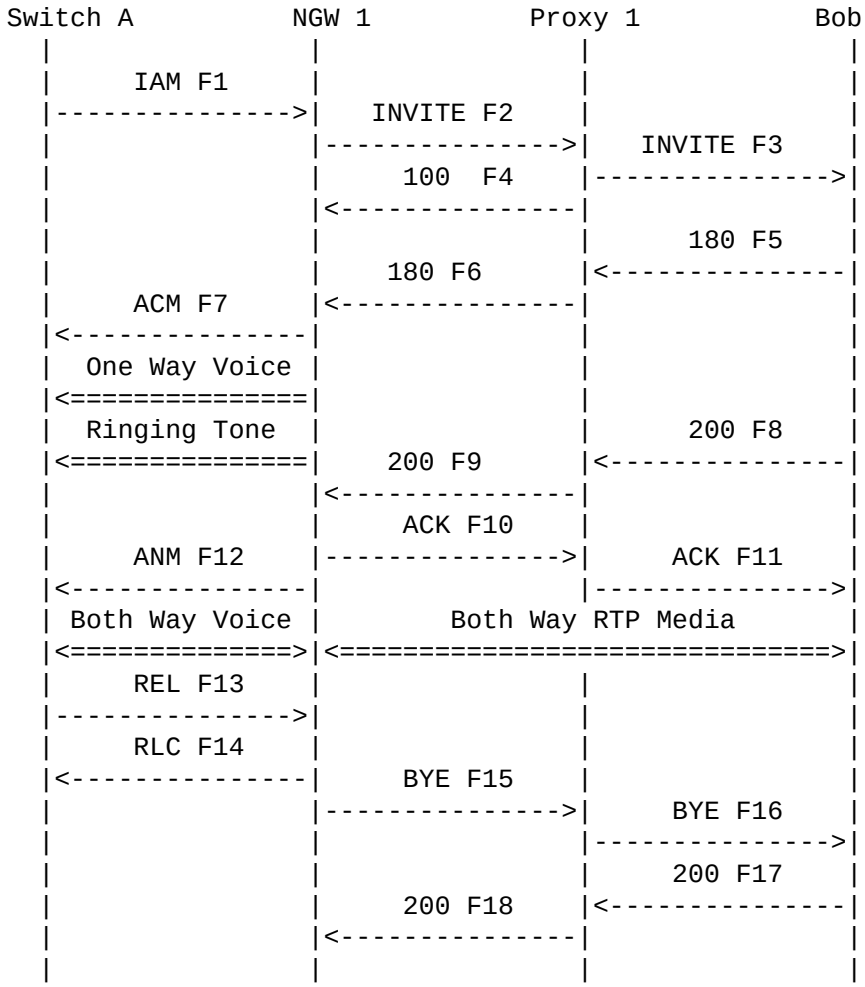
In these scenarios, Alice is placing calls from the PSTN to Bob in a SIP network. Alice's telephone switch signals to a Network Gateway (NGW 1) using ANSI ISUP.

Since the called SIP User Agent does not send in-band signaling information, no early media path needs to be established on the IP side. As a result, the 183 Session Progress response is not used. However, NGW 1 will establish a one way speech path prior to call completion, and generate ringing for the PSTN caller. Any tones or

recordings are generated by NGW 1 and played in this speech path. When the call completes successfully, NGW 1 bridges the PSTN speech path with the IP media path.

To reduce the number of messages, only a single proxy server is shown in these flows, which means that the a.example.com proxy server has access to the b.example.com location service.

3.1. Successful PSTN to SIP call



In this scenario, Alice from the PSTN calls Bob through a Network Gateway NGW1 and Proxy Server Proxy 1. When Bob answers the call, the media path is setup end-to-end. The call terminates when Alice hangs up the call, with Alice's telephone switch sending an ISUP RELease message that is mapped to a BYE by NGW 1.

Message Details

F1 IAM Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
```

F2 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine where B is
located.  Based upon location analysis the call is forwarded to NGW
1.  NGW 1  prepares to receive data on port 3456 from Alice.*/
```

F3 INVITE Proxy 1 -> Bob

```
INVITE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F4 100 Trying Bob -> Proxy 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F5 180 Ringing Bob -> Proxy 1

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
```

```
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com>
Content-Length: 0


F6 180 Ringing Proxy 1 -> NGW 1

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com>
Content-Length: 0


F7 ACM NGW 1 -> Alice

ACM


F8 200 OK Bob -> Proxy 1

SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
Contact: <sip:bob@client.b.example.com>
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 client.b.example.com
t=0 0
m=audio 3456 RTP/AVP 0
```

```
a=rtpmap:0 PCMU/8000


F9 200 OK Proxy 1 -> NGW 1

SIP/2.0 200 OK
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 client.b.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F10 ACK NGW 1 -> Proxy 1

ACK sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0


F11 ACK Proxy 1 -> Bob

ACK sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
```

```
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0


F12 ANM Bob -> NGW 1

ANM

/* RTP streams are established between A and B (via the GW) */

/* Alice Hangs Up with Bob. */


F13 REL Alice -> NGW 1

REL
CauseCode=16 Normal


F14 RLC NGW 1 -> Alice

RLC


F15 BYE NGW 1-> Proxy 1

BYE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 2 BYE
Content-Length: 0


F16 BYE Proxy 1 -> Bob

BYE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
```

```
CSeq: 2 BYE
Content-Length: 0
```

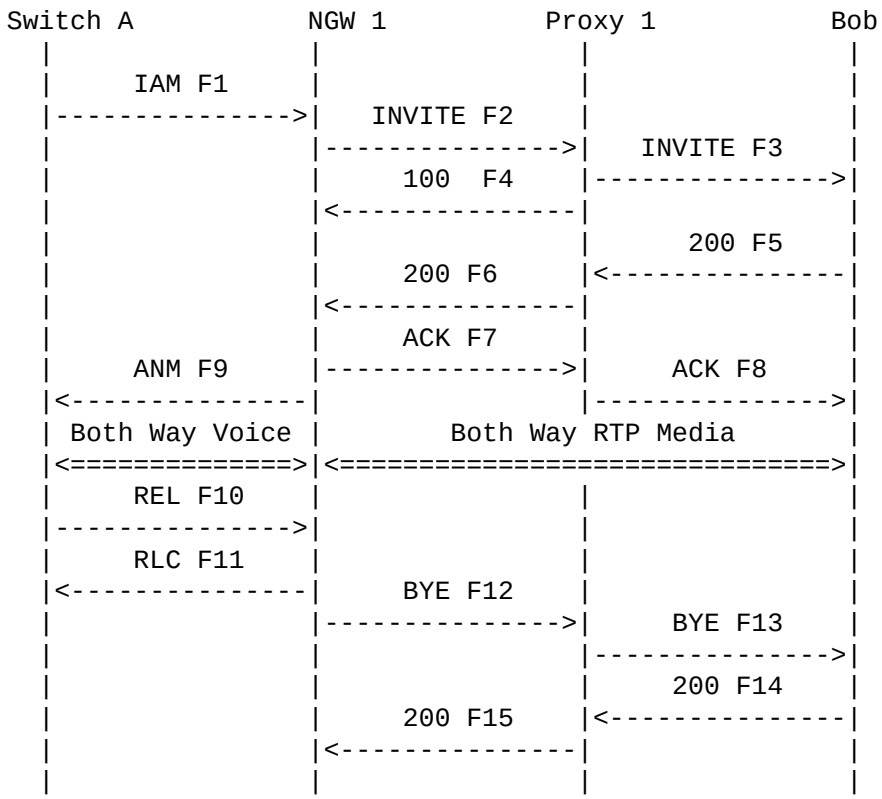
F17 200 OK Bob -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 2 BYE
Content-Length: 0
```

F18 200 OK Proxy 1 -> NGW 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 2 BYE
Content-Length: 0
```

3.2. Successful PSTN to SIP call, Fast Answer



This "fast answer" scenario is similar to 3.1., except that Bob immediately accepts the call, sending a 200 OK (F5) without sending a 180 Ringing response. The Gateway then sends an Answer Message (ANM) without sending an Address Complete Message (ACM). Note that for ETSI and some other ISUP variants, a CONnect message (CON) would be sent instead of the ANM.

Message Details

F1 IAM Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
```

F2 INVITE NGW 1 -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
```

```
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine where B is
located.  Based upon location analysis the call is forwarded to User
B.  Bob  prepares to receive data on port 3456 from Alice.*/


F3 INVITE Proxy 1 -> Bob

INVITE bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F4 100 Trying Proxy 1 -> NGW 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F5 200 OK Bob -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 client.b.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F6 200 OK Proxy 1 -> NGW 1

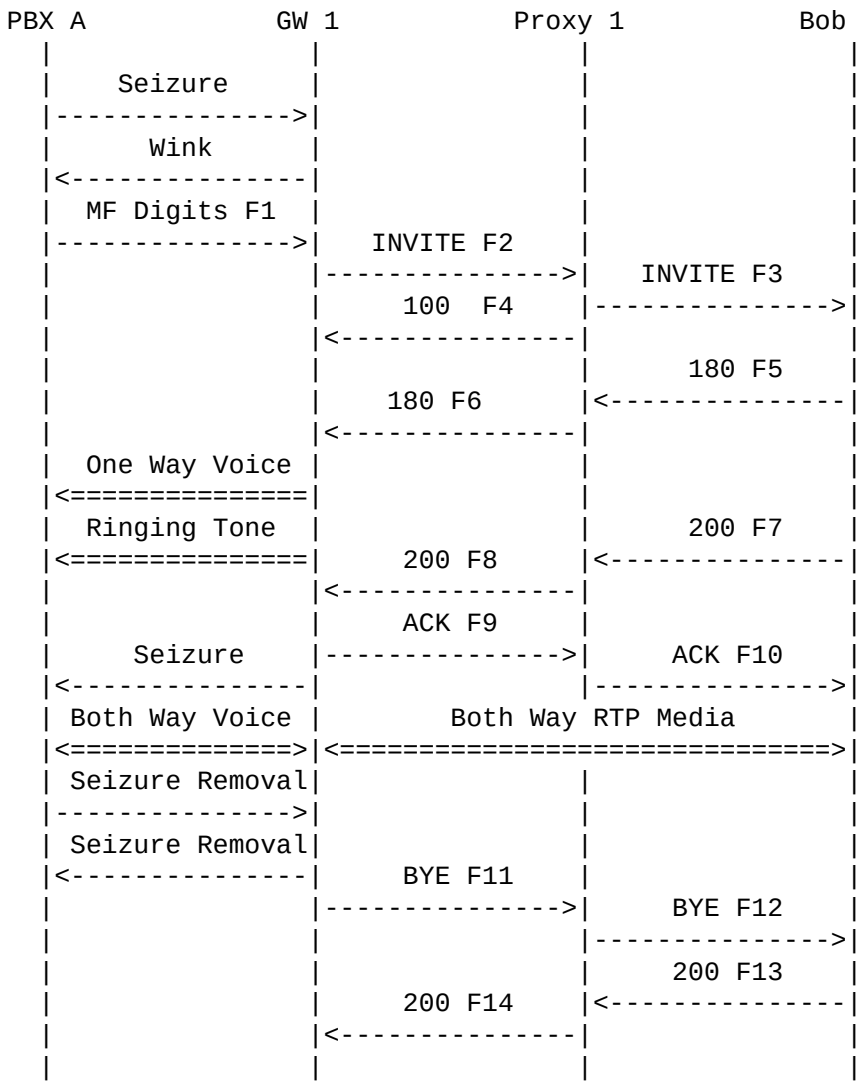
```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com;transport=tcp>
```

```
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=bob 2890844527 2890844527 IN IP4 client.b.example.com
   s=-
   c=IN IP4 client.b.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F7 ACK NGW 1 -> Proxy 1

   ACK bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F8 ACK Proxy 1 -> Bob

   ACK bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=130.131.132.14
   Max-Forwards: 69
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F9 ANM Bob -> NGW 1

   ANM

   /* RTP streams are established between A and B (via the GW) */

   /* Alice Hangs Up with Bob. */
```

```
   F10 REL ser Alice -> NGW 1

   REL
   CauseCode=16 Normal


   F11 RLC NGW 1 -> Alice

   RLC


   F12 BYE NGW 1 -> Proxy 1

   BYE sip:bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 2 BYE
   Content-Length: 0


   F13 BYE Proxy 1 -> Bob

   BYE sip:bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   Max-Forwards: 69
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 2 BYE
   Content-Length: 0


   F14 200 OK Bob -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
```

```
CSeq: 2 BYE
Content-Length: 0


F15 200 OK Proxy 1 -> NGW 1

SIP/2.0 200 OK
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 2 BYE
Content-Length: 0
```

3.3. Successful PBX to SIP call



In this scenario, Alice dials from PBX A to Bob through GW 1 and Proxy 1. This is an example of a call that appears destined for the PSTN but is instead routed to a SIP Client.

Signaling between PBX A and GW 1 is Feature Group B (FGB) circuit associated signaling, in-band Mult-Frequency (MF) outpulsing. After the receipt of the 180 Ringing from Bob, GW 1 generates a ringing tone for Alice.

Bob answers the call by sending a 200 OK. The call terminates when Alice hangs up, causing GW1 to send a BYE.

The Gateway can only identify the trunk group that the call came in on; it cannot identify the individual line on PBX A that is placing the call. The SIP URI used to identify the caller is shown in these flows as sip:551313@gw1.a.example.com.

Message Details

PBX Alice -> GW 1

Seizure

GW 1 -> PBX A

Wink

F1 MF Digits PBX Alice -> GW 1

KP 1 972 555 2222 ST

F2 INVITE GW 1 -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
Max-Forwards: 70
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:551313@gw1.a.example.com;user=phone>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

/* Proxy 1 uses a Location Service function to determine where the phone number +19725552222 is located. Based upon location analysis the call is forwarded to SIP Bob. */

F3 INVITE Proxy 1 -> Bob

```
INVITE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:551313@gw1.a.example.com;user=phone>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F4 100 Trying Proxy 1 -> GW 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F5 180 Ringing Bob -> Proxy 1

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
```

```
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com>
Content-Length: 0


F6 180 Ringing Proxy 1 -> GW 1

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com>
Content-Length: 0

/* One way Voice path is established between GW and the PBX for
ringing. */


F7 200 OK Bob -> Proxy 1

SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
Contact: <sip:bob@client.b.example.com>
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 151

v=0
o=bob 2890844527 2890844527 IN IP4 client.b.example.com
s=-
c=IN IP4 client.b.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

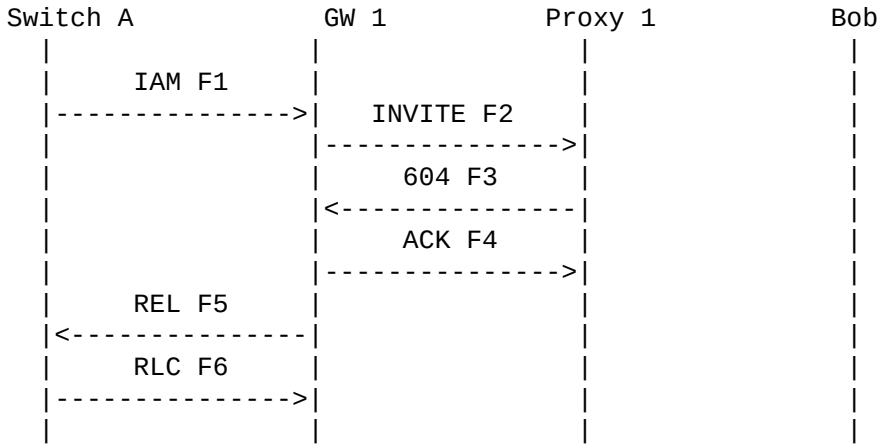
```
   F8 200 OK Proxy 1 -> GW 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:bob@client.b.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=bob 2890844527 2890844527 IN IP4 client.b.example.com
   s=-
   c=IN IP4 client.b.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F9 ACK GW 1 -> Proxy 1

   ACK sip:bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F10 ACK Proxy 1 -> Bob

   ACK sip:bob@client.b.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Max-Forwards: 69
   From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0
```

```
/* RTP streams are established between A and B (via the GW) */

/* Alice Hangs Up with Bob. */


F11 BYE GW 1 -> Proxy 1

BYE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
Max-Forwards: 70
Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 2 BYE
Content-Length: 0


F12 BYE Proxy 1 -> Bob

BYE sip:bob@client.b.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Max-Forwards: 69
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 2 BYE
Content-Length: 0


F13 200 OK Bob -> Proxy 1

SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 2 BYE
Content-Length: 0
```

F14 200 OK Proxy 1 -> GW 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
From: <sip:551313@gw1.a.example.com;user=phone>;tag=jwdkallkzm
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 2 BYE
Content-Length: 0
```

3.4. Unsuccessful PSTN to SIP REL, SIP error mapped to REL



Alice attempts to place a call through Gateway GW 1 and Proxy 1, which is unable to find any routing for the number. The call is rejected by Proxy 1 with a REL message containing a specific Cause value mapped by the gateway based on the SIP error.

Message Details

F1 IAM Alice -> GW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-9999,NPI=E.164,NOA=National
```

F2 INVITE Alice -> Proxy 1

```
INVITE sip:+1972559999@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@gw1.a.example.com;user=phone>;tag=076342s
```

```
To: <sip:+1972559999@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Contact:
<sip:+13145551111@gw1.a.example.com;user=phone;transport=tcp>
Content-Type: application/sdp
Content-Length: 144

v=0
o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service to find a route to +1-972-555-
9999.  A route is not found, so Proxy 1 rejects the call. */


F3 604 Does Not Exist Anywhere Proxy 1 -> GW 1

SIP/2.0 604 Does Not Exist Anywhere
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
From: <sip:+13145551111@gw1.a.example.com;user=phone>;tag=076342s
To: <sip:+1972559999@ss1.a.example.com;user=phone>;tag=6a34d410
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 INVITE
Error-Info: <sip:does-not-exist@ann.a.example.com>
Content-Length: 0


F4 ACK GW 1 -> Proxy 1

ACK sip:+1972559999@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@gw1.a.example.com;user=phone>;tag=076342s
To: <sip:+1972559999@ss1.a.example.com;user=phone>;tag=6a34d410
Call-ID: 4Fde34wkd11wsGFDs3@gw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

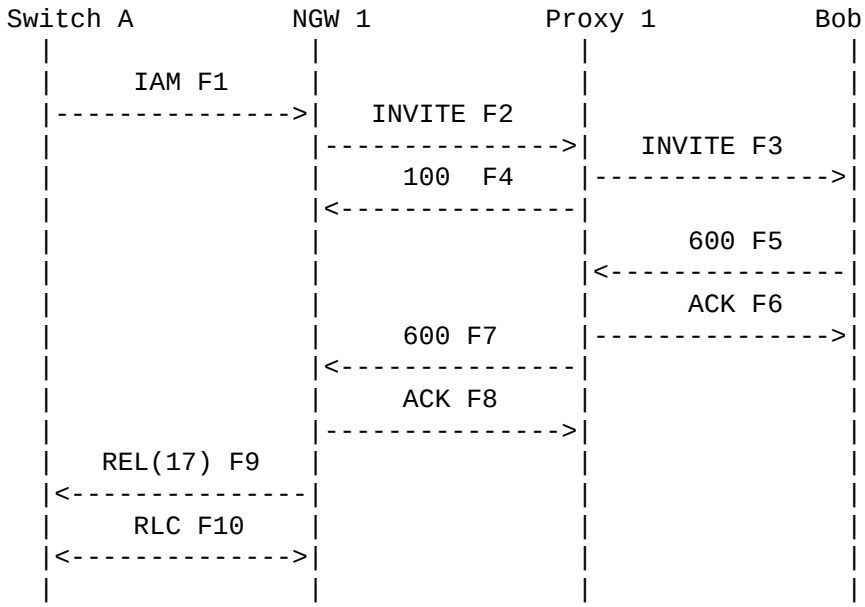
F5 REL GW 1 -> Alice

REL
CauseCode=1

F6 RLC Alice -> GW 1

RLC

3.5. Unsuccessful PSTN to SIP REL, SIP busy mapped to REL



In this scenario, Alice calls Bob through Network Gateway NGW 1 and Proxy 1. The call is routed to Bob by Proxy 1. The call is rejected by Bob who sends a 600 Busy Everywhere response. The Gateway sends a REL message containing a specific Cause value mapped by the gateway based on the SIP error.

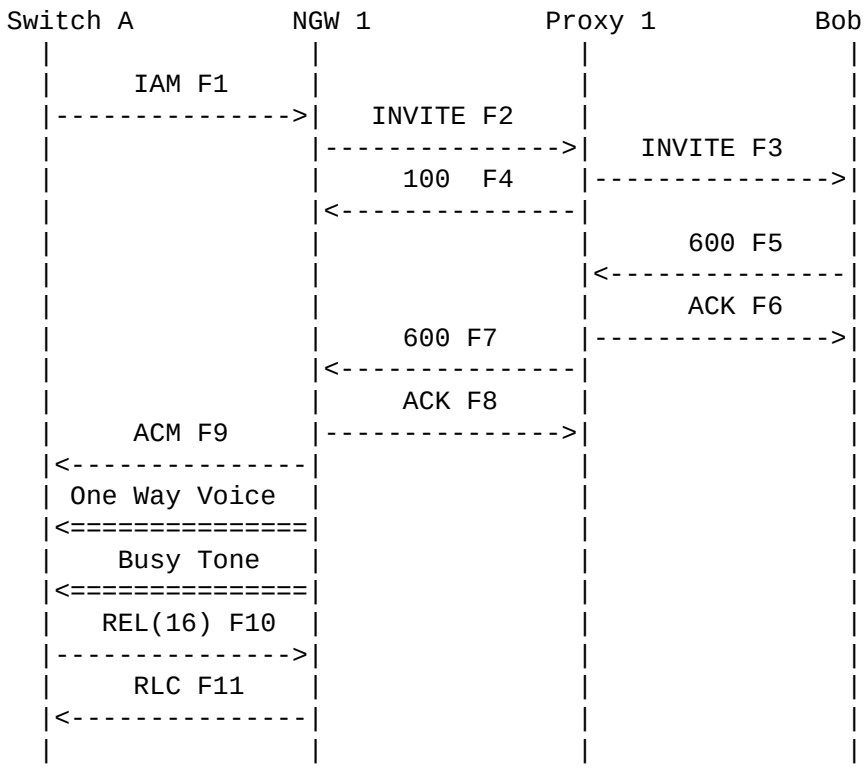
Since no interworking is indicated in the IAM (F1), the busy tone is generated locally by Alice's telephone switch. In some scenarios, the busy signal is generated by the Gateway since interworking is indicated. For more discussion on interworking, refer to [4].

```
   Message Details

   F1 IAM Alice -> NGW 1

   IAM
   CgPN=314-555-1111,NPI=E.164,NOA=National
   CdPN=972-555-2222,NPI=E.164,NOA=National


   F2 INVITE Alice -> Proxy 1

   INVITE sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
   Max-Forwards: 70
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 144

   v=0
   o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
   s=-
   c=IN IP4 gw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Proxy 1 uses a Location Service function to determine a route for
   +19725552222.  The call is then forwarded to Bob. */


   F3 INVITE F3 Proxy 1 -> Bob

   INVITE bob@b.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.201
   Max-Forwards: 69
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com;transport=tcp>
   Content-Type: application/sdp
```

```
   Content-Length: 144

   v=0
   o=GW 2890844527 2890844527 IN IP4 gw1.a.example.com
   s=-
   c=IN IP4 gw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F4 100 Trying Proxy 1 -> NGW 1

   SIP/2.0 100 Trying
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.201
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F5 600 Busy Everywhere Bob -> Proxy 1

   SIP/2.0 600 Busy Everywhere
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.201
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F6 ACK Proxy 1 -> Bob

   ACK bob@b.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Max-Forwards: 70
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0
```

```
F7 600 Busy Everywhere Proxy 1 -> NGW 1

SIP/2.0 600 Busy Everywhere
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0


F8 ACK NGW 1 -> Proxy 1

ACK bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0


F9 REL NGW 1 -> Alice

REL
CauseCode=17 Busy


F10 RLC Alice -> NGW 1

RLC
```

3.6. Unsuccessful PSTN->SIP, SIP error interworking to tones



In this scenario, Alice calls Bob through Network Gateway NGW 1 and Proxy 1. The call is routed to Bob by Proxy 1. The call is rejected by the Bob client. NGW 1 sets up a two way voice path to Alice and plays busy tone. The caller then disconnects

NGW 1 plays the busy tone since the IAM (F1) indicates the interworking is present. In scenario 5.2.2., with no interworking, the busy indication is carried in the REL Cause value and is generated locally instead.

Again, note that for ETSI or ITU ISUP, a CONnect message would be sent instead of the Answer Message.

Message Details

F1 IAM Alice -> NGW 1

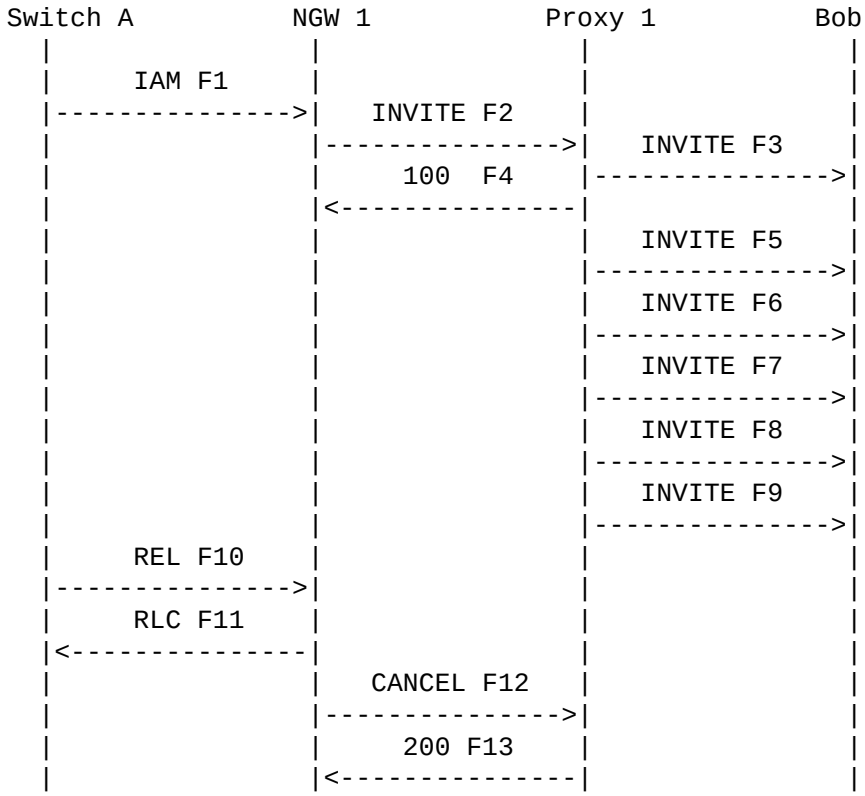
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
Interworking=encountered

```
F2 INVITE NGW1 -> Proxy 1

INVITE sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine a route for
+19725552222.  The call is then forwarded to Bob. */


F3 INVITE Proxy 1 -> Bob

INVITE bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
```

```
a=rtpmap:0 PCMU/8000


F4 100 Trying Bob -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0


F5 600 Busy Everywhere Bob -> Proxy 1

SIP/2.0 600 Busy Everywhere
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0


F6 ACK Proxy 1 -> Bob

ACK bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

```
   F7 600 Busy Everywhere Proxy 1 -> NGW 1

   SIP/2.0 600 Busy Everywhere
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F8 ACK NGW 1 -> Proxy 1

   ACK sip:ngw1@a.example.com SIP/2.0
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
   Max-Forwards: 70
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F9 ACM NGW 1 -> Alice

   ACM

   /* A one way speech path is established between NGW 1 and Alice. */

   /* Call Released after Alice hangs up. */


   F10 REL Alice -> NGW 1

   REL
   CauseCode=16


   F11 RLC NGW 1 -> Alice

   RLC
```

3.7. Unsuccessful PSTN->SIP, ACM timeout



Alice calls Bob through NGW 1 and Proxy 1. Proxy 1 re-sends the INVITE after the expiration of SIP timer T1 without receiving any response from Bob. Bob never responds with 180 Ringing or any other response (it is reachable but unresponsive). After the expiration of a timer, Alice's network disconnects the call by sending a Release message REL. The Gateway maps this to a CANCEL.

Message Details

F1 IAM Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
```

F2 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
```

```
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine a route for
+19725552222.  The call is then forwarded to Bob. */


F3 INVITE Proxy 1 -> Bob

INVITE sip:bob@b.example.com  SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
c c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
   F4 100 Trying Proxy 1 -> NGW 1

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F5 INVITE Proxy 1 -> Bob

   Same as Message F3


   F6 INVITE Proxy 1 -> Bob

   Same as Message F3


   F7 INVITE Proxy 1 -> Bob

   Same as Message F3


   F8 INVITE Proxy 1 -> Bob

   Same as Message F3


   F9 INVITE Proxy 1 -> Bob

   Same as Message F3

   /* Timer expires in Alice's access network. */


   F10 REL Alice -> NGW 1

   REL
   CauseCode=16 Normal


   F11 RLC NGW 1 -> Alice

   RLC
```

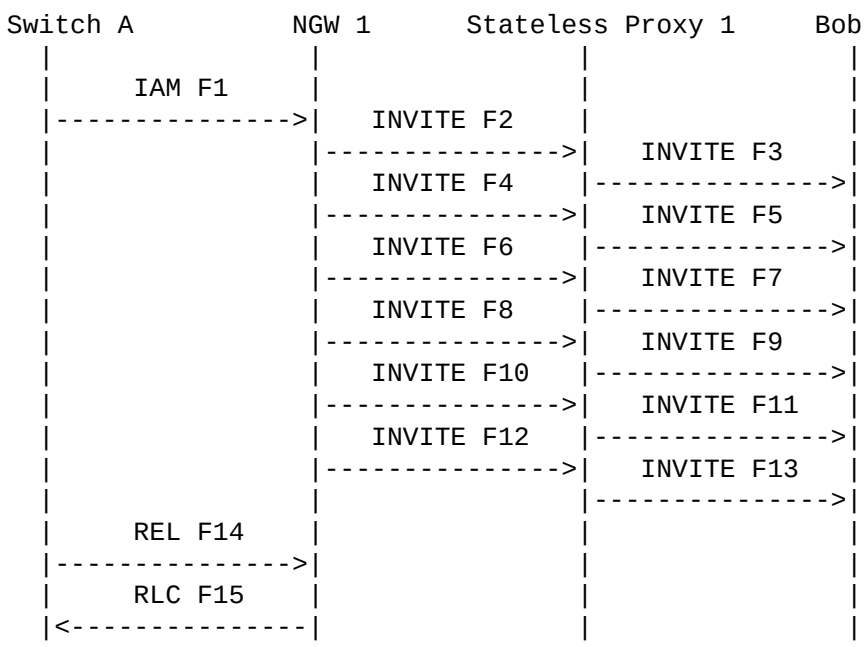
F12 CANCEL NGW 1 -> Proxy 1

```
CANCEL sip:+19725552222@ss1.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F13 200 OK Proxy 1 -> NGW 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

3.8. Unsuccessful PSTN->SIP, ACM timeout, stateless Proxy



In this scenario, Alice calls Bob through NGW 1 and Proxy 1. Since Proxy 1 is stateless (it does not send a 100 Trying response), NGW 1 re-sends the INVITE message after the expiration of SIP timer T1. Bob does not respond with 180 Ringing. Alice's network disconnects the call with a release REL (CauseCode=102 Timeout).

Message Details

F1 IAM Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
```

F2 INVITE NGW 1 -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
```

```
   Contact: <sip:ngw1@a.example.com>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 ngw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Proxy 1 uses a Location Service function to determine a route for
   +19725552222.  The call is then forwarded to Bob. */


   F3 INVITE Proxy 1 -> Bob

   INVITE sip:bob@b.example.com  SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.201
   Max-Forwards: 69
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:ngw1@a.example.com>
   Content-Type: application/sdp
   Content-Length: 146

   v=0
   o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
   s=-
   c=IN IP4 ngw1.a.example.com
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F4 INVITE NGW 1 -> Proxy 1

   Same as Message F2


   F5 INVITE Proxy 1 -> Bob

   Same as Message F3
```

F6 INVITE NGW 1 -> Proxy 1

Same as Message F2

F7 INVITE Proxy 1 -> Bob

Same as Message F3

F8 INVITE NGW 1 -> Proxy 1

Same as Message F2

F9 INVITE Proxy 1 -> Bob

Same as Message F3

F10 INVITE NGW 1 -> Proxy 1

Same as Message F2

F11 INVITE Proxy 1 -> Bob

Same as Message F3

F12 INVITE NGW 1 -> Proxy 1

Same as Message F2

F13 INVITE Proxy 1 -> Bob

Same as Message F3

/* A timer expires in Alice's access network. */

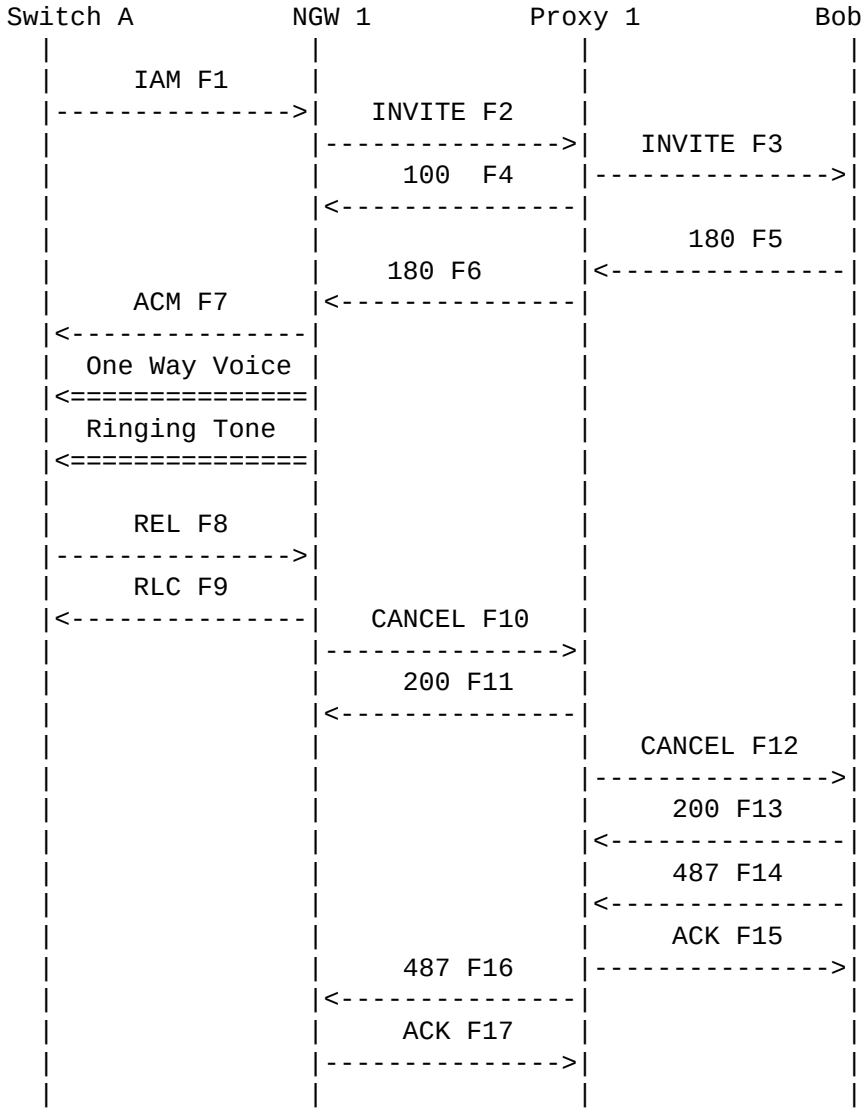
F14 REL Alice -> NGW 1

REL
CauseCode=102 Timeout

F15 RLC NGW 1 -> Alice

RLC

3.9. Unsuccessful PSTN->SIP, Caller Abandonment



In this scenario, Alice calls Bob through NGW 1 and Proxy 1. Bob does not respond with 200 OK. NGW 1 plays ringing tone since the ACM indicates that interworking has been encountered. Alice disconnects the call with a Release message REL which is mapped by NGW 1 to a

CANCEL. Note that if Bob had sent a 200 OK response after the REL, NGW 1 would have sent an ACK and then a BYE to properly terminate the call.

Message Details

F1 IAM Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=972-555-2222,NPI=E.164,NOA=National
```

F2 INVITE Alice -> Proxy 1

```
INVITE sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 uses a Location Service function to determine a route for
+19725552222.  The call is then forwarded to Bob. */
```

F3 INVITE Proxy 1 -> Bob

```
INVITE sip:bob@b.example.com  SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
```

```
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:ngw1@a.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844527 2890844527 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 100 Trying Bob -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.201
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0


F5 180 Ringing Bob -> Proxy 1

SIP/2.0 180 Ringing
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.b.example.com;transport=tcp>
Content-Length: 0


F6 180 Ringing Proxy 1 -> NGW 1

SIP/2.0 180 Ringing
```

```
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:bob@client.b.example.com>
   Content-Length: 0


   F7 ACM NGW 1 -> Alice

   ACM

   /* Alice hangs up */


   F8 REL Alice -> NGW 1

   REL
   CauseCode=16 Normal


   F9 RLC NGW 1 -> Alice

   RLC


   F10 CANCEL NGW 1 -> Proxy 1

   CANCEL sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
   Max-Forwards: 70
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
   Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
   CSeq: 1 CANCEL
   Content-Length: 0


   F11 200 OK Proxy 1 -> NGW 1

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sip:+19725552222@ss1.a.example.com;user=phone>
```

```
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 CANCEL
Content-Length: 0


F12 CANCEL Proxy 1 -> Bob

CANCEL sip:bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 CANCEL
Content-Length: 0


F13 200 OK Bob -> Proxy 1

SIP/2.0 200 OK
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 CANCEL
Content-Length: 0


F14 487 Request Terminated Bob -> Proxy 1

SIP/2.0 487 Request Terminated
Via: SIP/2.0/TCP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0


F15 ACK Proxy 1 -> Bob

ACK sip:bob@b.example.com SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
```

```
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

F16 487 Request Terminated Proxy 1 -> NGW 1

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
 ;received=192.0.2.103
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F17 ACK NGW 1 -> Proxy 1

```
ACK sip:+19725552222@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TCP ngw1.a.example.com:5060;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sip:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sip:+19725552222@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

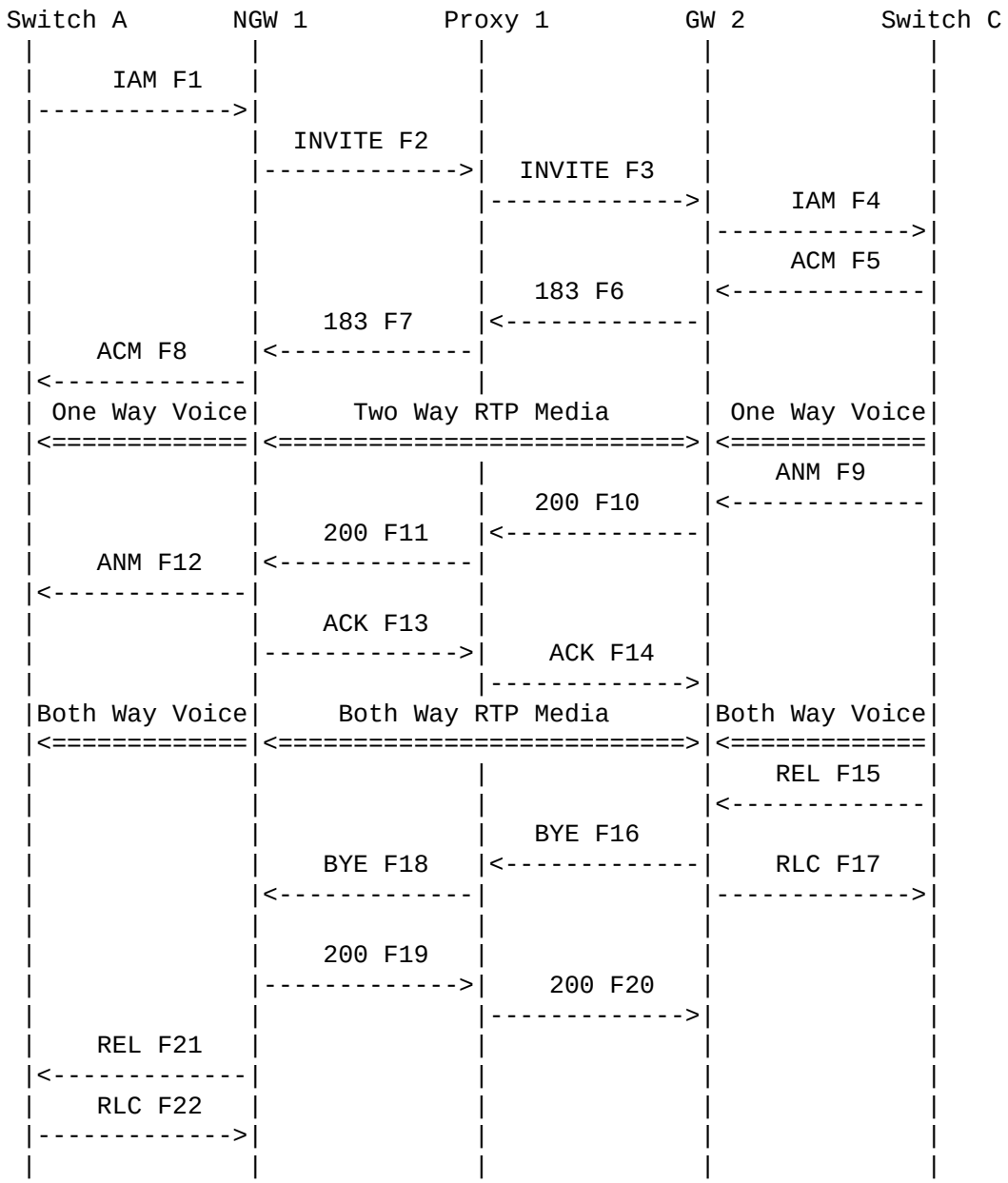
4. PSTN to PSTN Dialing via SIP Network

In these scenarios, both the caller and the called party are in the telephone network, either normal PSTN subscribers or PBX extensions. The calls route through two Gateways and at least one SIP Proxy Server. The Proxy Server performs the authentication and location of the Gateways.

Again it is noted that the intent of this call flows document is not to provide a detailed parameter level mapping of SIP to PSTN protocols. For information on SIP to ISUP mapping, the reader is referred to other references [4].

In these scenarios, the call is successfully completed between the two Gateways, allowing the PSTN or PBX users to communicate. The 183 Session Progress response is used to indicate that in-band alerting may flow from the called party telephone switch to the caller.

4.1. Successful ISUP PSTN to ISUP PSTN call



In this scenario, Alice in the PSTN calls Carol who is an extension on a PBX. Alice's telephone switch signals via SS7 to the Network Gateway NGW 1, while Carol's PBX signals via SS7 with the Gateway GW 2. The CdPN and CgPN are mapped by GW 1 into SIP URIs and placed in the To and From headers. Proxy 1 looks up the dialed digits in the Request-URI and maps the digits to the PBX extension of Carol, which

is served by GW 2. The Proxy in F3 uses the host portion of the Request-URI to identify what private dialing plan is being referenced. The INVITE is then forwarded to GW 2 for call completion. An early media path is established end-to-end so that Alice can hear the ringing tone generated by PBX C.

Carol answers the call and the media path is cut through in both directions. Bob hangs up terminating the call.

Message Details

F1 IAM Switch Alice -> NGW 1

```
IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=918-555-3333,NPI=E.164,NOA=National
```

F2 INVITE NGW 1 -> Proxy 1

```
INVITE sips:+19185553333@ss1.a.example.com;user=phone  SIP/2.0
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
Max-Forwards: 70
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sips:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844526 2890844526 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 consults Location Service and translates the dialed number
to a private number in the Request-URI*/
```

F3 INVITE Proxy 1 -> GW 2

```
INVITE sips:4443333@gw2.a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKwqwee65
```

```
 ;received=192.0.2.103
Max-Forwards: 69
Record-Route: <sips:ss1.a.example.com;lr>
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sips:ngw1@a.example.com>
Content-Type: application/sdp
Content-Length: 146

v=0
o=GW 2890844526 2890844526 IN IP4 ngw1.a.example.com
s=-
c=IN IP4 ngw1.a.example.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F4 IAM GW 2 -> Switch C

IAM
CgPN=314-555-1111,NPI=E.164,NOA=National
CdPN=444-3333,NPI=Private,NOA=Subscriber


F5 ACM Switch C -> GW 2

ACM

/* Based on the ACM message, GW 2 returns a 183 response.  In-band
call progress indications are sent to Alice through NGW 1. */


F6 183 Session Progress GW 2 -> Proxy 1

SIP/2.0 183 Session Progress
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sips:ss1.a.example.com;lr>
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sips:4443333@gw2.a.example.com>
```

```
   Content-Type: application/sdp
   Content-Length: 143

   v=0
   o=GW 987654321 987654321 IN IP4 gw2.a.example.com
   s=-
   c=IN IP4 gw2.a.example.com
   t=0 0
   m=audio 14918 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F7 183 Session Progress Proxy 1 -> GW 1

   SIP/2.0 183 Session Progress
   Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
    ;received=192.0.2.103
   Record-Route: <sips:ss1.a.example.com;lr>
   From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
   To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sips:4443333@gw2.a.example.com>
   Content-Type: application/sdp
   Content-Length: 143

   v=0
   o=GW 987654321 987654321 IN IP4 gw2.a.example.com
   s=-
   c=IN IP4 gw2.a.example.com
   t=0 0
   m=audio 14918 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* NGW 1 receives packets from GW 2 with encoded ringback, tones or
   other audio.  NGW 1 decodes this and places it on the originating
   trunk. */


   F8 ACM NGW 1 -> Switch A

   ACM

   /* Bob answers */
```

F9 ANM Switch C -> GW 2

ANM

F10 200 OK GW 2 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sips:ss1.a.example.com;lr>
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sips:4443333@gw2.a.example.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=GW 987654321 987654321 IN IP4 gw2.a.example.com
s=-
c=IN IP4 gw2.a.example.com
t=0 0
m=audio 14918 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F11 200 OK Proxy 1 -> NGW 1

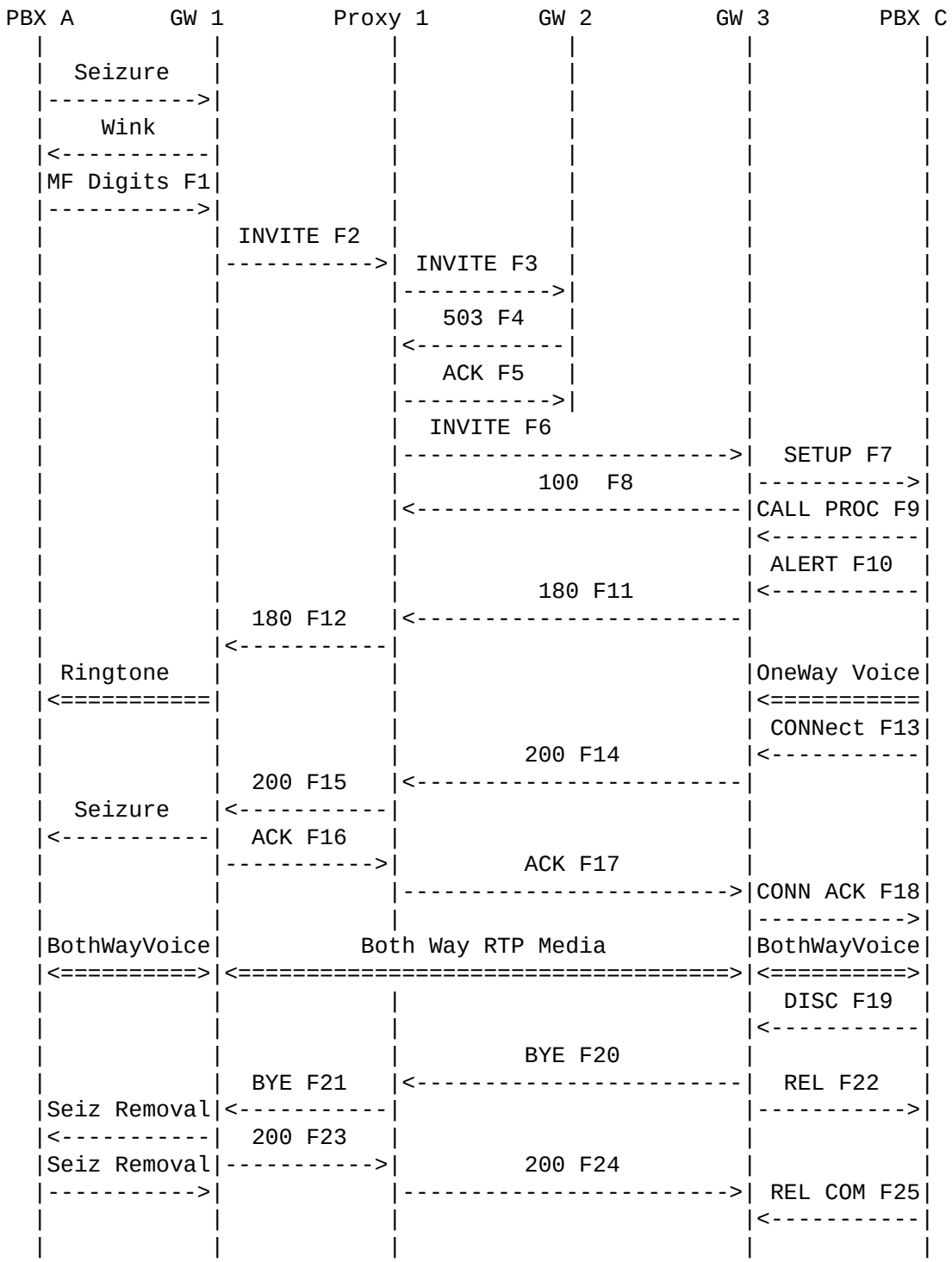
```
SIP/2.0 200 OK
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Record-Route: <sips:ss1.a.example.com;lr>
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 INVITE
Contact: <sips:4443333@gw2.a.example.com>
Content-Type: application/sdp
Content-Length: 143

v=0
o=GW 987654321 987654321 IN IP4 gw2.a.example.com
s=-
c=IN IP4 gw2.a.example.com
```

```
t=0 0
m=audio 14918 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F12 ANM NGW 1 -> Switch A

ANM


F13 ACK NGW 1 -> Proxy 1

ACK sips:4443333@gw2.a.example.com SIP/2.0
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
Max-Forwards: 70
Route: <sips:ss1.a.example.com;lr>
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 ACK
Content-Length: 0


F14 ACK Proxy 1 -> GW 2

ACK sips:4443333@gw2.a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS ngw1.a.example.com:5061;branch=z9hG4bKlueha2
 ;received=192.0.2.103
Max-Forwards: 69
From: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
To: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 1 ACK
Content-Length: 0

/* RTP streams are established between NGW 1 and GW 2. */

/* Bob Hangs Up with Alice. */


F15 REL Switch C -> GW 2

REL
CauseCode=16 Normal
```

```
F16 BYE GW 2 -> Proxy 1

BYE sips:ngw1@a.example.com SIP/2.0
Via: SIP/2.0/TLS gw2.a.example.com:5061;branch=z9hG4bKtexx6
Max-Forwards: 70
Route: <sips:ss1.a.example.com;lr>
From: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
To: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 4 BYE
Content-Length: 0


F17 RLC GW 2 -> Switch C

RLC


F18 BYE Proxy 1 -> NGW 1

BYE sips:ngw1@a.example.com SIP/2.0
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TLS gw2.a.example.com:5061;branch=z9hG4bKtexx6
 ;received=192.0.2.202
Max-Forwards: 69
From: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
To: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 4 BYE
Content-Length: 0


F19 200 OK NGW 1 -> Proxy 1

SIP/2.0 200 OK
Via: SIP/2.0/TLS ss1.a.example.com:5061;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TLS gw2.a.example.com:5061;branch=z9hG4bKtexx6
 ;received=192.0.2.202
From: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
To: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 4 BYE
Content-Length: 0
```

```
F20 200 OK Proxy 1 -> GW 2

SIP/2.0 200 OK
Via: SIP/2.0/TLS gw2.a.example.com:5061;branch=z9hG4bKtexx6
 ;received=192.0.2.202
From: <sips:+19185553333@ss1.a.example.com;user=phone>;tag=314159
To: <sips:+13145551111@ngw1.a.example.com;user=phone>;tag=7643kals
Call-ID: 2xTb9vxSit55XU7p8@ngw1.a.example.com
CSeq: 4 BYE
Content-Length: 0


F21 REL Switch C -> GW 2

REL
CauseCode=16 Normal


F22 RLC GW 2 -> Switch C

RLC
```

4.2. Successful FGB PBX to ISDN PBX call with overflow



PBX Alice calls PBX Carol via Gateway GW 1 and Proxy 1. During the attempt to reach Carol via GW 2, an error is encountered - Proxy 1 receives a 503 Service Unavailable (F4) response to the forwarded INVITE. This could be due to all circuits being busy, or some other outage at GW 2. Proxy 1 recognizes the error and uses an alternative route via GW 3 to terminate the call. From there, the call proceeds normally with Carol answering the call. The call is terminated when Carol hangs up.

Message Details

PBX Alice -> GW 1

Seizure

GW 1 -> PBX A

Wink

F1 MF Digits PBX Alice -> GW 1

KP 444 3333 ST

F2 INVITE GW 1 -> Proxy 1

```
INVITE sip:4443333@ss1.a.example.com SIP/2.0
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
Max-Forwards: 70
From: <sip:551313@gw1.a.example.com>;tag=63412s
To: <sip:4443333@ss1.a.example.com>
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:551313@gw1.a.example.com>
Content-Type: application/sdp
Content-Length: 155

v=0
o=GW 2890844526 2890844526 IN IP4 gw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
   /* Proxy 1 uses a Location Service function to determine where B is
   located.  Response is returned listing alternative routes, GW2 and
   GW3, which are then tried sequentially. */


   F3 INVITE Proxy 1 -> GW 2

   INVITE sip:4443333@gw2.a.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Max-Forwards: 69
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:551313@gw1.a.example.com>
   Content-Type: application/sdp
   Content-Length: 155

   v=0
   o=GW 2890844526 2890844526 IN IP4 gw1.a.example.com
   s=-
   c=IN IP4 gw1.a.example.com
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F4 503 Service Unavailable GW 2 -> Proxy 1

   SIP/2.0 503 Service Unavailable
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F5 ACK Proxy 1 -> GW 2

   ACK sip:4443333@ss1.a.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.1
```

```
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Max-Forward: 70
From: <sip:551313@gw1.a.example.com>;tag=63412s
To: <sip:4443333@ss1.a.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 ACK
Content-Length: 0


F6 INVITE Proxy 1 -> GW 3

INVITE sip:+19185553333@gw3.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Max-Forwards: 69
Record-Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com>;tag=63412s
To: <sip:4443333@ss1.a.example.com>
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:551313@gw1.a.example.com>
Content-Type: application/sdp
Content-Length: 155

v=0
o=GW 2890844526 2890844526 IN IP4 gw1.a.example.com
s=-
c=IN IP4 gw1.a.example.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F7 SETUP GW 3 -> PBX C

Protocol discriminator=Q.931
Message type=SETUP
Bearer capability: Information transfer capability=0 (Speech) or 16
(3.1 kHz audio)
Channel identification=Preferred or exclusive B-channel
Progress indicator=1 (Call is not end-to-end ISDN; further call
progress information may be available inband)
Called party number:
Type of number and numbering plan ID=33 (National number in ISDN
numbering plan)
Digits=918-555-3333
```

```
   F8 100 Trying GW 3 -> Proxy 1

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F9 CALL PROCeeding PBX C -> GW 3

   Protocol discriminator=Q.931
   Message type=CALL PROC


   F10 ALERT PBX C -> GW 3

   Protocol discriminator=Q.931
   Message type=PROG

   /* Based on ALERT message, GW 3 returns a 180 response. */


   F11 180 Ringing GW 3 -> Proxy 1

   SIP/2.0 180 Ringing
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
    ;received=192.0.2.111
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>;tag=123456789
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:+19185553333@gw3.a.example.com;user=phone>
   Content-Length: 0


   F12 180 Ringing Proxy 1 -> GW 1

   SIP/2.0 180 Ringing
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
```

```
    ;received=192.0.2.201
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>;tag=123456789
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:+19185553333@gw3.a.example.com;user=phone>
   Content-Length: 0


   F13 CONNect PBX C -> GW 3

   Protocol discriminator=Q.931
   Message type=CONN


   F14 200 OK GW 3 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
    ;received=192.0.2.111
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com>;tag=63412s
   To: <sip:4443333@ss1.a.example.com>;tag=123456789
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 INVITE
   Contact: <sip:+19185553333@gw3.a.example.com;user=phone>
   Content-Type: application/sdp
   Content-Length: 143

   v=0
   o=GW 987654321 987654321 IN IP4 gw3.a.example.com
   s=-
   c=IN IP4 gw3.a.example.com
   t=0 0
   m=audio 14918 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F15 200 OK Proxy 1 -> GW 1

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
    ;received=192.0.2.201
   Record-Route: <sip:ss1.a.example.com;lr>
   From: <sip:551313@gw1.a.example.com>;tag=63412s
```

```
To: <sip:4443333@ss1.a.example.com>;tag=123456789
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 INVITE
Contact: <sip:+19185553333@gw3.a.example.com;user=phone>
Content-Type: application/sdp
Content-Length: 143

v=0
o=GW 987654321 987654321 IN IP4 gw3.a.example.com
s=-
c=IN IP4 gw3.a.example.com
t=0 0
m=audio 14918 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

GW 1 -> PBX A

Seizure

F16 ACK GW 1 -> Proxy 1

```
ACK sip:+19185553333@gw3.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
Max-Forwards: 70
Route: <sip:ss1.a.example.com;lr>
From: <sip:551313@gw1.a.example.com>;tag=63412s
To: <sip:4443333@ss1.a.example.com>;tag=123456789
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

F17 ACK Proxy 1 -> GW 3

```
ACK sip:+19185553333@gw3.a.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
Via: SIP/2.0/UDP gw1.a.example.com:5060;branch=z9hG4bKwqwee65
 ;received=192.0.2.201
Max-Forwards: 69
From: <sip:551313@gw1.a.example.com>;tag=63412s
To: <sip:4443333@ss1.a.example.com>;tag=123456789
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 ACK
Content-Length: 0
```

```
   F18 CONNect ACK GW 3 -> PBX C

   Protocol discriminator=Q.931
   Message type=CONN ACK

   /* RTP streams are established between GW 1 and GW 3. */

   /* Bob Hangs Up with Alice. */


   F19 DISConnect PBX C -> GW 3

   Protocol discriminator=Q.931
   Message type=DISC
   Cause=16 (Normal clearing)


   F20 BYE GW 3 -> Proxy 1

   BYE sip:551313@gw1.a.example.com SIP/2.0
   Via: SIP/2.0/UDP gw3.a.example.com:5060;branch=z9hG4bKkdjuwq
   Max-Forwards: 70
   Route: <sip:ss1.a.example.com;lr>
   From: <sip:4443333@ss1.a.example.com>;tag=123456789
   To: <sip:551313@gw1.a.example.com>;tag=63412s
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 BYE
   Content-Length: 0


   F21 BYE Proxy 1 -> GW 1

   BYE sip:551313@gw1.a.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
   Via: SIP/2.0/UDP gw3.a.example.com:5060;branch=z9hG4bKkdjuwq
    ;received=192.0.2.203
   Max-Forwards: 69
   From: <sip:4443333@ss1.a.example.com>;tag=123456789
   To: <sip:551313@gw1.a.example.com>;tag=63412s
   Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
   CSeq: 1 BYE
   Content-Length: 0

   GW 1 -> PBX A

   Seizure removal
```

F22 RELease GW 3 -> PBX C

```
Protocol discriminator=Q.931
Message type=REL
```

F23 200 OK GW 1 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.a.example.com:5060;branch=z9hG4bK2d4790.2
 ;received=192.0.2.111
Via: SIP/2.0/UDP gw3.a.example.com:5060;branch=z9hG4bKkdjuwq
 ;received=192.0.2.203
From: <sip:4443333@ss1.a.example.com>;tag=123456789
To: <sip:551313@gw1.a.example.com>;tag=63412s
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 BYE
Content-Length: 0
```

F24 200 OK Proxy 1 -> GW 3

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP gw3.a.example.com:5060;branch=z9hG4bKkdjuwq
 ;received=192.0.2.203
From: <sip:4443333@ss1.a.example.com>;tag=123456789
To: <sip:551313@gw1.a.example.com>;tag=63412s
Call-ID: 2xTb9vxSit55XU7p8@gw1.a.example.com
CSeq: 1 BYE
Content-Length: 0
```

F25 RELease COMplete PBX C -> GW 3

```
Protocol discriminator=Q.931
Message type=REL COM
```

PBX Alice -> GW 1

Seizure removal

5. Security Considerations

This document provides examples of mapping from SIP to ISUP and ISUP to SIP. The gateways in these examples are compliant with the Security Considerations Section of RFC 3398 [4] which is summarized here.

There are few security concerns relating to the mapping of ISUP to SIP besides privacy considerations in the calling party number passing. Some concerns relating to the mapping from tel URI parameters to ISUP include the user creation of parameters and codes relating to called number and local number portability (LNP). An operator of a gateway should use policies similar to those present in PSTN switches to avoid security problems.

The mapping from a SIP response code to an ISUP Cause Code presents a theoretical risk, so a gateway operator may implement policies controlling this mapping. Gateways should also not rely on the contents of the From header field for identity information, as it may be arbitrarily populated by a user. Instead, some sort of cryptographic authentication and authorization should be used for identity determination. These flows show both HTTP Digest for authentication of users, although for brevity, the challenge is not always shown.

The early media cut-through shown in some flows is another potential security risk, but it is also required for proper interaction with the PSTN. Again, a gateway operator should use proper policies relating to early media to prevent fraud and misuse. Finally, a user agent (even a properly authenticated one) can launch multiple simultaneous requests through a gateway, constituting a denial of service attack. The adoption of policies to limit the number of simultaneous requests from a single entity may be used to prevent this attack.

As discussed in the SIP-T framework [7], SIP/ISUP interworking can be employed as an interdomain signaling mechanism that may be subject to pre-existing trust relationships between administrative domains. Any administrative domain implementing SIP-T or SIP/ISUP interworking should have an adequate security apparatus (including elements that manage any appropriate policies to manage fraud and billing in an interdomain environment) in place to ensure that the translation of ISUP information does not result in any security violations.

Although no examples of this are shown in this document, transporting ISUP in SIP bodies may provide opportunities for abuse, fraud, and privacy concerns, especially when SIP-T requests can be generated, inspected or modified by arbitrary SIP endpoints. ISUP MIME bodies should be secured (preferably with S/MIME as detailed in RFC 3261 [2]) to alleviate this concern. Authentication properties provided by S/MIME would allow the recipient of a SIP-T message to ensure that the ISUP MIME body was generated by an authorized entity. Encryption would ensure that only carriers possessing a particular decryption key are capable of inspecting encapsulated ISUP MIME bodies in a SIP request.

6. References

6.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[2] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. E. and Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[3] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with the Session Description Protocol (SDP)", RFC 3264, June 2002.

[4] Camarillo, G., Roach, A. B., Peterson, J. and L. Ong, "Integrated Services Digital Network (ISDN) User Part (ISUP) to Session Initiation Protocol (SIP) Mapping", RFC 3398, December 2002.

[5] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[6] Vaha-Sipila, A., "URLs for Telephone Calls", RFC 2806, April 2000.

[7] Vemuri, A. and J. Peterson, "Session Initiation Protocol for Telephones (SIP-T): Context and Architectures", BCP 63, RFC 3372, September 2002.

[8] Zimmerer, E., Peterson, J., Vemuri, A., Ong, L., Audet, F., Watson, M. and M. Zonoun, "MIME media types for ISUP and QSIG Objects", RFC 3204, December 2001.

[9] Faltstrom, P., "E.164 number and DNS", RFC 2916, September 2000.

6.2. Informative References

[10] Johnston, A., Donovan, S., Sparks, R., Cunningham, C. and K. Summers, "Session Initiation Protocol (SIP) Basic Call Flow Examples", RFC 3665, December 2003.

7. Acknowledgments

Thanks to Rohan Mahy, Adam Roach, Gonzalo Camarillo, Cullen Jennings, and Tom Taylor for their detailed comments during the final review. Thanks to Dean Willis for his early contributions to the development of this document. Thanks to Jon Peterson for his help on the security section.

The authors wish to thank Kundan Singh for performing parser validation of messages.

The authors wish to thank the following individuals for their participation in a detailed review of this call flows document: Aseem Agarwal, Rafi Assadi, Ben Campbell, Sunitha Kumar, Jon Peterson, Marc Petit-Huguenin, Vidhi Rastogi, and Bodgey Yin Shaohua.

The authors also wish to thank the following individuals for their assistance: Jean-Francois Mule, Hemant Agrawal, Henry Sinnreich, David Devanatham, Joe Pizzimenti, Matt Cannon, John Hearty, the whole MCI WorldCom IPOP Design team, Scott Orton, Greg Osterhout, Pat Sollee, Doug Weisenberg, Danny Mistry, Steve McKinnon, and Denise Ingram, Denise Caballero, Tom Redman, Ilya Slain, Pat Sollee, John Truetken, and others from MCI WorldCom, 3Com, Cisco, Lucent and Nortel.

8. Intellectual Property Statement

9. Authors' Addresses

All listed authors actively contributed large amounts of text to this document.

Alan Johnston
MCI
100 South 4th Street
St. Louis, MO 63102
USA

EMail: alan.johnston@mci.com

Steve Donovan
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: sdonovan@dynamicsoft.com

Robert Sparks
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: rsparks@dynamicsoft.com

Chris Cunningham
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: ccunningham@dynamicsoft.com

Kevin Summers
Sonus
1701 North Collins Blvd, Suite 3000
Richardson, TX 75080
USA

EMail: kevin.summers@sonusnet.com

10. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Request for Comments: 3665
BCP: 75
Category: Best Current Practice

A. Johnston
MCI
S. Donovan
R. Sparks
C. Cunningham
dynamicsoft
K. Summers
Sonus

December 2003

Session Initiation Protocol (SIP) Basic Call Flow Examples

Status of this Memo

Copyright Notice

Abstract

This document gives examples of Session Initiation Protocol (SIP) call flows. Elements in these call flows include SIP User Agents and Clients, SIP Proxy and Redirect Servers. Scenarios include SIP Registration and SIP session establishment. Call flow diagrams and message details are shown.

Table of Contents

1. Overview

The call flows shown in this document were developed in the design of a SIP IP communications network. They represent an example minimum set of functionality.

It is the hope of the authors that this document will be useful for SIP implementers, designers, and protocol researchers alike and will help further the goal of a standard implementation of RFC 3261 [1]. These flows represent carefully checked and working group reviewed scenarios of the most basic examples as a companion to the specifications.

These call flows are based on the current version 2.0 of SIP in RFC 3261 [1] with SDP usage described in RFC 3264 [2]. Other RFCs also comprise the SIP standard but are not used in this set of basic call flows.

Call flow examples of SIP interworking with the PSTN through gateways are contained in a companion document, RFC 3666 [5].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [4].

1.1. General Assumptions

A number of architecture, network, and protocol assumptions underlie the call flows in this document. Note that these assumptions are not requirements. They are outlined in this section so that they may be taken into consideration and to aid in the understanding of the call flow examples.

The authentication of SIP User Agents in these example call flows is performed using HTTP Digest as defined in [1] and [3].

Some Proxy Servers in these call flows insert Record-Route headers into requests to ensure that they are in the signaling path for future message exchanges.

These flows show TCP, TLS, and UDP for transport. See the discussion in RFC 3261 for details on the transport issues for SIP.

1.2. Legend for Message Flows

Dashed lines (---) represent signaling messages that are mandatory to the call scenario. These messages can be SIP or PSTN signaling. The arrow indicates the direction of message flow.

Double dashed lines (===) represent media paths between network elements.

Messages with parentheses around their name represent optional messages.

Messages are identified in the Figures as F1, F2, etc. This references the message details in the list that follows the Figure. Comments in the message details are shown in the following form:

```
 /* Comments. */
```

1.3. SIP Protocol Assumptions

This document does not prescribe the flows precisely as they are shown, but rather the flows illustrate the principles for best practice. They are best practices usages (orderings, syntax, selection of features for the purpose, handling of error) of SIP methods, headers and parameters. IMPORTANT: The exact flows here must not be copied as is by an implementer due to specific incorrect characteristics that were introduced into the document for convenience and are listed below. To sum up, the basic flows represent well-reviewed examples of SIP usage, which are best common practice according to IETF consensus.

For simplicity in reading and editing the document, there are a number of differences between some of the examples and actual SIP messages. For example, the HTTP Digest responses are not actual MD5 encodings. Call-IDs are often repeated, and CSeq counts often begin at 1. Header fields are usually shown in the same order. Usually only the minimum required header field set is shown, others that would normally be present such as Accept, Supported, Allow, etc are not shown.

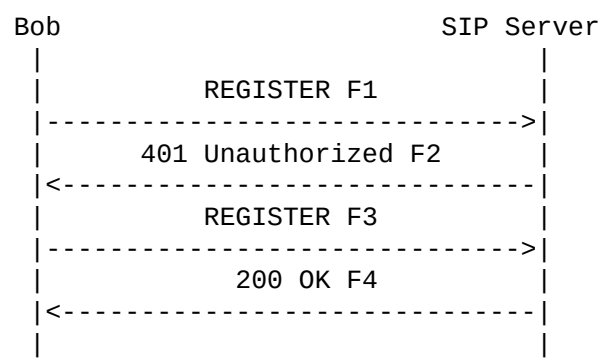
Actors:

Element	Display Name	URI	IP Address
User Agent	Alice	alice@atlanta.example.com	192.0.2.101
User Agent	Bob	bob@biloxi.example.com	192.0.2.201
User Agent		bob@chicago.example.com	192.0.2.100
Proxy Server		ss1.atlanta.example.com	192.0.2.111
Proxy/Registrar		ss2.biloxi.example.com	192.0.2.222
Proxy Server		ss3.chicago.example.com	192.0.2.233
ALG		alg1.atlanta.example.com	192.0.2.128

2. SIP Registration

Registration binds a particular device Contact URI with a SIP user Address of Record (AOR).

2.1. Successful New Registration



Bob sends a SIP REGISTER request to the SIP server. The request includes the user's contact list. This flow shows the use of HTTP Digest for authentication using TLS transport. TLS transport is used due to the lack of integrity protection in HTTP Digest and the danger of registration hijacking without it, as described in RFC 3261 [1]. The SIP server provides a challenge to Bob. Bob enters her/his valid user ID and password. Bob's SIP client encrypts the user information according to the challenge issued by the SIP server and sends the response to the SIP server. The SIP server validates the user's credentials. It registers the user in its contact database and returns a response (200 OK) to Bob's SIP client. The response includes the user's current contact list in Contact headers. The format of the authentication shown is HTTP digest. It is assumed that Bob has not previously registered with this Server.

Message Details

F1 REGISTER Bob -> SIP Server

```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>
Content-Length: 0
```

F2 401 Unauthorized SIP Server -> Bob

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="atlanta.example.com", qop="auth",
 nonce="ea9c8e88df84f1cec4341ae6cbe5a359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

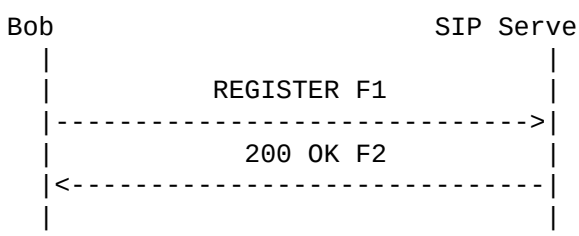
F3 REGISTER Bob -> SIP Server

```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashd92
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=ja743ks76zlflH
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER
Contact: <sips:bob@client.biloxi.example.com>
Authorization: Digest username="bob", realm="atlanta.example.com"
 nonce="ea9c8e88df84f1cec4341ae6cbe5a359", opaque="",
 uri="sips:ss2.biloxi.example.com",
 response="dfe56131d1958046689d83306477ecc"
Content-Length: 0
```

F4 200 OK SIP Server -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashd92
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=ja743ks76zlflH
To: Bob <sips:bob@biloxi.example.com>;tag=37GkEhwl6
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=3600
Content-Length: 0
```

2.2. Update of Contact List



Bob wishes to update the list of addresses where the SIP server will redirect or forward INVITE requests.

Bob sends a SIP REGISTER request to the SIP server. Bob's request includes an updated contact list. Since the user already has authenticated with the server, the user supplies authentication credentials with the request and is not challenged by the server. The SIP server validates the user's credentials. It registers the user in its contact database, updates the user's contact list, and returns a response (200 OK) to Bob's SIP client. The response includes the user's current contact list in Contact headers.

Message Details

F1 REGISTER Bob -> SIP Server

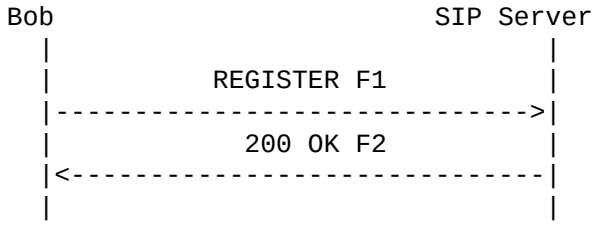
```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: mailto:bob@biloxi.example.com
Authorization: Digest username="bob", realm="atlanta.example.com",
 qop="auth", nonce="1cec4341ae6cbe5a359ea9c8e88df84f", opaque="",
 uri="sips:ss2.biloxi.example.com",
 response="71ba27c64bd01de719686aa4590d5824"
Content-Length: 0
```

F2 200 OK SIP Server -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>;tag=34095828jh
```

```
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=3600
Contact: <mailto:bob@biloxi.example.com>;expires=4294967295
Content-Length: 0
```

2.3. Request for Current Contact List



Bob sends a register request to the Proxy Server containing no Contact headers, indicating the user wishes to query the server for the user's current contact list. Since the user already has authenticated with the server, the user supplies authentication credentials with the request and is not challenged by the server. The SIP server validates the user's credentials. The server returns a response (200 OK) which includes the user's current registration list in Contact headers.

Message Details

F1 REGISTER Bob -> SIP Server

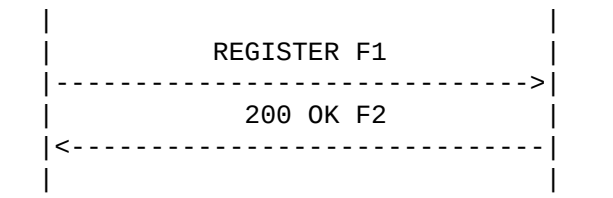
```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Authorization: Digest username="bob", realm="atlanta.example.com",
 nonce="df84f1cec4341ae6cbe5ap359a9c8e88", opaque="",
 uri="sips:ss2.biloxi.example.com",
 response="aa7ab4678258377c6f7d4be6087e2f60"
Content-Length: 0
```

F2 200 OK SIP Server -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
```

```
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>;tag=jqoiweu75
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>;expires=3600
Contact: <mailto:bob@biloxi.example.com>;expires=4294967295
Content-Length: 0
```

2.4. Cancellation of Registration



Bob wishes to cancel their registration with the SIP server. Bob sends a SIP REGISTER request to the SIP server. The request has an expiration period of 0 and applies to all existing contact locations. Since the user already has authenticated with the server, the user supplies authentication credentials with the request and is not challenged by the server. The SIP server validates the user's credentials. It clears the user's contact list, and returns a response (200 OK) to Bob's SIP client.

Message Details

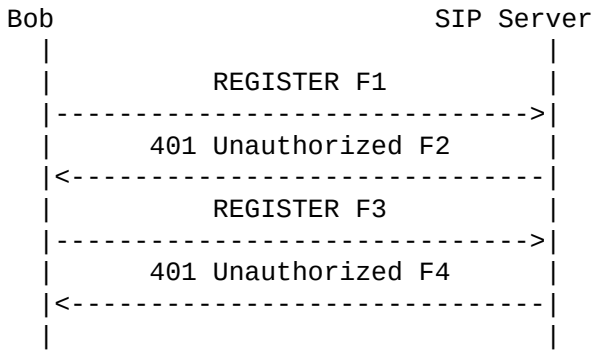
F1 REGISTER Bob -> SIP Server

```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Expires: 0
Contact: *
Authorization: Digest username="bob", realm="atlanta.example.com",
 nonce="88df84f1cac4341aea9c8ee6cbe5a359", opaque="",
 uri="sips:ss2.biloxi.example.com",
 response="ff0437c51696f9a76244f0cf1dbabbea"
Content-Length: 0
```

F2 200 OK SIP Server -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>;tag=1418nmdsrf
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Content-Length: 0
```

2.5. Unsuccessful Registration



Bob sends a SIP REGISTER request to the SIP Server. The SIP server provides a challenge to Bob. Bob enters her/his user ID and password. Bob's SIP client encrypts the user information according to the challenge issued by the SIP server and sends the response to the SIP server. The SIP server attempts to validate the user's credentials, but they are not valid (the user's password does not match the password established for the user's account). The server returns a response (401 Unauthorized) to Bob's SIP client.

Message Details

F1 REGISTER Bob -> SIP Server

```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
Contact: <sips:bob@client.biloxi.example.com>
Content-Length: 0
```

F2 Unauthorized SIP Server -> Bob

```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=a73kszlfl
To: Bob <sips:bob@biloxi.example.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="atlanta.example.com", qop="auth",
 nonce="f1cec4341ae6ca9c8e88df84be55a359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

F3 REGISTER Bob -> SIP Server

```
REGISTER sips:ss2.biloxi.example.com SIP/2.0
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashd92
Max-Forwards: 70
From: Bob <sips:bob@biloxi.example.com>;tag=JueHGuidj28dfga
To: Bob <sips:bob@biloxi.example.com>
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER
Contact: <sips:bob@client.biloxi.example.com>
Authorization: Digest username="bob", realm="atlanta.example.com",
 nonce="f1cec4341ae6ca9c8e88df84be55a359", opaque="",
 uri="sips:ss2.biloxi.example.com",
 response="61f8470ceb87d7ebf508220214ed438b"
Content-Length: 0

/*  The response above encodes the incorrect password */
```

F4 401 Unauthorized SIP Server -> Bob

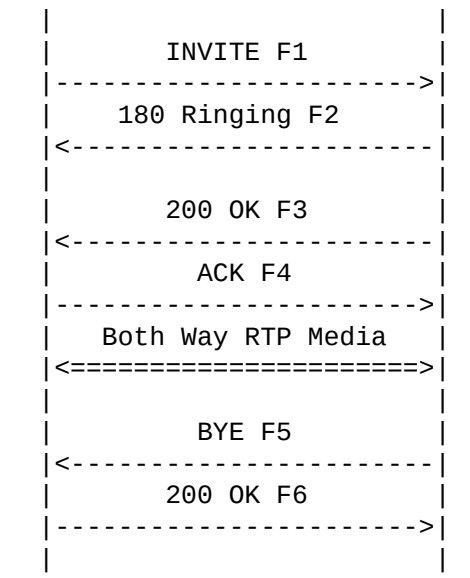
```
SIP/2.0 401 Unauthorized
Via: SIP/2.0/TLS client.biloxi.example.com:5061;branch=z9hG4bKnashd92
 ;received=192.0.2.201
From: Bob <sips:bob@biloxi.example.com>;tag=JueHGuidj28dfga
To: Bob <sips:bob@biloxi.example.com>;tag=1410948204
Call-ID: 1j9FpLxk3uxtm8tn@biloxi.example.com
CSeq: 2 REGISTER
WWW-Authenticate: Digest realm="atlanta.example.com", qop="auth",
 nonce="84f1c1ae6cbe5ua9c8e88dfa3ecm3459",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

3. SIP Session Establishment

This section details session establishment between two SIP User Agents (UAs): Alice and Bob. Alice (sip:alice@atlanta.example.com) and Bob (sip:bob@biloxi.example.com) are assumed to be SIP phones or SIP-enabled devices. The successful calls show the initial signaling, the exchange of media information in the form of SDP payloads, the establishment of the media session, then finally the termination of the call.

HTTP Digest authentication is used by Proxy Servers to authenticate the caller Alice. It is assumed that Bob has registered with Proxy Server Proxy 2 as per Section 2 to be able to receive the calls via the Proxy.

3.1. Successful Session Establishment



In this scenario, Alice completes a call to Bob directly.

Message Details

F1 INVITE Alice -> Bob

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
```

```
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F2 180 Ringing Bob -> Alice

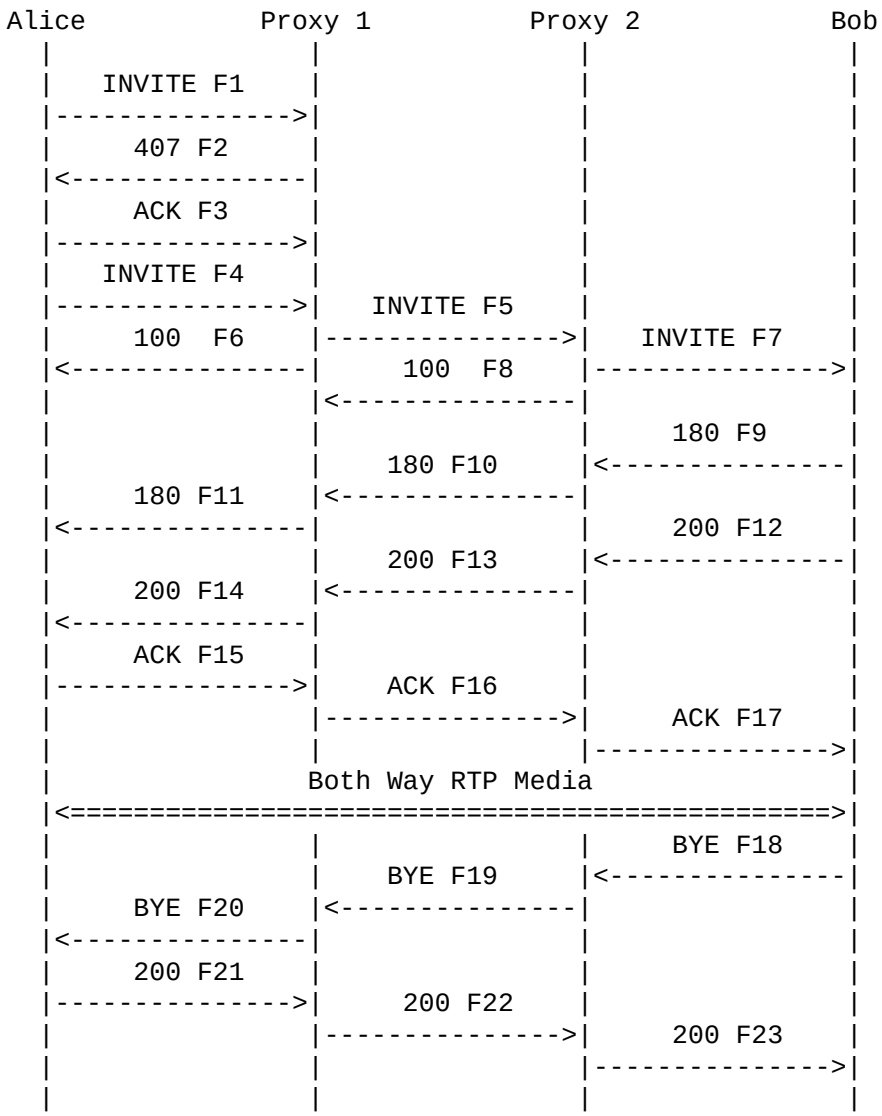
```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Length: 0
```

F3 200 OK Bob -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
F4 ACK Alice -> Bob

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bd5
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=8321234356
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0

/* RTP streams are established between Alice and Bob */

/* Bob Hangs Up with Alice. Note that the CSeq is NOT 2, since
   Alice and Bob maintain their own independent CSeq counts.
   (The INVITE was request 1 generated by Alice, and the BYE is
   request 1 generated by Bob) */


F5 BYE Bob -> Alice

BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0


F6 200 OK Alice -> Bob

SIP/2.0 200 OK
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=8321234356
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

3.2. Session Establishment Through Two Proxies



In this scenario, Alice completes a call to Bob using two proxies Proxy 1 and Proxy 2. The initial INVITE (F1) contains a pre-loaded Route header with the address of Proxy 1 (Proxy 1 is configured as a default outbound proxy for Alice). The request does not contain the Authorization credentials Proxy 1 requires, so a 407 Proxy Authorization response is sent containing the challenge information. A new INVITE (F4) is then sent containing the correct credentials and the call proceeds. The call terminates when Bob disconnects by initiating a BYE message.

Proxy 1 inserts a Record-Route header into the INVITE message to ensure that it is present in all subsequent message exchanges. Proxy 2 also inserts itself into the Record-Route header. The ACK (F15) and BYE (F18) both have a Route header.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b43
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 challenges Alice for authentication */
```

F2 407 Proxy Authorization Required Proxy 1 -> Alice

```
SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b43
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=3flal12sf
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="atlanta.example.com", qop="auth",
 nonce="f84f1cec41e6cbe5aea9c8e88d359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

```
   F3 ACK Alice -> Proxy 1

   ACK sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b43
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=3flal12sf
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0

   /* Alice responds be re-sending the INVITE with authentication
      credentials in it. */


   F4 INVITE Alice -> Proxy 1

   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
   Max-Forwards: 70
   Route: <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 INVITE
   Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
   Proxy-Authorization: Digest username="alice",
    realm="atlanta.example.com",
    nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
    uri="sip:bob@biloxi.example.com",
    response="42ce3cef44b22f50c6a6071bc8"
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Proxy 1 accepts the credentials and forwards the INVITE to Proxy
   2.  Client for Alice prepares to receive data on port 49172 from the
   network. */
```

F5 INVITE Proxy 1 -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F6 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 0
```

F7 INVITE Proxy 2 -> Bob

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 68
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
```

```
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F8 100 Trying Proxy 2 -> Proxy 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 0
```

F9 180 Ringing Bob -> Proxy 2

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
CSeq: 2 INVITE
Content-Length: 0
```

F10 180 Ringing Proxy 2 -> Proxy 1

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
CSeq: 2 INVITE
Content-Length: 0
```

F11 180 Ringing Proxy 1 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 3848276298220188511@atlanta.example.com
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
CSeq: 2 INVITE
Content-Length: 0
```

F12 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 2 INVITE
```

```
   Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 147

   v=0
   o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
   s=-
   c=IN IP4 192.0.2.201
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F13 200 OK Proxy 2 -> Proxy 1

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss2.biloxi.example.com;lr>,
    <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 INVITE
   Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
   Content-Type: application/sdp
   Content-Length: 147

   v=0
   o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
   s=-
   c=IN IP4 192.0.2.201
   t=0 0
   m=audio 3456 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F14 200 OK Proxy 1 -> Alice

   SIP/2.0 200 OK
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Record-Route: <sip:ss2.biloxi.example.com;lr>,
    <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
```

```
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.biloxi.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F15 ACK Alice -> Proxy 1

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b76
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>,
 <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0


F16 ACK Proxy 1 -> Proxy 2

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b76
 ;received=192.0.2.101
Max-Forwards: 69
Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 384827629822018851l@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0


F17 ACK Proxy 2 -> Bob

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
```

```
   Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74b76
    ;received=192.0.2.101
   Max-Forwards: 68
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 2 ACK
   Content-Length: 0

   /* RTP streams are established between Alice and Bob */

   /* Bob Hangs Up with Alice. */

   /* Again, note that the CSeq is NOT 3.  Alice and Bob maintain
      their own separate CSeq counts */


   F18 BYE Bob -> Proxy 2

   BYE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
   Max-Forwards: 70
   Route: <sip:ss2.biloxi.example.com;lr>,
    <sip:ss1.atlanta.example.com;lr>
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0


   F19 BYE Proxy 2 -> Proxy 1

   BYE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
    ;received=192.0.2.201
   Max-Forwards: 69
   Route: <sip:ss1.atlanta.example.com;lr>
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 3848276298220188511@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0
```

F20 BYE Proxy 1 -> Alice

```
BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
Max-Forwards: 68
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

F21 200 OK Alice -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

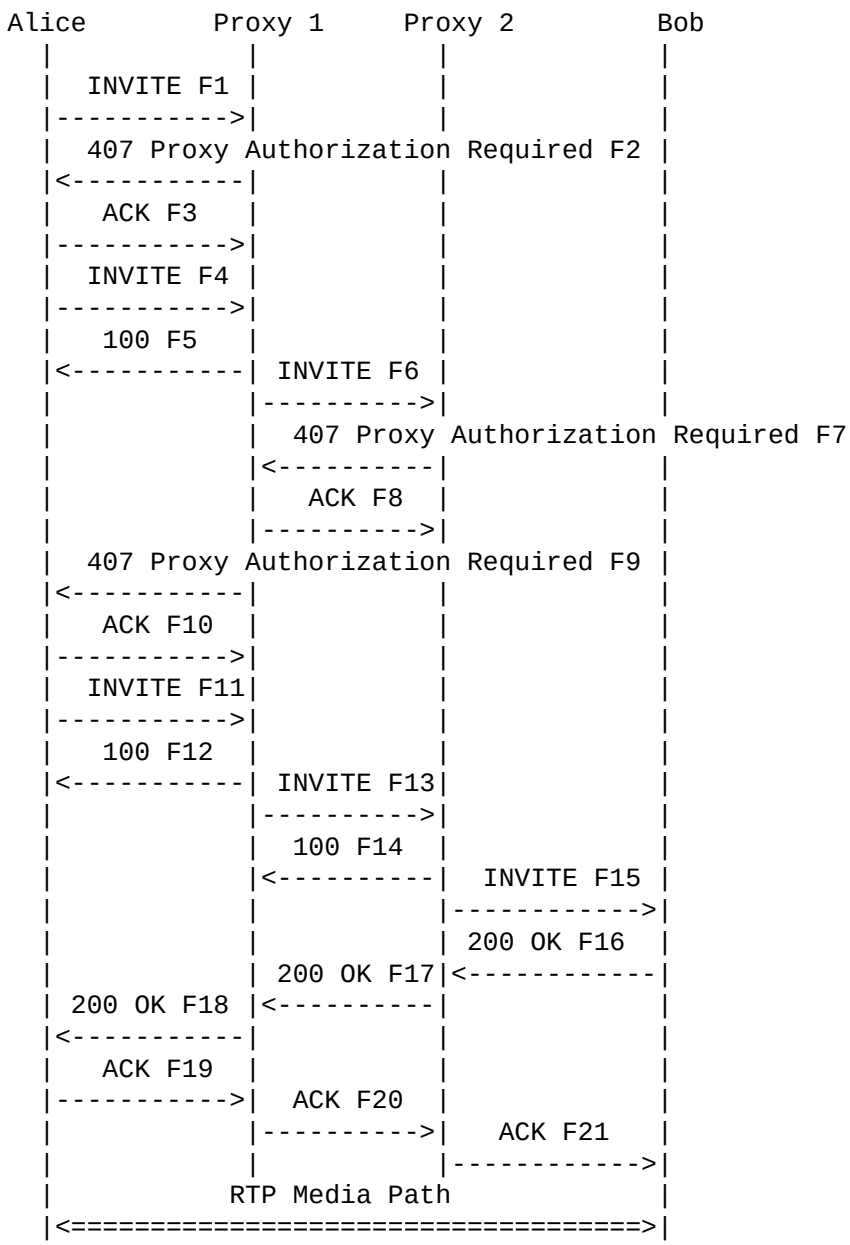
F22 200 OK Proxy 1 -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.101
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

F23 200 OK Proxy 2 -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

3.3. Session with Multiple Proxy Authentication



In this scenario, Alice completes a call to Bob using two proxies Proxy 1 and Proxy 2. Alice has valid credentials in both domains. Since the initial INVITE (F1) does not contain the Authorization credentials Proxy 1 requires, so a 407 Proxy Authorization response is sent containing the challenge information. A new INVITE (F4) is

then sent containing the correct credentials and the call proceeds after Proxy 2 challenges and receives valid credentials. The call terminates when Bob disconnects by initiating a BYE message.

Proxy 1 inserts a Record-Route header into the INVITE message to ensure that it is present in all subsequent message exchanges. Proxy 2 also inserts itself into the Record-Route header.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b03
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 challenges Alice for authentication */
```

F2 407 Proxy Authorization Required Proxy 1 -> Alice

```
SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b03
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=876321
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="atlanta.example.com", qop="auth",
 nonce="wf84f1cczx41ae6cbeaea9ce88d359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

```
   F3 ACK Alice -> Proxy 1

   ACK sip:bob@biloxi.example.com SIP/2.0
   Max-Forwards: 70
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b03
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=876321
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0

   /* Alice responds be re-sending the INVITE with authentication
      credentials in it.  The same Call-ID is used, so the CSeq is
      increased. */


   F4 INVITE Alice -> Proxy 1

   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
   Max-Forwards: 70
   Route: <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 2 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Proxy-Authorization: Digest username="alice",
    realm="atlanta.example.com",
    nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
    uri="sip:bob@biloxi.example.com",
    response="42ce3cef44b22f50c6a6071bc8"
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Proxy 1 accepts the credentials and forwards the INVITE to Proxy
   2.  Client for Alice prepares to receive data on port 49172 from the
   network. */
```

F5 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 0
```

F6 INVITE Proxy 1 -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 2 challenges Alice for authentication */
```

F7 407 Proxy Authorization Required Proxy 2 -> Proxy 1

```
SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
 ;received=192.0.2.101
```

```
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=838209
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Proxy-Authenticate: Digest realm="biloxi.example.com", qop="auth",
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0


F8 ACK Proxy 1 -> Proxy 2

ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=838209
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0

/* Proxy 1 forwards the challenge to Alice for authentication from
Proxy 2 */


F9 407 Proxy Authorization Required Proxy 1 -> Alice

SIP/2.0 407 Proxy Authorization Required
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=838209
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Proxy-Authenticate: Digest realm="biloxi.example.com", qop="auth",
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0


F10 ACK Alice -> Proxy 1

ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b21
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=838209
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
```

```
CSeq: 2 ACK
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="42ce3cef44b22f50c6a6071bc8"
Content-Length: 0

/* Alice responds be re-sending the INVITE with authentication
credentials for Proxy 1 AND Proxy 2.  */


F11 INVITE Alice -> Proxy 1

INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="42ce3cef44b22f50c6a6071bc8"
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com", response="f44ab22f150c6a56071bce8"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 1 finds its credentials and authorizes Alice, forwarding the
INVITE to Proxy.  */
```

F12 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Content-Length: 0
```

F13 INVITE Proxy 1 -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com", response="f44ab22f150c6a56071bce8"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Proxy 2 finds its credentials and authorizes Alice, forwarding the
INVITE to Bob.  */
```

F14 100 Trying Proxy 2 -> Proxy 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Content-Length: 0
```

F15 INVITE Proxy 2 -> Bob

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK31972.1
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 68
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Bob answers the call immediately */
```

F16 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK31972.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F17 200 OK Proxy 2 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
```

```
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F18 200 OK Proxy 1 -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F19 ACK Alice -> Proxy 1

ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b44
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>,
 <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 ACK
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="wf84f1ceczx41ae6cbe5aea9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="42ce3cef44b22f50c6a6071bc8"
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
```

```
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com", response="f44ab22f150c6a56071bce8"
Content-Length: 0
```

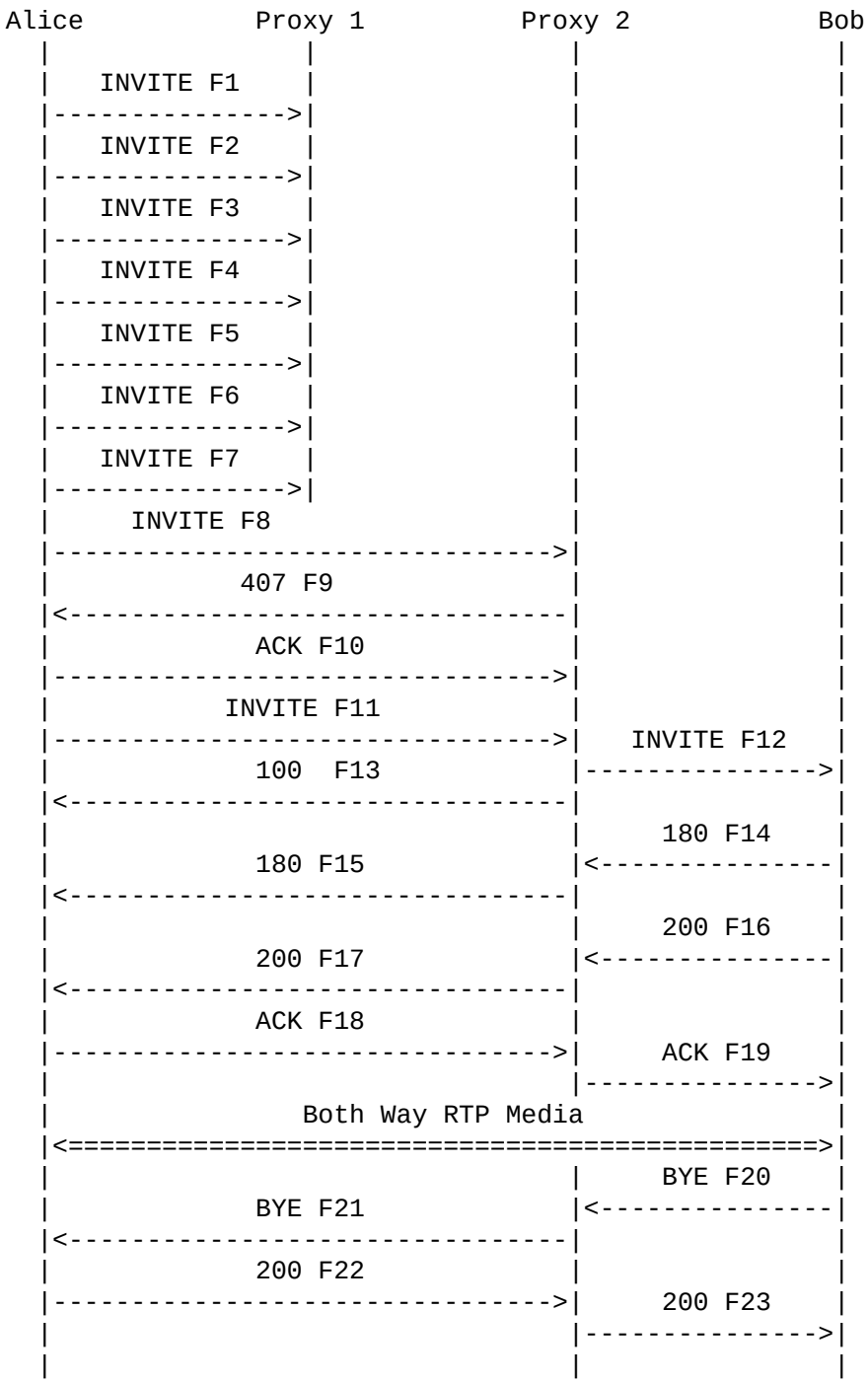
F20 ACK Proxy 1 -> Proxy 2

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b44
 ;received=192.0.2.101
Max-Forwards: 69
Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 ACK
Contact: <sip:bob@client.biloxi.example.com>
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
 nonce="c1e22c41ae6cbe5ae983a9c8e88d359", opaque="",
 uri="sip:bob@biloxi.example.com", response="f44ab22f150c6a56071bce8"
Content-Length: 0
```

F21 ACK Proxy 2 -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK31972.1
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK230f2.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b44
 ;received=192.0.2.101
Max-Forwards: 68
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=9103874
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 3 ACK
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

3.4. Successful Session with Proxy Failure



In this scenario, Alice completes a call to Bob via a Proxy Server. Alice is configured for a primary SIP Proxy Server Proxy 1 and a secondary SIP Proxy Server Proxy 2 (Or is able to use DNS SRV records to locate Proxy 1 and Proxy 2). Alice has valid credentials for both domains. Proxy 1 is out of service and does not respond to INVITEs (it is reachable, but unresponsive). Alice then completes the call to Bob using Proxy 2.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK465b6d
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F2 INVITE Alice -> Proxy 1

Same as Message F1

F3 INVITE Alice -> Proxy 1

Same as Message F1

F4 INVITE Alice -> Proxy 1

Same as Message F1

```
   F5 INVITE Alice -> Proxy 1

   Same as Message F1


   F6 INVITE Alice -> Proxy 1

   Same as Message F1


   F7 INVITE Alice -> Proxy 1

   Same as Message F1

   /* Alice gives up on the unresponsive proxy */


   F8 INVITE Alice -> Proxy 2

   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b8a
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* Proxy 2 challenges Alice for authentication */


   F9 407 Proxy Authorization Required Proxy 2 -> Alice

   SIP/2.0 407 Proxy Authorization Required
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b8a
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=2421452
```

```
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 1 INVITE
Proxy-Authenticate: Digest realm="biloxi.example.com", qop="auth",
 nonce="1ae6cbe5ea9c8e8df84fqnlec434a359",
 opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0


F10 ACK Alice -> Proxy 2

ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b8a
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=2421452
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0

/* Alice responds by re-sending the INVITE with authentication
credentials in it.  */


F11 INVITE Alice -> Proxy 2

INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
 nonce="1ae6cbe5ea9c8e8df84fqnlec434a359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="8a880c919d1a52f20a1593e228adf599"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

```
   /* Proxy 2 accepts the credentials and forwards the INVITE to Bob.
   Client for Alice prepares to receive data on port 49172 from the
   network.
   */


   F12 INVITE Proxy 2 -> Bob

   INVITE sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss2.biloxi.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 2 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F13 100 Trying Proxy 2 -> Alice

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 2 INVITE
   Content-Length: 0


   F14 180 Ringing Bob -> Proxy 2

   SIP/2.0 180 Ringing
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
    ;received=192.0.2.222
```

```
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F15 180 Ringing Proxy 2 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F16 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F17 200 OK Proxy 2 -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F18 ACK Alice -> Proxy 2

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b8g
Max-Forwards: 70
Route: <sip:ss2.biloxi.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 2 ACK
Content-Length: 0
```

F19 ACK Proxy 2 -> Bob

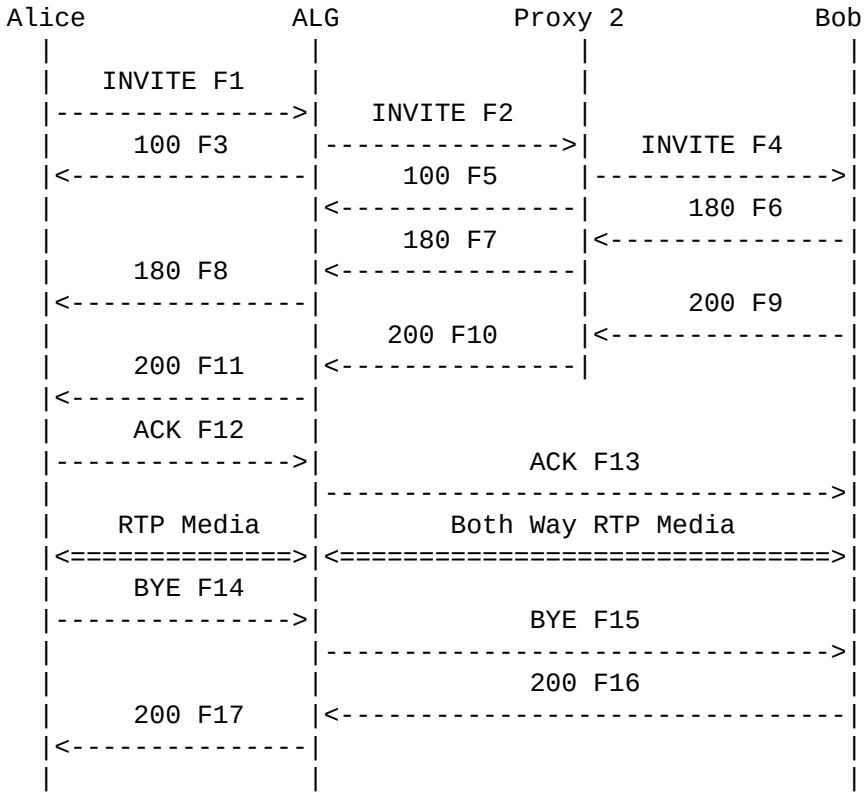
```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b8g
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
```

```
   CSeq: 2 ACK
   Content-Length: 0

   /* RTP streams are established between Alice and Bob */

   /* Bob Hangs Up with Alice. */


   F20 BYE Bob -> Proxy 2

   BYE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
   Max-Forwards: 70
   Route: <sip:ss2.biloxi.example.com;lr>
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0


   F21 BYE Proxy 2 -> Alice

   BYE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
    ;received=192.0.2.201
   Max-Forwards: 69
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0


   F22 200 OK Alice -> Proxy 2

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
    ;received=192.0.2.222
   Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
    ;received=192.0.2.201
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
   CSeq: 1 BYE
   Content-Length: 0
```

F23 200 OK Proxy 2 -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.biloxi.example.com:5060;branch=z9hG4bKnashds7
 ;received=192.0.2.201
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 4Fde34wkd11wsGFDs3@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

3.5. Session Through a SIP ALG



Alice completes a call to Bob through a ALG (Application Layer Gateway) and a SIP Proxy. The routing through the ALG is accomplished using a pre-loaded Route header in the INVITE F1. Note that the media stream setup is not end-to-end - the ALG terminates both media streams and bridges them. This is done by the ALG modifying the SDP in the INVITE (F1) and 200 OK (F10) messages, and possibly any 18x or ACK messages containing SDP.

In addition to firewall traversal, this Back-to-Back User Agent (B2BUA) could be used as part of an anonymizer service (in which all identifying information on Alice would be removed), or to perform codec media conversion, such as mu-law to A-law conversion of PCM on an international call.

Also note that Proxy 2 does not Record-Route in this call flow.

Message Details

F1 INVITE Alice -> SIP ALG

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Route: <sip:alg1.atlanta.example.com;lr>
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
 nonce="85b4f1cen4341ae6cbe5a3a9c8e88df9", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="b3f392f9218a328b9294076d708e6815"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* Client for Alice prepares to receive data on port 49172 from the
network. */
```

F2 INVITE SIP ALG -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="biloxi.example.com",
```

```
    nonce="85b4f1cen4341ae6cbe5a3a9c8e88df9", opaque="",
    uri="sip:bob@biloxi.example.com",
    response="b3f392f9218a328b9294076d708e6815"
   Content-Type: application/sdp
   Content-Length: 150

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.128
   t=0 0
   m=audio 2000 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F3 100 Trying SIP ALG -> Alice

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0

   /* SIP ALG prepares to proxy data from port 192.0.2.128/2000 to
   192.0.2.101/49172.   Proxy 2 uses a Location Service function to
   determine where Bob is located. Based upon location analysis the call
   is forwarded to Bob */


   F4 INVITE Proxy 2 -> Bob

   INVITE sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
    ;received=192.0.2.128
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 68
   Record-Route: <sip:alg1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
```

```
Content-Length: 150

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.128
t=0 0
m=audio 2000 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F5 100 Trying Proxy 2 -> SIP ALG

SIP/2.0 100 Trying
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0


F6 180 Ringing Bob -> Proxy 2

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0


F7 180 Ringing Proxy 2 -> SIP ALG

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
```

```
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F8 180 Ringing SIP ALG -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F9 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
```

```
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F10 200 OK Proxy 2 -> SIP ALG

SIP/2.0 200 OK
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F11 200 OK SIP ALG -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:alg1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.128
t=0 0
```

```
   m=audio 1734 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /* The ALG prepares to proxy packets from 192.0.2.128/
      1734 to 192.0.2.201/3456 */


   F12 ACK Alice -> SIP ALG

   ACK sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bhh
   Max-Forwards: 70
   Route: <sip:alg1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F13 ACK SIP ALG -> Bob

   ACK sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bhh
    ;received=192.0.2.101
   Max-Forwards: 69
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0

   /* RTP streams are established between Alice and the ALG and
   between the ALG and B*/

   /* Alice Hangs Up with Bob. */


   F14 BYE Alice -> SIP ALG

   BYE sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74be5
   Max-Forwards: 70
   Route: <sip:alg1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
```

```
CSeq: 2 BYE
Content-Length: 0
```

F15 BYE SIP ALG -> Bob

```
BYE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74be5
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

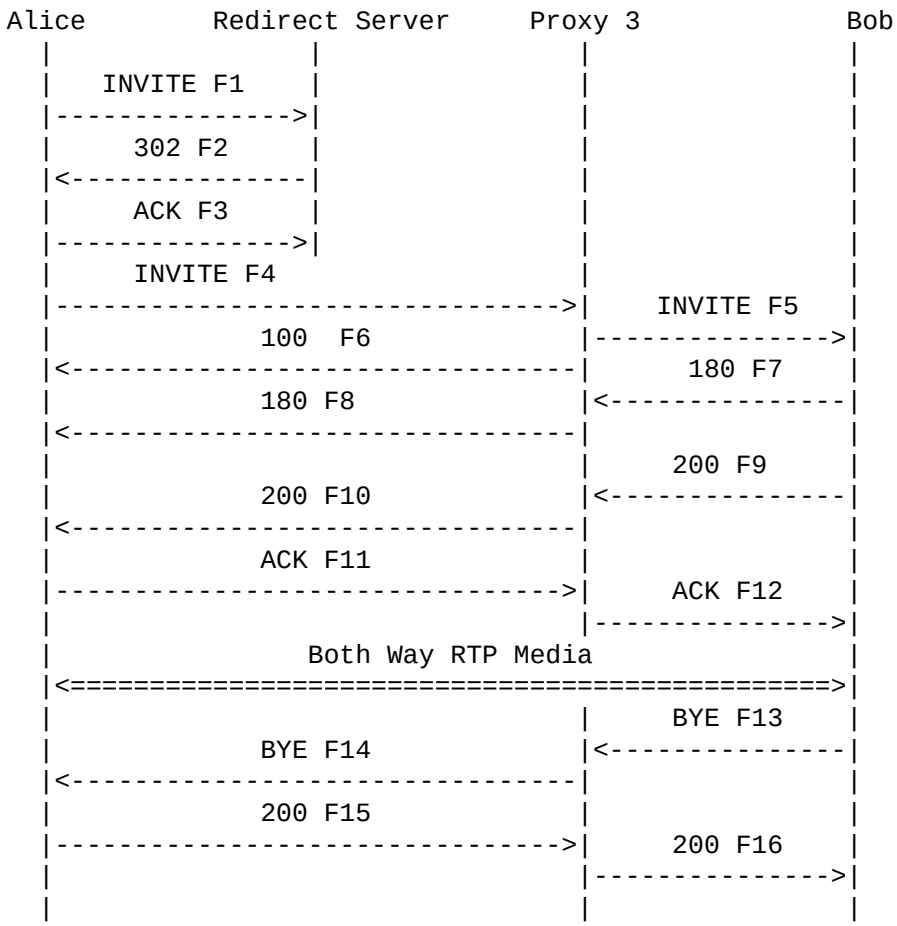
F16 200 OK Bob -> SIP ALG

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP alg1.atlanta.example.com:5060;branch=z9hG4bK739578.1
 ;received=192.0.2.128
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74be5
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

F17 200 OK SIP ALG -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74be5
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

3.6. Session via Redirect and Proxy Servers with SDP in ACK



In this scenario, Alice places a call to Bob using first a Redirect server then a Proxy Server. The INVITE message is first sent to the Redirect Server. The Server returns a 302 Moved Temporarily response (F2) containing a Contact header with Bob's current SIP address. Alice then generates a new INVITE and sends to Bob via the Proxy Server and the call proceeds normally. In this example, no SDP is present in the INVITE, so the SDP is carried in the ACK message.

The call is terminated when Bob sends a BYE message.

Message Details

F1 INVITE Alice -> Redirect Server

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Length: 0
```

F2 302 Moved Temporarily Redirect Proxy -> Alice

```
SIP/2.0 302 Moved Temporarily
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=53fHlqlQ2
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@chicago.example.com;transport=tcp>
Content-Length: 0
```

F3 ACK Alice -> Redirect Server

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bKbf9f44
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=53fHlqlQ2
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

F4 INVITE Alice -> Proxy 3

```
INVITE sip:bob@chicago.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
```

```
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Length: 0
```

F5 INVITE Proxy 3 -> Bob

```
INVITE sip:bob@client.chicago.example.com SIP/2.0
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Length: 0
```

F6 100 Trying Proxy 3 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Content-Length: 0
```

F7 180 Ringing Bob -> Proxy 3

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
 ;received=192.0.2.233
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.chicago.example.com;transport=tcp>
Content-Length: 0
```

F8 180 Ringing Proxy 3 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.chicago.example.com;transport=tcp>
Content-Length: 0
```

F9 200 OK Bob -> Proxy 3

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
 ;received=192.0.2.233
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.chicago.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 148

v=0
o=bob 2890844527 2890844527 IN IP4 client.chicago.example.com
s=-
c=IN IP4 192.0.2.100
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F10 200 OK Proxy -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
```

```
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 INVITE
Contact: <sip:bob@client.chicago.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 148

v=0
o=bob 2890844527 2890844527 IN IP4 client.chicago.example.com
s=-
c=IN IP4 192.0.2.100
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* ACK contains SDP of Alice since none present in INVITE */


F11 ACK Alice -> Proxy 3

ACK sip:bob@client.chicago.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bq9
Max-Forwards: 70
Route: <sip:ss3.chicago.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 ACK
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F12 ACK Proxy 3 -> Bob

ACK sip:bob@client.chicago.example.com SIP/2.0
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bq9
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
```

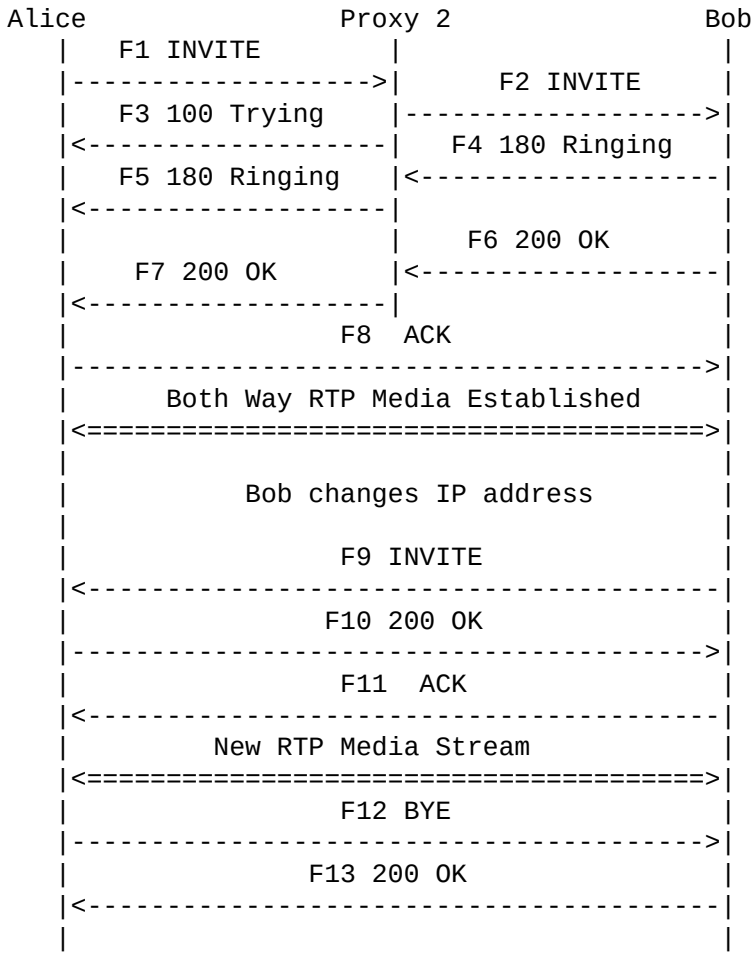
```
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 ACK
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/* RTP streams are established between Alice and Bob */

/* Bob Hangs Up with Alice. */


F13 BYE Bob -> Proxy 3

BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP client.chicago.example.com:5060;branch=z9hG4bKfgaw2
Max-Forwards: 70
Route: <sip:ss3.chicago.example.com;lr>
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0


F14 BYE Proxy 3 -> Alice

BYE sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
 ;received=192.0.2.100
Via: SIP/2.0/TCP client.chicago.example.com:5060;branch=z9hG4bKfgaw2
Max-Forwards: 69
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0


F15 200 OK Alice -> Proxy 3

SIP/2.0 200 OK
```

```
Via: SIP/2.0/TCP ss3.chicago.example.com:5060;branch=z9hG4bK721e.1
 ;received=192.0.2.233
Via: SIP/2.0/TCP client.chicago.example.com:5060;branch=z9hG4bKfgaw2
 ;received=192.0.2.100
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

F16 200 OK Proxy 3 -> Bob

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP client.chicago.example.com:5060;branch=z9hG4bKfgaw2
 ;received=192.0.2.100
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 BYE
Content-Length: 0
```

3.7. Session with re-INVITE (IP Address Change)



This example shows a session in which the media changes midway through the session. When Bob's IP address changes during the session, Bob sends a re-INVITE containing a new Contact and SDP (version number incremented) information to A. In this flow, the proxy does not Record-Route so is not in the SIP messaging path after the initial exchange.

Message Details

F1 INVITE Alice -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F2 INVITE Proxy 2 -> Bob

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F3 100 Trying Proxy 2 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F4 180 Ringing Bob -> Proxy 2

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F5 180 Ringing Proxy 2 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F6 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
```

```
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F7 200 OK Proxy 2 -> Alice

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=bob 2890844527 2890844527 IN IP4 client.biloxi.example.com
s=-
c=IN IP4 192.0.2.201
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

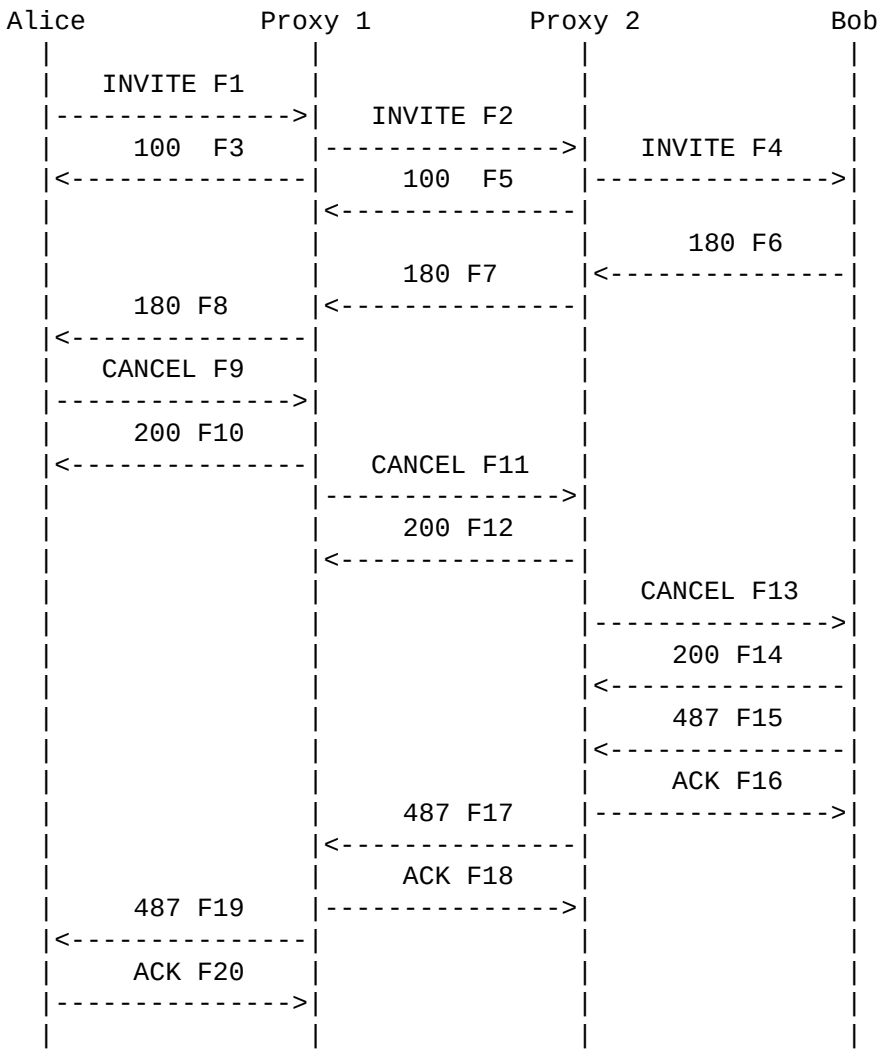
F8 ACK Alice -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74b7b
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

```
   /* RTP streams are established between Alice and Bob */

   /* Bob changes IP address and re-INVITEs Alice with new Contact and
   SDP */


   F9 INVITE Bob -> Alice

   INVITE sip:alice@client.atlanta.example.com SIP/2.0
   Via: SIP/2.0/UDP client.chicago.example.com:5060;branch=z9hG4bKlkld5l
   Max-Forwards: 70
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 14 INVITE
   Contact: <sip:bob@client.chicago.example.com>
   Content-Type: application/sdp
   Content-Length: 149

   v=0
   o=bob 2890844527 2890844528 IN IP4 client.chicago.example.com
   s=-
   c=IN IP4 192.0.2.100
   t=0 0
   m=audio 47172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F10 200 OK Alice -> Bob

   SIP/2.0 200 OK
   Via: SIP/2.0/UDP client.chicago.example.com:5060;branch=z9hG4bKlkld5l
    ;received=192.0.2.100
   Max-Forwards: 70
   From: Bob <sip:bob@biloxi.example.com>;tag=314159
   To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 14 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 150

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
```

```
m=audio 1000 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F11 ACK Bob -> Alice

ACK sip:alice@client.atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP client.chicago.example.com:5060;branch=z9hG4bKlkldcc
Max-Forwards: 70
From: Bob <sip:bob@biloxi.example.com>;tag=314159
To: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 14 ACK
Content-Length: 0

/* New RTP stream established between Alice and Bob */

/* Alice hangs up with Bob */


F12 BYE Alice -> Bob

BYE sip:bob@client.chicago.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bo4
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0


F13 200 OK Bob -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bo4
 ;received=192.0.2.101
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 2 BYE
Content-Length: 0
```

3.8. Unsuccessful No Answer



In this scenario, Alice gives up on the call before Bob answers (sends a 200 OK response). Alice sends a CANCEL (F9) since no final response had been received from Bob. If a 200 OK to the INVITE had crossed with the CANCEL, Alice would have sent an ACK then a BYE to Bob in order to properly terminate the call.

Note that the CANCEL message is acknowledged with a 200 OK on a hop by hop basis, rather than end to end.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="ze7k1ee88df84f1cec431ae6cbe5a359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="b00b416324679d7e243f55708d44be7b"
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*Client for Alice prepares to receive data on port 49172 from the
network.*/
```

F2 INVITE Proxy 1 -> Proxy 2

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151
```

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F3 100 Trying Proxy 1 -> Alice

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F4 INVITE Proxy 2 -> Bob

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
Max-Forwards: 68
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F5 100 Trying Proxy 2 -> Proxy 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F6 180 Ringing Bob -> Proxy 2

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F7 180 Ringing Proxy 2 -> Proxy 1

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
```

```
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0


F8 180 Ringing Proxy 1 -> Alice

SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0


F9 CANCEL Alice -> Proxy 1

CANCEL sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Route: <sip:ss1.atlanta.example.com;lr>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0


F10 200 OK Proxy 1 -> Alice

SIP/2.0 200 OK
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F11 CANCEL Proxy 1 -> Proxy 2

```
CANCEL sip:alice@atlanta.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F12 200 OK Proxy 2 -> Proxy 1

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F13 CANCEL Proxy 2 -> Bob

```
CANCEL sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F14 200 OK Bob -> Proxy 2

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 CANCEL
Content-Length: 0
```

F15 487 Request Terminated Bob -> Proxy 2

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F16 ACK Proxy 2 -> Bob

```
ACK sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

F17 487 Request Terminated Proxy 2 -> Proxy 1

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F18 ACK Proxy 1 -> Proxy 2

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

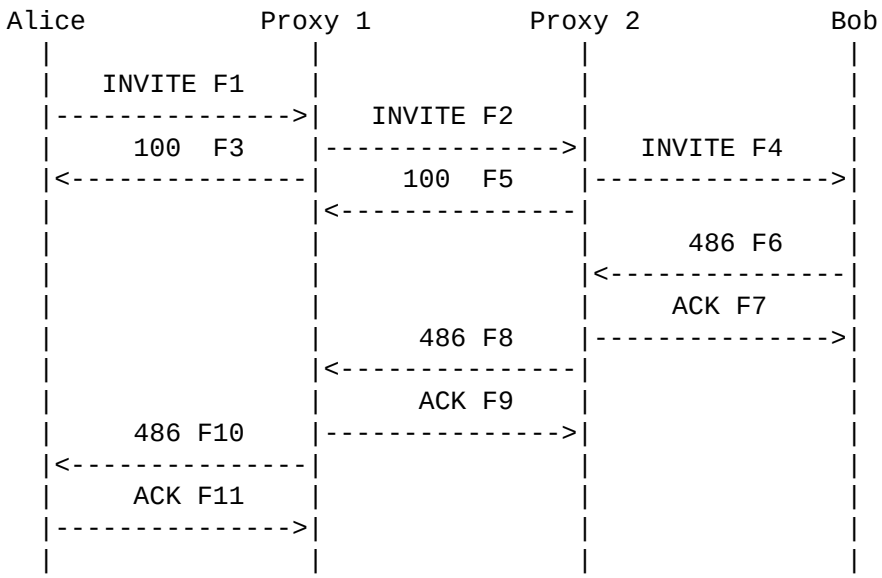
F19 487 Request Terminated Proxy 1 -> Alice

```
SIP/2.0 487 Request Terminated
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
```

F20 ACK Alice -> Proxy 1

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="ze7k1ee88df84f1cec431ae6cbe5a359", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="b00b416324679d7e243f55708d44be7b"
CSeq: 1 ACK
Content-Length: 0
```

3.9. Unsuccessful Busy



In this scenario, Bob is busy and sends a 486 Busy Here response to Alice's INVITE. Note that the non-2xx response is acknowledged on a hop-by-hop basis instead of end-to-end. Also note that many SIP UAs will not return a 486 response, as they have multiple line and other features.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="dc3a5ab2530aa93112cf5904ba7d88fa", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="702138b27d869ac8741e10ec643d55be"
Content-Type: application/sdp
Content-Length: 151
```

```
v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000

/*Client for Alice prepares to receive data on port 49172 from the
network.*/


F2 INVITE Proxy 1 -> Proxy 2

INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 69
Record-Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F3 100 Trying Proxy 1 -> Alice

SIP/2.0 100 Trying
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F4 INVITE Proxy 2 -> Bob

```
INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 68
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com;transport=tcp>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

F5 100 Trying Proxy 2 -> Proxy 1

```
SIP/2.0 100 Trying
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F6 486 Busy Here Bob -> Proxy 2

```
SIP/2.0  486 Busy Here
Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
```

```
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F7 ACK Proxy 2 -> Bob

   ACK sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/TCP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F8 486 Busy Here Proxy 2 -> Proxy 1

   SIP/2.0  486 Busy Here
   Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F9 ACK Proxy 1 -> Proxy 2

   ACK sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/TCP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0
```

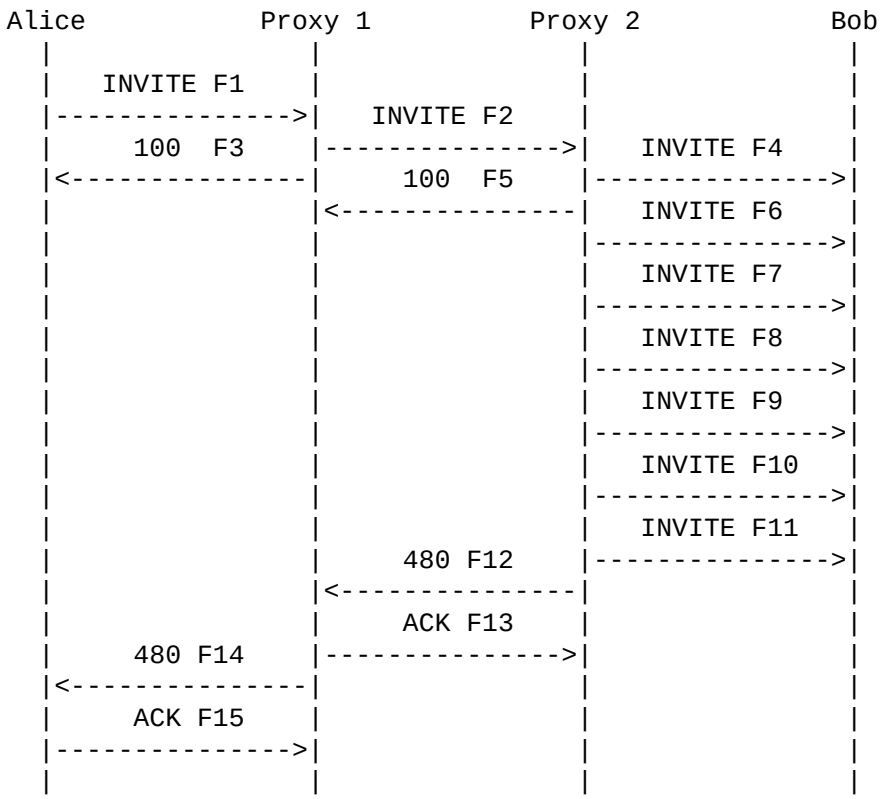
F10 486 Busy Here Proxy 1 -> Alice

```
SIP/2.0  486 Busy Here
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F11 ACK Alice -> Proxy 1

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/TCP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="dc3a5ab2530aa93112cf5904ba7d88fa", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="702138b27d869ac8741e10ec643d55be"
Content-Length: 0
```

3.10. Unsuccessful No Response from User Agent



In this example, there is no response from Bob to Alice's INVITE messages being re-transmitted by Proxy 2. After the sixth re-transmission, Proxy 2 gives up and sends a 480 No Response to Alice.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
```

```
    nonce="cf5904ba7d8dc3a5ab2530aa931128fa", opaque="",
    uri="sip:bob@biloxi.example.com",
    response="7afc04be7961f053c24f80e7dbaf888f"
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /*Client for Alice prepares to receive data on port 49172 from the
   network.*/


   F2 INVITE Proxy 1 -> Proxy 2

   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F3 100 Trying Proxy 1 -> Alice

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
```

```
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0


F4 INVITE Proxy 2 -> Bob

INVITE sip:bob@client.biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Max-Forwards: 68
Record-Route: <sip:ss2.biloxi.example.com;lr>,
<sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000


F5 100 Trying Proxy 2 -> Proxy 1

SIP/2.0 100 Trying
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F6 INVITE Proxy 2 -> Bob

Resend of Message F4

F7 INVITE Proxy 2 -> Bob

Resend of Message F4

F8 INVITE Proxy 2 -> Bob

Resend of Message F4

F9 INVITE Proxy 2 -> Bob

Resend of Message F4

F10 INVITE Proxy 2 -> Bob

Resend of Message F4

F11 INVITE Proxy 2 -> Bob

Resend of Message F4

/* Proxy 2 gives up */

F12 480 No Response Proxy 2 -> Proxy 1

```
SIP/2.0 480 No Response
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F13 ACK Proxy 1 -> Proxy 2

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

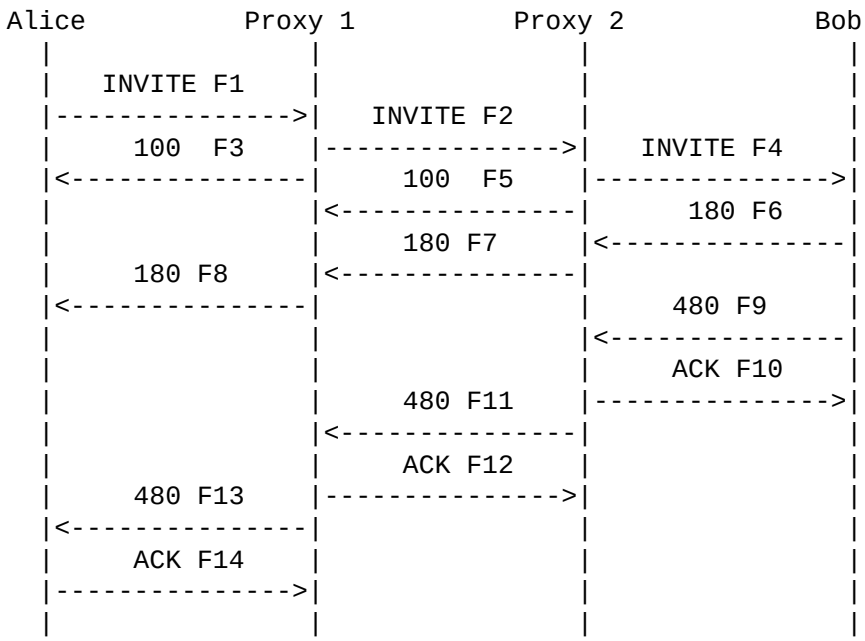
F14 480 No Response Proxy 1 -> Alice

```
SIP/2.0 480 No Response
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F15 ACK Alice -> Proxy 1

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="cf5904ba7d8dc3a5ab2530aa931128fa", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="7afc04be7961f053c24f80e7dbaf888f"
Content-Length: 0
```

3.11. Unsuccessful Temporarily Unavailable



In this scenario, Bob initially sends a 180 Ringing response to Alice, indicating that alerting is taking place. However, then a 480 Unavailable is then sent to Alice. This response is acknowledged then proxied back to Alice.

Message Details

F1 INVITE Alice -> Proxy 1

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
Route: <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@client.atlanta.example.com>
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="aa9311cf5904ba7d8dc3a5ab253028fa", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="59a46a91bf1646562a4d486c84b399db"
Content-Type: application/sdp
```

```
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000

   /*Client for Alice prepares to receive data on port 49172 from the
   network.*/


   F2 INVITE Proxy 1 -> Proxy 2

   INVITE sip:bob@biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 69
   Record-Route: <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F3 100 Trying Proxy 1 -> Alice

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
```

```
   Content-Length: 0


   F4 INVITE Proxy 2 -> Bob

   INVITE sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   Max-Forwards: 68
   Record-Route: <sip:ss2.biloxi.example.com;lr>,
    <sip:ss1.atlanta.example.com;lr>
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Contact: <sip:alice@client.atlanta.example.com>
   Content-Type: application/sdp
   Content-Length: 151

   v=0
   o=alice 2890844526 2890844526 IN IP4 client.atlanta.example.com
   s=-
   c=IN IP4 192.0.2.101
   t=0 0
   m=audio 49172 RTP/AVP 0
   a=rtpmap:0 PCMU/8000


   F5 100 Trying Proxy 2 -> Proxy 1

   SIP/2.0 100 Trying
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0
```

F6 180 Ringing Bob -> Proxy 2

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
 ;received=192.0.2.222
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F7 180 Ringing Proxy 2 -> Proxy 1

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
 ;received=192.0.2.111
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
Content-Length: 0
```

F8 180 Ringing Proxy 1 -> Alice

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
Record-Route: <sip:ss2.biloxi.example.com;lr>,
 <sip:ss1.atlanta.example.com;lr>
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:bob@client.biloxi.example.com>
```

```
   Content-Length: 0


   F9 480 Temporarily Unavailable Bob -> Proxy 2

   SIP/2.0 480 Temporarily Unavailable
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
    ;received=192.0.2.222
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0


   F10 ACK Proxy 2 -> Bob

   ACK sip:bob@client.biloxi.example.com SIP/2.0
   Via: SIP/2.0/UDP ss2.biloxi.example.com:5060;branch=z9hG4bK721e4.1
   Max-Forwards: 70
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 ACK
   Content-Length: 0


   F11 480 Temporarily Unavailable Proxy 2 -> Proxy 1

   SIP/2.0 480 Temporarily Unavailable
   Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
    ;received=192.0.2.111
   Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
    ;received=192.0.2.101
   From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
   To: Bob <sip:bob@biloxi.example.com>;tag=314159
   Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
   CSeq: 1 INVITE
   Content-Length: 0
```

F12 ACK Proxy 1 -> Proxy 2

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP ss1.atlanta.example.com:5060;branch=z9hG4bK2d4790.1
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 ACK
Content-Length: 0
```

F13 480 Temporarily Unavailable Proxy 1 -> Alice

```
SIP/2.0 480 Temporarily Unavailable
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
 ;received=192.0.2.101
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
CSeq: 1 INVITE
Content-Length: 0
```

F14 ACK Alice -> Proxy 1

```
ACK sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP client.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>;tag=314159
Call-ID: 2xTb9vxSit55XU7p8@atlanta.example.com
Proxy-Authorization: Digest username="alice",
 realm="atlanta.example.com",
 nonce="aa9311cf5904ba7d8dc3a5ab253028fa", opaque="",
 uri="sip:bob@biloxi.example.com",
 response="59a46a91bf1646562a4d486c84b399db"
CSeq: 1 ACK
Content-Length: 0
```

4. Security Considerations

Since this document contains examples of SIP session establishment, the security considerations in RFC 3261 [1] apply. RFC 3261 describes the basic threats including registration hijacking, server impersonation, message body tampering, session modifying or teardown, and denial of service and amplification attacks. The use of HTTP Digest as shown in this document provides one-way authentication and protection against replay attacks. TLS transport is used in registration scenarios due to the lack of integrity protection in HTTP Digest and the danger of registration hijacking without it, as described in RFC 3261 [1]. A full discussion of the weaknesses of HTTP Digest is provided in RFC 3261 [1]. The use of TLS and the Secure SIP (sips) URI scheme provides a better level of security including two-way authentication. S/MIME can provide end-to-end confidentiality and integrity protection of message bodies, as described in RFC 3261.

5. References

5.1. Normative References

[1] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[2] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with SDP", RFC 3264, April 2002.

[3] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

5.2. Informative References

[5] Johnston, A., Donovan, S., Sparks, R., Cunningham, C. and K. Summers, "Session Initiation Protocol (SIP) Public Switched Telephone Network (PSTN) Call Flows", BCP 76, RFC 3666, December 2003.

6. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in

this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

7. Acknowledgments

This document is has been a group effort by the SIP and SIPPING WGs. The authors wish to thank everyone who has read, reviewed, commented, or made suggestions to improve this document.

Thanks to Rohan Mahy, Adam Roach, Gonzalo Camarillo, Cullen Jennings, and Tom Taylor for their detailed comments during the final review. Thanks to Dean Willis for his early contributions to the development of this document.

The authors wish to thank Kundan Singh for performing parser validation of messages.

The authors wish to thank the following individuals for their participation in the review of this call flows document: Aseem Agarwal, Rafi Assadi, Ben Campbell, Sunitha Kumar, Jon Peterson, Marc Petit-Huguenin, Vidhi Rastogi, and Bodgey Yin Shaohua.

The authors also wish to thank the following individuals for their assistance: Jean-Francois Mule, Hemant Agrawal, Henry Sinnreich, David Devanatham, Joe Pizzimenti, Matt Cannon, John Hearty, the whole MCI WorldCom IPOP Design team, Scott Orton, Greg Osterhout, Pat Sollee, Doug Weisenberg, Danny Mistry, Steve McKinnon, and Denise Ingram, Denise Caballero, Tom Redman, Ilya Slain, Pat Sollee, John Truetken, and others from MCI WorldCom, 3Com, Cisco, Lucent and Nortel.

8. Authors' Addresses

All listed authors actively contributed large amounts of text to this document.

Alan Johnston
MCI
100 South 4th Street
St. Louis, MO 63102
USA

EMail: alan.johnston@mci.com

Steve Donovan
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: sdonovan@dynamicsoft.com

Robert Sparks
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: rsparks@dynamicsoft.com

Chris Cunningham
dynamicsoft, Inc.
5100 Tennyson Parkway
Suite 1200
Plano, Texas 75024
USA

EMail: ccunningham@dynamicsoft.com

Kevin Summers
Sonus
1701 North Collins Blvd, Suite 3000
Richardson, TX 75080
USA

EMail: kevin.summers@sonusnet.com

9. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

Network Working Group
Request for Comments: 2327
Category: Standards Track

M. Handley
V. Jacobson
ISI/LBNL

April 1998

SDP: Session Description Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document defines the Session Description Protocol, SDP. SDP is intended for describing multimedia sessions for the purposes of session announcement, session invitation, and other forms of multimedia session initiation.

This document is a product of the Multiparty Multimedia Session Control (MMUSIC) working group of the Internet Engineering Task Force. Comments are solicited and should be addressed to the working group's mailing list at confctrl@isi.edu and/or the authors.

1. Introduction

On the Internet multicast backbone (Mbone), a session directory tool is used to advertise multimedia conferences and communicate the conference addresses and conference tool-specific information necessary for participation. This document defines a session description protocol for this purpose, and for general real-time multimedia session description purposes. This memo does not describe multicast address allocation or the distribution of SDP messages in detail. These are described in accompanying memos. SDP is not intended for negotiation of media encodings.

2. Background

The Mbone is the part of the internet that supports IP multicast, and thus permits efficient many-to-many communication. It is used extensively for multimedia conferencing. Such conferences usually have the property that tight coordination of conference membership is not necessary; to receive a conference, a user at an Mbone site only has to know the conference's multicast group address and the UDP ports for the conference data streams.

Session directories assist the advertisement of conference sessions and communicate the relevant conference setup information to prospective participants. SDP is designed to convey such information to recipients. SDP is purely a format for session description - it does not incorporate a transport protocol, and is intended to use different transport protocols as appropriate including the Session Announcement Protocol [4], Session Initiation Protocol [11], Real-Time Streaming Protocol [12], electronic mail using the MIME extensions, and the Hypertext Transport Protocol.

SDP is intended to be general purpose so that it can be used for a wider range of network environments and applications than just multicast session directories. However, it is not intended to support negotiation of session content or media encodings - this is viewed as outside the scope of session description.

3. Glossary of Terms

The following terms are used in this document, and have specific meaning within the context of this document.

Conference
: A multimedia conference is a set of two or more communicating users along with the software they are using to communicate.

Session
: A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session.

Session Advertisement
: See session announcement.

Session Announcement
: A session announcement is a mechanism by which a session description is conveyed to users in a proactive fashion, i.e., the session description was not explicitly requested by the user.

Session Description
A well defined format for conveying sufficient information to discover and participate in a multimedia session.

3.1. Terminology

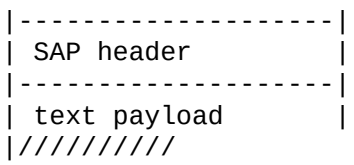
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

4. SDP Usage

4.1. Multicast Announcements

SDP is a session description protocol for multimedia sessions. A common mode of usage is for a client to announce a conference session by periodically multicasting an announcement packet to a well known multicast address and port using the Session Announcement Protocol (SAP).

SAP packets are UDP packets with the following format:



The header is the Session Announcement Protocol header. SAP is described in more detail in a companion memo [4]

The text payload is an SDP session description, as described in this memo. The text payload should be no greater than 1 Kbyte in length. If announced by SAP, only one session announcement is permitted in a single packet.

4.2. Email and WWW Announcements

Alternative means of conveying session descriptions include electronic mail and the World Wide Web. For both email and WWW distribution, the use of the MIME content type "application/sdp" should be used. This enables the automatic launching of applications for participation in the session from the WWW client or mail reader in a standard manner.

Note that announcements of multicast sessions made only via email or the World Wide Web (WWW) do not have the property that the receiver of a session announcement can necessarily receive the session because the multicast sessions may be restricted in scope, and access to the WWW server or reception of email is possible outside this scope. SAP announcements do not suffer from this mismatch.

5. Requirements and Recommendations

The purpose of SDP is to convey information about media streams in multimedia sessions to allow the recipients of a session description to participate in the session. SDP is primarily intended for use in an internetwork, although it is sufficiently general that it can describe conferences in other network environments.

A multimedia session, for these purposes, is defined as a set of media streams that exist for some duration of time. Media streams can be many-to-many. The times during which the session is active need not be continuous.

Thus far, multicast based sessions on the Internet have differed from many other forms of conferencing in that anyone receiving the traffic can join the session (unless the session traffic is encrypted). In such an environment, SDP serves two primary purposes. It is a means to communicate the existence of a session, and is a means to convey sufficient information to enable joining and participating in the session. In a unicast environment, only the latter purpose is likely to be relevant.

Thus SDP includes:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)

As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the conference
- Contact information for the person responsible for the session

In general, SDP must convey sufficient information to be able to join a session (with the possible exception of encryption keys) and to announce the resources to be used to non-participants that may need to know.

5.1. Media Information

SDP includes:

o The type of media (video, audio, etc)

o The transport protocol (RTP/UDP/IP, H.320, etc)

o The format of the media (H.261 video, MPEG video, etc)

For an IP multicast session, the following are also conveyed:

o Multicast address for media

o Transport Port for media

This address and port are the destination address and destination port of the multicast stream, whether being sent, received, or both.

For an IP unicast session, the following are conveyed:

o Remote address for media

o Transport port for contact address

The semantics of this address and port depend on the media and transport protocol defined. By default, this is the remote address and remote port to which data is sent, and the remote address and local port on which to receive data. However, some media may define to use these to establish a control channel for the actual media flow.

5.2. Timing Information

Sessions may either be bounded or unbounded in time. Whether or not they are bounded, they may be only active at specific times.

SDP can convey:

o An arbitrary list of start and stop times bounding the session

o For each bound, repeat times such as "every Wednesday at 10am for one hour"

This timing information is globally consistent, irrespective of local time zone or daylight saving time.

5.3. Private Sessions

It is possible to create both public sessions and private sessions. Private sessions will typically be conveyed by encrypting the session description to distribute it. The details of how encryption is performed are dependent on the mechanism used to convey SDP - see [4] for how this is done for session announcements.

If a session announcement is private it is possible to use that private announcement to convey encryption keys necessary to decode each of the media in a conference, including enough information to know which encryption scheme is used for each media.

5.4. Obtaining Further Information about a Session

A session description should convey enough information to decide whether or not to participate in a session. SDP may include additional pointers in the form of Universal Resources Identifiers (URIs) for more information about the session.

5.5. Categorisation

When many session descriptions are being distributed by SAP or any other advertisement mechanism, it may be desirable to filter announcements that are of interest from those that are not. SDP supports a categorisation mechanism for sessions that is capable of being automated.

5.6. Internationalization

The SDP specification recommends the use of the ISO 10646 character sets in the UTF-8 encoding (RFC 2044) to allow many different languages to be represented. However, to assist in compact representations, SDP also allows other character sets such as ISO 8859-1 to be used when desired. Internationalization only applies to free-text fields (session name and background information), and not to SDP as a whole.

6. SDP Specification

SDP session descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attributes names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set. The textual form, as opposed to a binary encoding such as ASN/1 or XDR,

was chosen to enhance portability, to enable a variety of transports to be used (e.g, session description in a MIME email message) and to allow flexible, text-based toolkits (e.g., Tcl/Tk) to be used to generate and to process session descriptions. However, since the total bandwidth allocated to all SAP announcements is strictly limited, the encoding is deliberately compact. Also, since announcements may be transported via very unreliable means (e.g., email) or damaged by an intermediate caching server, the encoding was designed with strict order and formatting rules so that most errors would result in malformed announcements which could be detected easily and discarded. This also allows rapid discarding of encrypted announcements for which a receiver does not have the correct key.

An SDP session description consists of a number of lines of text of the form <type>=<value> <type> is always exactly one character and is case-significant. <value> is a structured text string whose format depends on <type>. It also will be case-significant unless a specific field defines otherwise. Whitespace is not permitted either side of the '=' sign. In general <value> is either a number of fields delimited by a single space character or a free format string.

A session description consists of a session-level description (details that apply to the whole session and all media streams) and optionally several media-level descriptions (details that apply onto to a single media stream).

An announcement consists of a session-level section followed by zero or more media-level sections. The session-level part starts with a 'v=' line and continues to the first media-level section. The media description starts with an 'm=' line and continues to the next media description or end of the whole session description. In general, session-level values are the default for all media unless overridden by an equivalent media-level value.

When SDP is conveyed by SAP, only one session description is allowed per packet. When SDP is conveyed by other means, many SDP session descriptions may be concatenated together (the 'v=' line indicating the start of a session description terminates the previous description). Some lines in each description are required and some are optional but all must appear in exactly the order given here (the fixed order greatly enhances error detection and allows for a simple parser). Optional items are marked with a '*'.

```
Session description
        v=  (protocol version)
        o=  (owner/creator and session identifier).
        s=  (session name)
        i=* (session information)
```

```
        u=* (URI of description)
        e=* (email address)
        p=* (phone number)
        c=* (connection information - not required if included in all media)
        b=* (bandwidth information)
        One or more time descriptions (see below)
        z=* (time zone adjustments)
        k=* (encryption key)
        a=* (zero or more session attribute lines)
        Zero or more media descriptions (see below)

Time description
        t=  (time the session is active)
        r=* (zero or more repeat times)

Media description
        m=  (media name and transport address)
        i=* (media title)
        c=* (connection information - optional if included at session-level)
        b=* (bandwidth information)
        k=* (encryption key)
        a=* (zero or more media attribute lines)
```

The set of ‘type’ letters is deliberately small and not intended to be extensible -- SDP parsers must completely ignore any announcement that contains a ‘type’ letter that it does not understand. The ‘attribute’ mechanism ("a=" described below) is the primary means for extending SDP and tailoring it to particular applications or media. Some attributes (the ones listed in this document) have a defined meaning but others may be added on an application-, media- or session-specific basis. A session directory must ignore any attribute it doesn’t understand.

The connection (‘c=’) and attribute (‘a=’) information in the session-level section applies to all the media of that session unless overridden by connection information or an attribute of the same name in the media description. For instance, in the example below, each media behaves as if it were given a ‘recvonly’ attribute.

An example SDP description is:

```
        v=0
        o=mhandley 2890844526 2890842807 IN IP4 126.16.64.4
        s=SDP Seminar
        i=A Seminar on the session description protocol
        u=http://www.cs.ucl.ac.uk/staff/M.Handley/sdp.03.ps
        e=mjh@isi.edu (Mark Handley)
        c=IN IP4 224.2.17.12/127
```

```
t=2873397496 2873404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 31
m=application 32416 udp wb
a=orient:portrait
```

Text records such as the session name and information are bytes strings which may contain any byte with the exceptions of 0x00 (Nul), 0x0a (ASCII newline) and 0x0d (ASCII carriage return). The sequence CRLF (0x0d0a) is used to end a record, although parsers should be tolerant and also accept records terminated with a single newline character. By default these byte strings contain ISO-10646 characters in UTF-8 encoding, but this default may be changed using the 'charset' attribute.

Protocol Version

v=0

The "v=" field gives the version of the Session Description Protocol. There is no minor version number.

Origin

o=<username> <session id> <version> <network type> <address type> <address>

The "o=" field gives the originator of the session (their username and the address of the user's host) plus a session id and session version number.

<username> is the user's login on the originating host, or it is "-" if the originating host does not support the concept of user ids. <username> must not contain spaces. <session id> is a numeric string such that the tuple of <username>, <session id>, <network type>, <address type> and <address> form a globally unique identifier for the session.

The method of <session id> allocation is up to the creating tool, but it has been suggested that a Network Time Protocol (NTP) timestamp be used to ensure uniqueness [1].

<version> is a version number for this announcement. It is needed for proxy announcements to detect which of several announcements for the same session is the most recent. Again its usage is up to the

creating tool, so long as <version> is increased when a modification is made to the session data. Again, it is recommended (but not mandatory) that an NTP timestamp is used.

<network type> is a text string giving the type of network. Initially "IN" is defined to have the meaning "Internet". <address type> is a text string giving the type of the address that follows. Initially "IP4" and "IP6" are defined. <address> is the globally unique address of the machine from which the session was created. For an address type of IP4, this is either the fully-qualified domain name of the machine, or the dotted-decimal representation of the IP version 4 address of the machine. For an address type of IP6, this is either the fully-qualified domain name of the machine, or the compressed textual representation of the IP version 6 address of the machine. For both IP4 and IP6, the fully-qualified domain name is the form that SHOULD be given unless this is unavailable, in which case the globally unique address may be substituted. A local IP address MUST NOT be used in any context where the SDP description might leave the scope in which the address is meaningful.

In general, the "o=" field serves as a globally unique identifier for this version of this session description, and the subfields excepting the version taken together identify the session irrespective of any modifications.

Session Name

s=<session name>

The "s=" field is the session name. There must be one and only one "s=" field per session description, and it must contain ISO 10646 characters (but see also the ‘charset’ attribute below).

Session and Media Information

i=<session description>

The "i=" field is information about the session. There may be at most one session-level "i=" field per session description, and at most one "i=" field per media. Although it may be omitted, this is discouraged for session announcements, and user interfaces for composing sessions should require text to be entered. If it is present it must contain ISO 10646 characters (but see also the ‘charset’ attribute below).

A single "i=" field can also be used for each media definition. In media definitions, "i=" fields are primarily intended for labeling media streams. As such, they are most likely to be useful when a

single session has more than one distinct media stream of the same media type. An example would be two different whiteboards, one for slides and one for feedback and questions.

URI

u=<URI>

o A URI is a Universal Resource Identifier as used by WWW clients

o The URI should be a pointer to additional information about the conference

o This field is optional, but if it is present it should be specified before the first media field

o No more than one URI field is allowed per session description

Email Address and Phone Number

e=<email address>
p=<phone number>

o These specify contact information for the person responsible for the conference. This is not necessarily the same person that created the conference announcement.

o Either an email field or a phone field must be specified. Additional email and phone fields are allowed.

o If these are present, they should be specified before the first media field.

o More than one email or phone field can be given for a session description.

o Phone numbers should be given in the conventional international

format - preceded by a "+ and the international country code. There must be a space or a hyphen ("-") between the country code and the rest of the phone number. Spaces and hyphens may be used to split up a phone field to aid readability if desired. For example:

p=+44-171-380-7777 or p=+1 617 253 6011

o Both email addresses and phone numbers can have an optional free text string associated with them, normally giving the name of the person who may be contacted. This should be enclosed in parenthesis if it is present. For example:

e=mjh@isi.edu (Mark Handley)

The alternative RFC822 name quoting convention is also allowed for both email addresses and phone numbers. For example,

e=Mark Handley <mjh@isi.edu>

The free text string should be in the ISO-10646 character set with UTF-8 encoding, or alternatively in ISO-8859-1 or other encodings if the appropriate charset session-level attribute is set.

Connection Data

c=<network type> <address type> <connection address>

The "c=" field contains connection data.

A session announcement must contain one "c=" field in each media description (see below) or a "c=" field at the session-level. It may contain a session-level "c=" field and one additional "c=" field per media description, in which case the per-media values override the session-level settings for the relevant media.

The first sub-field is the network type, which is a text string giving the type of network. Initially "IN" is defined to have the meaning "Internet".

The second sub-field is the address type. This allows SDP to be used for sessions that are not IP based. Currently only IP4 is defined.

The third sub-field is the connection address. Optional extra subfields may be added after the connection address depending on the value of the <address type> field.

For IP4 addresses, the connection address is defined as follows:

o Typically the connection address will be a class-D IP multicast

group address. If the session is not multicast, then the connection address contains the fully-qualified domain name or the unicast IP address of the expected data source or data relay or data sink as determined by additional attribute fields. It is not expected that fully-qualified domain names or unicast addresses

will be given in a session description that is communicated by a multicast announcement, though this is not prohibited. If a unicast data stream is to pass through a network address translator, the use of a fully-qualified domain name rather than an unicast IP address is RECOMMENDED. In other cases, the use of an IP address to specify a particular interface on a multi-homed host might be required. Thus this specification leaves the decision as to which to use up to the individual application, but all applications MUST be able to cope with receiving both formats.

o Conferences using an IP multicast connection address must also have a time to live (TTL) value present in addition to the multicast address. The TTL and the address together define the scope with which multicast packets sent in this conference will be sent. TTL values must be in the range 0-255.

  The TTL for the session is appended to the address using a slash as a separator. An example is:

```
c=IN IP4 224.2.1.1/127
```

  Hierarchical or layered encoding schemes are data streams where the encoding from a single media source is split into a number of layers. The receiver can choose the desired quality (and hence bandwidth) by only subscribing to a subset of these layers. Such layered encodings are normally transmitted in multiple multicast groups to allow multicast pruning. This technique keeps unwanted traffic from sites only requiring certain levels of the hierarchy. For applications requiring multiple multicast groups, we allow the following notation to be used for the connection address:

```
<base multicast address>/<ttl>/<number of addresses>
```

  If the number of addresses is not given it is assumed to be one. Multicast addresses so assigned are contiguously allocated above the base address, so that, for example:

```
c=IN IP4 224.2.1.1/127/3
```

  would state that addresses 224.2.1.1, 224.2.1.2 and 224.2.1.3 are to be used at a ttl of 127. This is semantically identical to including multiple "c=" lines in a media description:

```
c=IN IP4 224.2.1.1/127
c=IN IP4 224.2.1.2/127
c=IN IP4 224.2.1.3/127
```

Multiple addresses or "c=" lines can only be specified on a per-media basis, and not for a session-level "c=" field.

It is illegal for the slash notation described above to be used for IP unicast addresses.

Bandwidth

b=<modifier>:<bandwidth-value>

o This specifies the proposed bandwidth to be used by the session or media, and is optional.

o <bandwidth-value> is in kilobits per second

o <modifier> is a single alphanumeric word giving the meaning of the bandwidth figure.

o Two modifiers are initially defined:

CT Conference Total: An implicit maximum bandwidth is associated with each TTL on the Mbone or within a particular multicast administrative scope region (the Mbone bandwidth vs. TTL limits are given in the MBone FAQ). If the bandwidth of a session or media in a session is different from the bandwidth implicit from the scope, a ‘b=CT:...’ line should be supplied for the session giving the proposed upper limit to the bandwidth used. The primary purpose of this is to give an approximate idea as to whether two or more conferences can co-exist simultaneously.

AS Application-Specific Maximum: The bandwidth is interpreted to be application-specific, i.e., will be the application’s concept of maximum bandwidth. Normally this will coincide with what is set on the application’s "maximum bandwidth" control if applicable.

Note that CT gives a total bandwidth figure for all the media at all sites. AS gives a bandwidth figure for a single media at a single site, although there may be many sites sending simultaneously.

o Extension Mechanism: Tool writers can define experimental bandwidth modifiers by prefixing their modifier with "X-". For example:

b=X-YZ:128

SDP parsers should ignore bandwidth fields with unknown modifiers. Modifiers should be alpha-numeric and, although no length limit is given, they are recommended to be short.

Times, Repeat Times and Time Zones

t=<start time> <stop time>

o "t=" fields specify the start and stop times for a conference session. Multiple "t=" fields may be used if a session is active at multiple irregularly spaced times; each additional "t=" field specifies an additional period of time for which the session will be active. If the session is active at regular times, an "r=" field (see below) should be used in addition to and following a "t=" field - in which case the "t=" field specifies the start and stop times of the repeat sequence.

o The first and second sub-fields give the start and stop times for the conference respectively. These values are the decimal representation of Network Time Protocol (NTP) time values in seconds [1]. To convert these values to UNIX time, subtract decimal 2208988800.

o If the stop-time is set to zero, then the session is not bounded, though it will not become active until after the start-time. If the start-time is also zero, the session is regarded as permanent.

User interfaces should strongly discourage the creation of unbounded and permanent sessions as they give no information about when the session is actually going to terminate, and so make scheduling difficult.

The general assumption may be made, when displaying unbounded sessions that have not timed out to the user, that an unbounded session will only be active until half an hour from the current time or the session start time, whichever is the later. If behaviour other than this is required, an end-time should be given and modified as appropriate when new information becomes available about when the session should really end.

Permanent sessions may be shown to the user as never being active unless there are associated repeat times which state precisely when the session will be active. In general, permanent sessions should not be created for any session expected to have a duration of less than 2 months, and should be discouraged for sessions expected to have a duration of less than 6 months.

r=<repeat interval> <active duration> <list of offsets from start-time>

o "r=" fields specify repeat times for a session. For example, if a session is active at 10am on Monday and 11am on Tuesday for one

hour each week for three months, then the <start time> in the corresponding "t=" field would be the NTP representation of 10am on the first Monday, the <repeat interval> would be 1 week, the <active duration> would be 1 hour, and the offsets would be zero and 25 hours. The corresponding "t=" field stop time would be the NTP representation of the end of the last session three months later. By default all fields are in seconds, so the "r=" and "t=" fields might be:

```
t=3034423619 3042462419
r=604800 3600 0 90000
```

To make announcements more compact, times may also be given in units of days, hours or minutes. The syntax for these is a number immediately followed by a single case-sensitive character. Fractional units are not allowed - a smaller unit should be used instead. The following unit specification characters are allowed:

```
d - days (86400 seconds)
h - minutes (3600 seconds)
m - minutes (60 seconds)
s - seconds (allowed for completeness but not recommended)
```

Thus, the above announcement could also have been written:

```
r=7d 1h 0 25h
```

Monthly and yearly repeats cannot currently be directly specified with a single SDP repeat time - instead separate "t" fields should be used to explicitly list the session times.

```
z=<adjustment time> <offset> <adjustment time> <offset> ....
```

o To schedule a repeated session which spans a change from daylight-saving time to standard time or vice-versa, it is necessary to specify offsets from the base repeat times. This is required because different time zones change time at different times of day, different countries change to or from daylight time on different dates, and some countries do not have daylight saving time at all.

Thus in order to schedule a session that is at the same time winter and summer, it must be possible to specify unambiguously by whose time zone a session is scheduled. To simplify this task for receivers, we allow the sender to specify the NTP time that a time zone adjustment happens and the offset from the time when the session was first scheduled. The "z" field allows the sender to specify a list of these adjustment times and offsets from the base time.

An example might be:

z=2882844526 -1h 2898848070 0

This specifies that at time 2882844526 the time base by which the session's repeat times are calculated is shifted back by 1 hour, and that at time 2898848070 the session's original time base is restored. Adjustments are always relative to the specified start time - they are not cumulative.

o If a session is likely to last several years, it is expected that the session announcement will be modified periodically rather than transmit several years worth of adjustments in one announcement.

Encryption Keys

k=<method>
k=<method>:<encryption key>

o The session description protocol may be used to convey encryption keys. A key field is permitted before the first media entry (in which case it applies to all media in the session), or for each media entry as required.

o The format of keys and their usage is outside the scope of this document, but see [3].

o The method indicates the mechanism to be used to obtain a usable key by external means, or from the encoded encryption key given.

The following methods are defined:

k=clear:<encryption key>
The encryption key (as described in [3] for RTP media streams under the AV profile) is included untransformed in this key field.

k=base64:<encoded encryption key>
The encryption key (as described in [3] for RTP media streams under the AV profile) is included in this key field but has been base64 encoded because it includes characters that are prohibited in SDP.

k=uri:<URI to obtain key>
A Universal Resource Identifier as used by WWW clients is included in this key field. The URI refers to the data containing the key, and may require additional authentication

before the key can be returned. When a request is made to the given URI, the MIME content-type of the reply specifies the encoding for the key in the reply. The key should not be obtained until the user wishes to join the session to reduce synchronisation of requests to the WWW server(s).

k=prompt
No key is included in this SDP description, but the session or media stream referred to by this key field is encrypted. The user should be prompted for the key when attempting to join the session, and this user-supplied key should then be used to decrypt the media streams.

Attributes

a=<attribute>
a=<attribute>:<value>

Attributes are the primary means for extending SDP. Attributes may be defined to be used as "session-level" attributes, "media-level" attributes, or both.

A media description may have any number of attributes ("a=" fields) which are media specific. These are referred to as "media-level" attributes and add information about the media stream. Attribute fields can also be added before the first media field; these "session-level" attributes convey additional information that applies to the conference as a whole rather than to individual media; an example might be the conference's floor control policy.

Attribute fields may be of two forms:

o property attributes. A property attribute is simply of the form "a=<flag>". These are binary attributes, and the presence of the attribute conveys that the attribute is a property of the session. An example might be "a=recvonly".

o value attributes. A value attribute is of the form "a=<attribute>:<value>". An example might be that a whiteboard could have the value attribute "a=orient:landscape"

Attribute interpretation depends on the media tool being invoked. Thus receivers of session descriptions should be configurable in their interpretation of announcements in general and of attributes in particular.

Attribute names must be in the US-ASCII subset of ISO-10646/UTF-8.

Attribute values are byte strings, and MAY use any byte value except 0x00 (Nul), 0x0A (LF), and 0x0D (CR). By default, attribute values are to be interpreted as in ISO-10646 character set with UTF-8 encoding. Unlike other text fields, attribute values are NOT normally affected by the 'charset' attribute as this would make comparisons against known values problematic. However, when an attribute is defined, it can be defined to be charset-dependent, in which case it's value should be interpreted in the session charset rather than in ISO-10646.

Attributes that will be commonly used can be registered with IANA (see Appendix B). Unregistered attributes should begin with "X-" to prevent inadvertent collision with registered attributes. In either case, if an attribute is received that is not understood, it should simply be ignored by the receiver.

Media Announcements

m=<media> <port> <transport> <fmt list>

A session description may contain a number of media descriptions. Each media description starts with an "m=" field, and is terminated by either the next "m=" field or by the end of the session description. A media field also has several sub-fields:

- The first sub-field is the media type. Currently defined media are "audio", "video", "application", "data" and "control", though this list may be extended as new communication modalities emerge (e.g., telepresense). The difference between "application" and "data" is that the former is a media flow such as whiteboard information, and the latter is bulk-data transfer such as multicasting of program executables which will not typically be displayed to the user. "control" is used to specify an additional conference control channel for the session.

- The second sub-field is the transport port to which the media stream will be sent. The meaning of the transport port depends on the network being used as specified in the relevant "c" field and on the transport protocol defined in the third sub-field. Other ports used by the media application (such as the RTCP port, see [2]) should be derived algorithmically from the base media port.

  Note: For transports based on UDP, the value should be in the range 1024 to 65535 inclusive. For RTP compliance it should be an even number.

For applications where hierarchically encoded streams are being sent to a unicast address, it may be necessary to specify multiple transport ports. This is done using a similar notation to that used for IP multicast addresses in the "c=" field:

m=<media> <port>/<number of ports> <transport> <fmt list>

In such a case, the ports used depend on the transport protocol. For RTP, only the even ports are used for data and the corresponding one-higher odd port is used for RTCP. For example:

m=video 49170/2 RTP/AVP 31

would specify that ports 49170 and 49171 form one RTP/RTCP pair and 49172 and 49173 form the second RTP/RTCP pair. RTP/AVP is the transport protocol and 31 is the format (see below).

It is illegal for both multiple addresses to be specified in the "c=" field and for multiple ports to be specified in the "m=" field in the same session description.

o The third sub-field is the transport protocol. The transport protocol values are dependent on the address-type field in the "c=" fields. Thus a "c=" field of IP4 defines that the transport protocol runs over IP4. For IP4, it is normally expected that most media traffic will be carried as RTP over UDP. The following transport protocols are preliminarily defined, but may be extended through registration of new protocols with IANA:

- RTP/AVP - the IETF's Realtime Transport Protocol using the Audio/Video profile carried over UDP.

- udp - User Datagram Protocol

If an application uses a single combined proprietary media format and transport protocol over UDP, then simply specifying the transport protocol as udp and using the format field to distinguish the combined protocol is recommended. If a transport protocol is used over UDP to carry several distinct media types that need to be distinguished by a session directory, then specifying the transport protocol and media format separately is necessary. RTP is an example of a transport-protocol that carries multiple payload formats that must be distinguished by the session directory for it to know how to start appropriate tools, relays, mixers or recorders.

The main reason to specify the transport-protocol in addition to the media format is that the same standard media formats may be carried over different transport protocols even when the network protocol is the same - a historical example is vat PCM audio and RTP PCM audio. In addition, relays and monitoring tools that are transport-protocol-specific but format-independent are possible.

For RTP media streams operating under the RTP Audio/Video Profile [3], the protocol field is "RTP/AVP". Should other RTP profiles be defined in the future, their profiles will be specified in the same way. For example, the protocol field "RTP/XYZ" would specify RTP operating under a profile whose short name is "XYZ".

o The fourth and subsequent sub-fields are media formats. For audio and video, these will normally be a media payload type as defined in the RTP Audio/Video Profile.

When a list of payload formats is given, this implies that all of these formats may be used in the session, but the first of these formats is the default format for the session.

For media whose transport protocol is not RTP or UDP the format field is protocol specific. Such formats should be defined in an additional specification document.

For media whose transport protocol is RTP, SDP can be used to provide a dynamic binding of media encoding to RTP payload type. The encoding names in the RTP AV Profile do not specify unique audio encodings (in terms of clock rate and number of audio channels), and so they are not used directly in SDP format fields. Instead, the payload type number should be used to specify the format for static payload types and the payload type number along with additional encoding information should be used for dynamically allocated payload types.

An example of a static payload type is u-law PCM coded single channel audio sampled at 8KHz. This is completely defined in the RTP Audio/Video profile as payload type 0, so the media field for such a stream sent to UDP port 49232 is:

```
m=video 49232 RTP/AVP 0
```

An example of a dynamic payload type is 16 bit linear encoded stereo audio sampled at 16KHz. If we wish to use dynamic RTP/AVP payload type 98 for such a stream, additional information is required to decode it:

```
m=video 49232 RTP/AVP 98
```

a=rtpmap:98 L16/16000/2

The general form of an rtpmap attribute is:

a=rtpmap:<payload type> <encoding name>/<clock rate>[/<encoding parameters>]

For audio streams, <encoding parameters> may specify the number of audio channels. This parameter may be omitted if the number of channels is one provided no additional parameters are needed. For video streams, no encoding parameters are currently specified.

Additional parameters may be defined in the future, but codecspecific parameters should not be added. Parameters added to an rtpmap attribute should only be those required for a session directory to make the choice of appropriate media too to participate in a session. Codec-specific parameters should be added in other attributes.

Up to one rtpmap attribute can be defined for each media format specified. Thus we might have:

```
m=audio 49230 RTP/AVP 96 97 98
      a=rtpmap:96 L8/8000
     a=rtpmap:97 L16/8000
    a=rtpmap:98 L16/11025/2
```

RTP profiles that specify the use of dynamic payload types must define the set of valid encoding names and/or a means to register encoding names if that profile is to be used with SDP.

Experimental encoding formats can also be specified using rtpmap. RTP formats that are not registered as standard format names must be preceded by "X-". Thus a new experimental redundant audio stream called GSMLPC using dynamic payload type 99 could be specified as:

```
m=video 49232 RTP/AVP 99
a=rtpmap:99 X-GSMLPC/8000
```

Such an experimental encoding requires that any site wishing to receive the media stream has relevant configured state in its session directory to know which tools are appropriate.

Note that RTP audio formats typically do not include information about the number of samples per packet. If a non-default (as defined in the RTP Audio/Video Profile) packetisation is required, the "ptime" attribute is used as given below.

For more details on RTP audio and video formats, see [3].

o Formats for non-RTP media should be registered as MIME content types as described in Appendix B. For example, the LBL whiteboard application might be registered as MIME content-type application/wb with encoding considerations specifying that it operates over UDP, with no appropriate file format. In SDP this would then be expressed using a combination of the "media" field and the "fmt" field, as follows:

m=application 32416 udp wb

Suggested Attributes

The following attributes are suggested. Since application writers may add new attributes as they are required, this list is not exhaustive.

a=cat:<category>
This attribute gives the dot-separated hierarchical category of the session. This is to enable a receiver to filter unwanted sessions by category. It would probably have been a compulsory separate field, except for its experimental nature at this time. It is a session-level attribute, and is not dependent on charset.

a=keywds:<keywords>
Like the cat attribute, this is to assist identifying wanted sessions at the receiver. This allows a receiver to select interesting session based on keywords describing the purpose of the session. It is a session-level attribute. It is a charset dependent attribute, meaning that its value should be interpreted in the charset specified for the session description if one is specified, or by default in ISO 10646/UTF-8.

a=tool:<name and version of tool>
This gives the name and version number of the tool used to create the session description. It is a session-level attribute, and is not dependent on charset.

a=ptime:<packet time>
This gives the length of time in milliseconds represented by the media in a packet. This is probably only meaningful for audio data. It should not be necessary to know ptime to decode RTP or vat audio, and it is intended as a recommendation for the encoding/packetisation of audio. It is a media attribute, and is not dependent on charset.

a=recvonly
This specifies that the tools should be started in receive-only mode where applicable. It can be either a session or media attribute, and is not dependent on charset.

a=sendrecv
This specifies that the tools should be started in send and receive mode. This is necessary for interactive conferences with tools such as wb which defaults to receive only mode. It can be either a session or media attribute, and is not dependent on charset.

a=sendonly
This specifies that the tools should be started in send-only mode. An example may be where a different unicast address is to be used for a traffic destination than for a traffic source. In such a case, two media descriptions may be use, one sendonly and one recvonly. It can be either a session or media attribute, but would normally only be used as a media attribute, and is not dependent on charset.

a=orient:<whiteboard orientation>
Normally this is only used in a whiteboard media specification. It specifies the orientation of a the whiteboard on the screen. It is a media attribute. Permitted values are ‘portrait’, ‘landscape’ and ‘seascape’ (upside down landscape). It is not dependent on charset

a=type:<conference type>
This specifies the type of the conference. Suggested values are ‘broadcast’, ‘meeting’, ‘moderated’, ‘test’ and ‘H332’. ‘recvonly’ should be the default for ‘type:broadcast’ sessions, ‘type:meeting’ should imply ‘sendrecv’ and ‘type:moderated’ should indicate the use of a floor control tool and that the media tools are started so as to "mute" new sites joining the conference.

Specifying the attribute type:H332 indicates that this loosely coupled session is part of a H.332 session as defined in the ITU H.332 specification [10]. Media tools should be started ‘recvonly’.

Specifying the attribute type:test is suggested as a hint that, unless explicitly requested otherwise, receivers can safely avoid displaying this session description to users.

The type attribute is a session-level attribute, and is not dependent on charset.

a=charset:<character set>

This specifies the character set to be used to display the session name and information data. By default, the ISO-10646 character set in UTF-8 encoding is used. If a more compact representation is required, other character sets may be used such as ISO-8859-1 for Northern European languages. In particular, the ISO 8859-1 is specified with the following SDP attribute:

a=charset:ISO-8859-1

This is a session-level attribute; if this attribute is present, it must be before the first media field. The charset specified MUST be one of those registered with IANA, such as ISO-8859-1. The character set identifier is a US-ASCII string and MUST be compared against the IANA identifiers using a case-insensitive comparison. If the identifier is not recognised or not supported, all strings that are affected by it SHOULD be regarded as byte strings.

Note that a character set specified MUST still prohibit the use of bytes 0x00 (Nul), 0x0A (LF) and 0x0d (CR). Character sets requiring the use of these characters MUST define a quoting mechanism that prevents these bytes appearing within text fields.

a=sdplang:<language tag>

This can be a session level attribute or a media level attribute. As a session level attribute, it specifies the language for the session description. As a media level attribute, it specifies the language for any media-level SDP information field associated with that media. Multiple sdplang attributes can be provided either at session or media level if multiple languages in the session description or media use multiple languages, in which case the order of the attributes indicates the order of importance of the various languages in the session or media from most important to least important.

In general, sending session descriptions consisting of multiple languages should be discouraged. Instead, multiple descriptions should be sent describing the session, one in each language. However this is not possible with all transport mechanisms, and so multiple sdplang attributes are allowed although not recommended.

The sdplang attribute value must be a single RFC 1766 language tag in US-ASCII. It is not dependent on the charset attribute. An sdplang attribute SHOULD be specified when a session is of

sufficient scope to cross geographic boundaries where the language of recipients cannot be assumed, or where the session is in a different language from the locally assumed norm.

a=lang:<language tag>
This can be a session level attribute or a media level attribute. As a session level attribute, it specifies the default language for the session being described. As a media level attribute, it specifies the language for that media, overriding any session-level language specified. Multiple lang attributes can be provided either at session or media level if multiple languages if the session description or media use multiple languages, in which case the order of the attributes indicates the order of importance of the various languages in the session or media from most important to least important.

The lang attribute value must be a single RFC 1766 language tag in US-ASCII. It is not dependent on the charset attribute. A lang attribute SHOULD be specified when a session is of sufficient scope to cross geographic boundaries where the language of recipients cannot be assumed, or where the session is in a different language from the locally assumed norm.

a=framerate:<frame rate>
This gives the maximum video frame rate in frames/sec. It is intended as a recommendation for the encoding of video data. Decimal representations of fractional values using the notation "<integer>.<fraction>" are allowed. It is a media attribute, is only defined for video media, and is not dependent on charset.

a=quality:<quality>
This gives a suggestion for the quality of the encoding as an integer value.

The intention of the quality attribute for video is to specify a non-default trade-off between frame-rate and still-image quality. For video, the value in the range 0 to 10, with the following suggested meaning:

10 - the best still-image quality the compression scheme can give.

5 - the default behaviour given no quality suggestion.

0 - the worst still-image quality the codec designer thinks is still usable.

It is a media attribute, and is not dependent on charset.

a=fmtp:<format> <format specific parameters>
This attribute allows parameters that are specific to a particular format to be conveyed in a way that SDP doesn't have to understand them. The format must be one of the formats specified for the media. Format-specific parameters may be any set of parameters required to be conveyed by SDP and given unchanged to the media tool that will use this format.

It is a media attribute, and is not dependent on charset.

6.1. Communicating Conference Control Policy

There is some debate over the way conference control policy should be communicated. In general, the authors believe that an implicit declarative style of specifying conference control is desirable where possible.

A simple declarative style uses a single conference attribute field before the first media field, possibly supplemented by properties such as 'recvonly' for some of the media tools. This conference attribute conveys the conference control policy. An example might be:

a=type:moderated

In some cases, however, it is possible that this may be insufficient to communicate the details of an unusual conference control policy. If this is the case, then a conference attribute specifying external control might be set, and then one or more "media" fields might be used to specify the conference control tools and configuration data for those tools. An example is an ITU H.332 session:

```
c=IN IP4 224.5.6.7
a=type:H332
m=audio 49230 RTP/AVP 0
m=video 49232 RTP/AVP 31
m=application 12349 udp wb
m=control 49234 H323 mc
c=IN IP4 134.134.157.81
```

In this example, a general conference attribute (type:H332) is specified stating that conference control will be provided by an external H.332 tool, and a contact addresses for the H.323 session multipoint controller is given.

In this document, only the declarative style of conference control declaration is specified. Other forms of conference control should specify an appropriate type attribute, and should define the implications this has for control media.

7. Security Considerations

SDP is a session description format that describes multimedia sessions. A session description should not be trusted unless it has been obtained by an authenticated transport protocol from a trusted source. Many different transport protocols may be used to distribute session description, and the nature of the authentication will differ from transport to transport.

One transport that will frequently be used to distribute session descriptions is the Session Announcement Protocol (SAP). SAP provides both encryption and authentication mechanisms but due to the nature of session announcements it is likely that there are many occasions where the originator of a session announcement cannot be authenticated because they are previously unknown to the receiver of the announcement and because no common public key infrastructure is available.

On receiving a session description over an unauthenticated transport mechanism or from an untrusted party, software parsing the session should take a few precautions. Session description contain information required to start software on the receivers system. Software that parses a session description MUST not be able to start other software except that which is specifically configured as appropriate software to participate in multimedia sessions. It is normally considered INAPPROPRIATE for software parsing a session description to start, on a user's system, software that is appropriate to participate in multimedia sessions, without the user first being informed that such software will be started and giving their consent. Thus a session description arriving by session announcement, email, session invitation, or WWW page SHOULD not deliver the user into an {it interactive} multimedia session without the user being aware that this will happen. As it is not always simple to tell whether a session is interactive or not, applications that are unsure should assume sessions are interactive.

In this specification, there are no attributes which would allow the recipient of a session description to be informed to start multimedia tools in a mode where they default to transmitting. Under some circumstances it might be appropriate to define such attributes. If this is done an application parsing a session description containing such attributes SHOULD either ignore them, or inform the user that joining this session will result in the automatic transmission of multimedia data. The default behaviour for an unknown attribute is to ignore it.

Session descriptions may be parsed at intermediate systems such as firewalls for the purposes of opening a hole in the firewall to allow the participation in multimedia sessions. It is considered INAPPROPRIATE for a firewall to open such holes for unicast data streams unless the session description comes in a request from inside the firewall.

For multicast sessions, it is likely that local administrators will apply their own policies, but the exclusive use of "local" or "site-local" administrative scope within the firewall and the refusal of the firewall to open a hole for such scopes will provide separation of global multicast sessions from local ones.

Appendix A: SDP Grammar

This appendix provides an Augmented BNF grammar for SDP. ABNF is defined in RFC 2234.

```
announcement =          proto-version
                        origin-field
                        session-name-field
                        information-field
                        uri-field
                        email-fields
                        phone-fields
                        connection-field
                        bandwidth-fields
                        time-fields
                        key-field
                        attribute-fields
                        media-descriptions

proto-version =         "v=" 1*DIGIT CRLF
                        ;this memo describes version 0

origin-field =          "o=" username space
                        sess-id space sess-version space
                        nettype space addrtype space
                        addr CRLF

session-name-field =    "s=" text CRLF

information-field =     ["i=" text CRLF]

uri-field =             ["u=" uri CRLF]

email-fields =          *("e=" email-address CRLF)

phone-fields =          *("p=" phone-number CRLF)


connection-field =      ["c=" nettype space addrtype space
                        connection-address CRLF]
                        ;a connection field must be present
                        ;in every media description or at the
                        ;session-level


bandwidth-fields =      *("b=" bwtype ":" bandwidth CRLF)
```

```
time-fields =          1*( "t=" start-time space stop-time
                       *(CRLF repeat-fields) CRLF)
                       [zone-adjustments CRLF]

repeat-fields =        "r=" repeat-interval space typed-time
                       1*(space typed-time)

zone-adjustments =     time space ["-"] typed-time
                       *(space time space ["-"] typed-time)

key-field =            ["k=" key-type CRLF]

key-type =             "prompt" |
                       "clear:" key-data |
                       "base64:" key-data |
                       "uri:" uri

key-data =             email-safe | "~" | "

attribute-fields =     *("a=" attribute CRLF)

media-descriptions =   *( media-field
                       information-field
                       *(connection-field)
                       bandwidth-fields
                       key-field
                       attribute-fields )

media-field =          "m=" media space port ["/" integer]
                       space proto 1*(space fmt) CRLF

media =                1*(alpha-numeric)
                       ;typically "audio", "video", "application"
                       ;or "data"

fmt =                  1*(alpha-numeric)
                       ;typically an RTP payload type for audio
                       ;and video media
```

```
   proto =              1*(alpha-numeric)
                        ;typically "RTP/AVP" or "udp" for IP4

   port =               1*(DIGIT)
                        ;should in the range "1024" to "65535" inclusive
                        ;for UDP based media

   attribute =          (att-field ":" att-value) | att-field

   att-field =          1*(alpha-numeric)

   att-value =          byte-string

   sess-id =            1*(DIGIT)
                        ;should be unique for this originating username/host

   sess-version =       1*(DIGIT)
                        ;0 is a new session

   connection-address = multicast-address
                        | addr

   multicast-address =  3*(decimal-uchar ".") decimal-uchar "/" ttl
                        [ "/" integer ]
                        ;multicast addresses may be in the range
                        ;224.0.0.0 to 239.255.255.255

   ttl =                decimal-uchar

   start-time =         time | "0"

   stop-time =          time | "0"

   time =               POS-DIGIT 9*(DIGIT)
                        ;sufficient for 2 more centuries

   repeat-interval =    typed-time
```

```
typed-time =          1*(DIGIT) [fixed-len-time-unit]

fixed-len-time-unit = "d" | "h" | "m" | "s"

bwtype =              1*(alpha-numeric)

bandwidth =           1*(DIGIT)

username =            safe
                      ;pretty wide definition, but doesn't include space

email-address =       email | email "(" email-safe ")" |
                      email-safe "<" email ">"

email =               ;defined in RFC822

uri=                  ;defined in RFC1630

phone-number =        phone | phone "(" email-safe ")" |
                      email-safe "<" phone ">"

phone =               "+" POS-DIGIT 1*(space | "-" | DIGIT)
                      ;there must be a space or hyphen between the
                      ;international code and the rest of the number.

nettype =             "IN"
                      ;list to be extended

addrtype =            "IP4" | "IP6"
                      ;list to be extended

addr =                FQDN | unicast-address

FQDN =                4*(alpha-numeric|"-"|".")
                      ;fully qualified domain name as specified in RFC1035
```

```
unicast-address =     IP4-address | IP6-address


IP4-address =         b1 "." decimal-uchar "." decimal-uchar "." b4
b1 =                  decimal-uchar
                      ;less than "224"; not "0" or "127"
b4 =                  decimal-uchar
                      ;not "0"

IP6-address =         ;to be defined


text =                byte-string
                      ;default is to interpret this as IS0-10646 UTF8
                      ;ISO 8859-1 requires a "a=charset:ISO-8859-1"
                      ;session-level attribute to be used


byte-string =         1*(0x01..0x09|0x0b|0x0c|0x0e..0xff)
                      ;any byte except NUL, CR or LF


decimal-uchar =       DIGIT
                      | POS-DIGIT DIGIT
                      | ("1" 2*(DIGIT))
                      | ("2" ("0"|"1"|"2"|"3"|"4") DIGIT)
                      | ("2" "5" ("0"|"1"|"2"|"3"|"4"|"5"))


integer =             POS-DIGIT *(DIGIT)


alpha-numeric =       ALPHA | DIGIT


DIGIT =               "0" | POS-DIGIT


POS-DIGIT =           "1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"


ALPHA =               "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|
                      "l"|"m"|"n"|"o "|"p"|"q"|"r"|"s"|"t"|"u"|"v"|
                      "w"|"x"|"y"|"z"|"A"|"B"|"C "|"D"|"E"|"F"|"G"|
                      "H"|"I"|"J"|"K"|"L"|"M"|"N"|"O"|"P"|" Q"|"R"|
                      "S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"
```

```
email-safe =        safe | space | tab


safe =              alpha-numeric |
                    "'" | "'" | "-" | "." | "/" | ":" | "?" | """ |
                    "#" | "$" | "&" | "*" | ";" | "=" | "@" | "[" |
                    "]" | "^" | "_" | "`" | "{" | "|" | "}" | "+" |
                    "~" | "


space =             %d32
tab =               %d9
CRLF =              %d13.10
```

Appendix B: Guidelines for registering SDP names with IANA

There are seven field names that may be registered with IANA. Using the terminology in the SDP specification BNF, they are "media", "proto", "fmt", "att-field", "bwtype", "nettype" and "addrtype".

"media" (eg, audio, video, application, data).

Packetized media types, such as those used by RTP, share the namespace used by media types registry [RFC 2048] (i.e. "MIME types"). The list of valid media names is the set of top-level MIME content types. The set of media is intended to be small and not to be extended except under rare circumstances. (The MIME subtype corresponds to the "fmt" parameter below).

"proto"

In general this should be an IETF standards-track transport protocol identifier such as RTP/AVP (rfc 1889 under the rfc 1890 profile).

However, people will want to invent their own proprietary transport protocols. Some of these should be registered as a "fmt" using "udp" as the protocol and some of which probably can't be.

Where the protocol and the application are intimately linked, such as with the LBL whiteboard wb which used a proprietary and special purpose protocol over UDP, the protocol name should be "udp" and the format name that should be registered is "wb". The rules for formats (see below) apply to such registrations.

Where the proprietary transport protocol really carries many different data formats, it is possible to register a new protocol name with IANA. In such a case, an RFC MUST be produced describing the protocol and referenced in the registration. Such an RFC MAY be informational, although it is preferable if it is standards-track.

"fmt"

The format namespace is dependent on the context of the "proto" field, so a format cannot be registered without specifying one or more transport protocols that it applies to.

Formats cover all the possible encodings that might want to be transported in a multimedia session.

For RTP formats that have been assigned static payload types, the payload type number is used. For RTP formats using a dynamic payload type number, the dynamic payload type number is given as the format and an additional "rtpmap" attribute specifies the format and parameters.

For non-RTP formats, any unregistered format name may be registered through the MIME-type registration process [RFC 2048]. The type given here is the MIME subtype only (the top-level MIME content type is specified by the media parameter). The MIME type registration SHOULD reference a standards-track RFC which describes the transport protocol for this media type. If there is an existing MIME type for this format, the MIME registration should be augmented to reference the transport specification for this media type. If there is not an existing MIME type for this format, and there exists no appropriate file format, this should be noted in the encoding considerations as "no appropriate file format".

"att-field" (Attribute names)

Attribute field names MAY be registered with IANA, although this is not compulsory, and unknown attributes are simply ignored.

When an attribute is registered, it must be accompanied by a brief specification stating the following:

o contact name, email address and telephone number

o attribute-name (as it will appear in SDP)

o long-form attribute name in English

o type of attribute (session level, media level, or both)

o whether the attribute value is subject to the charset attribute.

o a one paragraph explanation of the purpose of the attribute.

o a specification of appropriate attribute values for this attribute.

IANA will not sanity check such attribute registrations except to ensure that they do not clash with existing registrations.

Although the above is the minimum that IANA will accept, if the attribute is expected to see widespread use and interoperability is an issue, authors are encouraged to produce a standards-track RFC that specifies the attribute more precisely.

Submitters of registrations should ensure that the specification is in the spirit of SDP attributes, most notably that the attribute is platform independent in the sense that it makes no implicit assumptions about operating systems and does not name specific pieces of software in a manner that might inhibit interoperability.

"bwtype" (bandwidth specifiers)

A proliferation of bandwidth specifiers is strongly discouraged.

New bandwidth specifiers may be registered with IANA. The submission MUST reference a standards-track RFC specifying the semantics of the bandwidth specifier precisely, and indicating when it should be used, and why the existing registered bandwidth specifiers do not suffice.

"nettype" (Network Type)

New network types may be registered with IANA if SDP needs to be used in the context of non-internet environments. Whilst these are not normally the preserve of IANA, there may be circumstances when an Internet application needs to interoperate with a non-internet application, such as when gatewaying an internet telephony call into the PSTN. The number of network types should be small and should be rarely extended. A new network type cannot be registered without registering at least one address type to be used with that network type. A new network type registration MUST reference an RFC which gives details of the network type and address type and specifies how and when they would be used. Such an RFC MAY be Informational.

"addrtype" (Address Type)

New address types may be registered with IANA. An address type is only meaningful in the context of a network type, and any registration of an address type MUST specify a registered network type, or be submitted along with a network type registration. A new address type registration MUST reference an RFC giving details of the syntax of the address type. Such an RFC MAY be Informational. Address types are not expected to be registered frequently.

Registration Procedure

To register a name the above guidelines should be followed regarding the required level of documentation that is required. The registration itself should be sent to IANA. Attribute registrations should include the information given above. Other registrations should include the following additional information:

o contact name, email address and telephone number

o name being registered (as it will appear in SDP)

o long-form name in English

o type of name ("media", "proto", "fmt", "bwtype", "nettype", or "addrtype")

o a one paragraph explanation of the purpose of the registered name.

o a reference to the specification (eg RFC number) of the registered name.

IANA may refer any registration to the IESG or to any appropriate IETF working group for review, and may request revisions to be made before a registration will be made.

Appendix C: Authors' Addresses

Mark Handley
Information Sciences Institute
c/o MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
United States
electronic mail: mjh@isi.edu

Van Jacobson
MS 46a-1121
Lawrence Berkeley Laboratory
Berkeley, CA 94720
United States
electronic mail: van@ee.lbl.gov

Acknowledgments

Many people in the IETF MMUSIC working group have made comments and suggestions contributing to this document. In particular, we would like to thank Eve Schooler, Steve Casner, Bill Fenner, Allison Mankin, Ross Finlayson, Peter Parnes, Joerg Ott, Carsten Bormann, Rob Lanphier and Steve Hanna.

References

[1] Mills, D., "Network Time Protocol (version 3) specification and implementation", RFC 1305, March 1992.

[2] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[3] Schulzrinne, H., "RTP Profile for Audio and Video Conferences with Minimal Control", RFC 1890, January 1996

[4] Handley, M., "SAP - Session Announcement Protocol", Work in Progress.

[5] V. Jacobson, S. McCanne, "vat - X11-based audio teleconferencing tool" vat manual page, Lawrence Berkeley Laboratory, 1994.

[6] The Unicode Consortium, "The Unicode Standard -- Version 2.0", Addison-Wesley, 1996.

[7] ISO/IEC 10646-1:1993. International Standard -- Information technol- ogy -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. Five amendments and a techn- ical corrigendum have been published up to now. UTF-8 is described in Annex R, published as Amendment 2.

[8] Goldsmith, D., and M. Davis, "Using Unicode with MIME", RFC 1641, July 1994.

[9] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

[10] ITU-T Recommendation H.332 (1998): "Multimedia Terminal for Receiving Internet-based H.323 Conferences", ITU, Geneva.

[11] Handley, M., Schooler, E., and H. Schulzrinne, "Session Initiation Protocol (SIP)", Work in Progress.

[12] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

Full Copyright Statement

INTERNATIONAL TELECOMMUNICATION UNION

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Q.931

(05/98)

SERIES Q: SWITCHING AND SIGNALLING

Digital subscriber Signalling System No. 1 – Network layer

ISDN user-network interface layer 3 specification for basic call control

ITU-T Recommendation Q.931

(Previously CCITT Recommendation)

ITU-T Q-SERIES RECOMMENDATIONS

SWITCHING AND SIGNALLING

SIGNALLING IN THE INTERNATIONAL MANUAL SERVICE	Q.1–Q.3
INTERNATIONAL AUTOMATIC AND SEMI-AUTOMATIC WORKING	Q.4–Q.59
FUNCTIONS AND INFORMATION FLOWS FOR SERVICES IN THE ISDN	Q.60–Q.99
CLAUSES APPLICABLE TO ITU-T STANDARD SYSTEMS	Q.100–Q.119
SPECIFICATIONS OF SIGNALLING SYSTEMS No. 4 AND No. 5	Q.120–Q.249
SPECIFICATIONS OF SIGNALLING SYSTEM No. 6	Q.250–Q.309
SPECIFICATIONS OF SIGNALLING SYSTEM R1	Q.310–Q.399
SPECIFICATIONS OF SIGNALLING SYSTEM R2	Q.400–Q.499
DIGITAL EXCHANGES	Q.500–Q.599
INTERWORKING OF SIGNALLING SYSTEMS	Q.600–Q.699
SPECIFICATIONS OF SIGNALLING SYSTEM No. 7	Q.700–Q.849
DIGITAL SUBSCRIBER SIGNALLING SYSTEM No. 1	Q.850–Q.999
General	Q.850–Q.919
Data link layer	Q.920–Q.929
Network layer	**Q.930–Q.939**
User-network management	Q.940–Q.949
Stage 3 description for supplementary services using DSS 1	Q.950–Q.999
PUBLIC LAND MOBILE NETWORK	Q.1000–Q.1099
INTERWORKING WITH SATELLITE MOBILE SYSTEMS	Q.1100–Q.1199
INTELLIGENT NETWORK	Q.1200–Q.1999
BROADBAND ISDN	Q.2000–Q.2999

For further details, please refer to ITU-T List of Recommendations.

ITU-T RECOMMENDATION Q.931

ISDN USER-NETWORK INTERFACE LAYER 3 SPECIFICATION FOR BASIC CALL CONTROL

Summary

This Recommendation specifies the procedures for the establishing, maintaining and clearing of network connections at the ISDN user-network interface. These procedures are defined in terms of messages exchanged over the D-channel of basic and primary rate interface structures. The functions and procedures of this protocol, and the relationship with other layers, are described in general terms in Recommendation Q.930/I.450 [1]. Annex M contains the additional basic call signalling requirement for the support of private network interconnection for VPN applications.

Source

ITU-T Recommendation Q.931 was revised by ITU-T Study Group 11 (1997-2000) and was approved under the WTSC Resolution No. 1 procedure on the 15th of May 1998.

FOREWORD

ITU (International Telecommunication Union) is the United Nations Specialized Agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of the ITU. The ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Conference (WTSC), which meets every four years, establishes the topics for study by the ITU-T Study Groups which, in their turn, produce Recommendations on these topics.

The approval of Recommendations by the Members of the ITU-T is covered by the procedure laid down in WTSC Resolution No. 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation the term *recognized operating agency (ROA)* includes any individual, company, corporation or governmental organization that operates a public correspondence service. The terms *Administration, ROA* and *public correspondence* are defined in the *Constitution of the ITU (Geneva, 1992)*.

INTELLECTUAL PROPERTY RIGHTS

The ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. The ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, the ITU had not received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

CONTENTS

Page

Page

Page

Recommendation Q.931

ISDN USER-NETWORK INTERFACE LAYER 3 SPECIFICATION FOR BASIC CALL CONTROL

(Malaga-Torremolinos, 1984; modified at Helsinki, 1993; revised in 1998)

1 General

This Recommendation specifies the procedures for the establishing, maintaining and clearing of network connections at the ISDN user-network interface. These procedures are defined in terms of messages exchanged over the D-channel of basic and primary rate interface structures. The functions and procedures of this protocol, and the relationship with other layers, are described in general terms in Recommendation Q.930/I.450 [1].

This Recommendation is intended to specify the essential features, procedures, and messages required for call control in the D-channel. However, there are some details of procedure which have not yet been specified, and which will be the subject of further study.

1.1 Scope of this Recommendation

The procedures currently described in this Recommendation are for the control of circuit-switched connections, user-to-user signalling connections and packet-switched connections. The transport of other message-based information flows on the D-channel is a subject for further study and will be included in related Recommendations.

NOTE 1 – The term "layer 3" is used for the functions and protocol described in this Recommendation (see Recommendation Q.930/I.450). The terms "data link layer" and "layer 2" are used interchangeably to refer to the layer immediately below layer 3.

NOTE 2 – Alignment of the functions and protocol with those of OSI network layer is for further study.

1.2 Application to interface structures

The layer 3 procedures apply to the interface structures defined in Recommendation I.412 [2]. They use the functions and services provided by layer 2. The unacknowledged information transfer service is used by layer 3 to provide point-to-multipoint operation as described in 5.2.

The layer 3 procedures request the services of layer 2 and receive information from layer 2 using the primitives defined in Recommendation Q.921 [3]. These primitives are used to illustrate the communication between the protocol layers and are not intended to specify or constrain implementations.

2 Overview of call control

In this Recommendation, the terms "incoming" and "outgoing" are used to describe the call as viewed by the user side of the interface.

In the subclauses which follow, states are defined for circuit-switched calls in 2.1 (call states), for packet-mode access connections in 2.2 (access connection states), for temporary signalling connections in 2.3 (call states) and for the interface in 2.4 (global call reference states).

This clause defines the basic call control states that individual calls may have. These definitions do not apply to the state of the interface itself, any attached equipment, the D-channel, or the logical

links used for signalling on the D-channel. Because several calls may exist simultaneously at a user-network interface, and each call may be in a different state, the state of the interface itself cannot be unambiguously defined.

NOTE – Additional states and SDL diagrams may be defined when new procedures are developed.

A detailed description of the procedures for call control is given in clauses 5, 6, 7, and 8 in terms of:

a) the messages defined in clause 3 which are transferred across the user-network interface; and

b) the information processing and actions that take place at the user side and the network side.

Overview and detailed SDL diagrams for call control of circuit-switched calls are contained in Annex A.

Throughout this Recommendation, references are made to B-channels. For services using H-channels, the references to B-channels should be taken to refer to the appropriate H-channel. Further study may be needed on other enhancements to support such services.

2.1 Circuit-switched calls

This subclause defines the basic call control states for circuit-switched calls. The procedures for call control are given in clause 5.

Annex D contains optional procedures (as an extension to the basic procedures) to allow symmetric signalling. These states are defined in Annex D.

2.1.1 Call states at the user side of the interface

The states which may exist on the user side of the user-network interface are defined in this subclause.

2.1.1.1 null state (U0): No call exists.

2.1.1.2 call initiated (U1): This state exists for an outgoing call, when the user requests call establishment from the network.

2.1.1.3 overlap sending (U2): This state exists for an outgoing call when the user has received acknowledgement of the call establishment request which permits the user to send additional call information to the network in overlap mode.

2.1.1.4 outgoing call proceeding (U3): This state exists for an outgoing call when the user has received acknowledgement that the network has received all call information necessary to effect call establishment.

2.1.1.5 call delivered (U4): This state exists for an outgoing call when the calling user has received an indication that remote user alerting has been initiated.

2.1.1.6 call present (U6): This state exists for an incoming call when the user has received a call establishment request but has not yet responded.

2.1.1.7 call received (U7): This state exists for an incoming call when the user has indicated alerting but has not yet answered.

2.1.1.8 connect request (U8): This state exists for an incoming call when the user has answered the call and is waiting to be awarded the call.

2.1.1.9 incoming call proceeding (U9): This state exists for an incoming call when the user has sent acknowledgement that the user has received all call information necessary to effect call establishment.

2.1.1.10 active (U10): This state exists for an incoming call when the user has received an acknowledgement from the network that the user has been awarded the call. This state exists for an outgoing call when the user has received an indication that the remote user has answered the call.

2.1.1.11 disconnect request (U11): This state exists when the user has requested the network to clear the end-to-end connection (if any) and is waiting for a response.

2.1.1.12 disconnect indication (U12): This state exists when the user has received an invitation to disconnect because the network has disconnected the end-to-end connection (if any).

2.1.1.13 suspend request (U15): This state exists when the user has requested the network to suspend the call and is waiting for a response.

2.1.1.14 resume request (U17): This state exists when the user has requested the network to resume a previously suspended call and is waiting for a response.

2.1.1.15 release request (U19): This state exists when the user has requested the network to release and is waiting for a response.

2.1.1.16 overlap receiving (U25): This state exists for an incoming call when the user has acknowledged the call establishment request from the network and is prepared to receive additional call information (if any) in overlap mode.

2.1.2 Network call states

The call states that may exist on the network side of the user-network interface are defined in this subclause.

2.1.2.1 null state (N0): No call exists.

2.1.2.2 call initiated (N1): This state exists for an outgoing call when the network has received a call establishment request but has not yet responded.

2.1.2.3 overlap sending (N2): This state exists for an outgoing call when the network has acknowledged the call establishment request and is prepared to receive additional call information (if any) in overlap mode.

2.1.2.4 outgoing call proceeding (N3): This state exists for an outgoing call when the network has sent acknowledgement that the network has received all call information necessary to effect call establishment.

2.1.2.5 call delivered (N4): This state exists for an outgoing call when the network has indicated that remote user alerting has been initiated.

2.1.2.6 call present (N6): This state exists for an incoming call when the network has sent a call establishment request but has not yet received a satisfactory response.

2.1.2.7 call received (N7): This state exists for an incoming call when the network has received an indication that the user is alerting but has not yet received an answer.

2.1.2.8 connect request (N8): This state exists for an incoming call when the network has received an answer but the network has not yet awarded the call.

2.1.2.9 incoming call proceeding (N9): This state exists for an incoming call when the network has received acknowledgement that the user has received all call information necessary to effect call establishment.

2.1.2.10 active (N10): This state exists for an incoming call when the network has awarded the call to the called user. This state exists for an outgoing call when the network has indicated that the remote user has answered the call.

2.1.2.11 disconnect request (N11): This state exists when the network has received a request from the user to clear the end-to-end connection (if any).

2.1.2.12 disconnect indication (N12): This state exists when the network has disconnected the end-to-end connection (if any) and has sent an invitation to disconnect the user-network connection.

2.1.2.13 suspend request (N15): This state exists when the network has received a request to suspend the call but has not yet responded.

2.1.2.14 resume request (N17): This state exists when the network has received a request to resume a previously suspended call but has not yet responded.

2.1.2.15 release request (N19): This state exists when the network has requested the user to release and is waiting for a response.

2.1.2.16 call abort (N22): This state exists for an incoming call for the point-to-multipoint configuration when the call is being cleared before any user has been awarded the call.

2.1.2.17 overlap receiving (N25): This state exists for an incoming call when the network has received acknowledgement of the call establishment request which permits the network to send additional call information (if any) in the overlap mode.

2.2 Packet-mode access connections

This subclause defines the basic packet-mode access connection control states for access to the ISDN virtual circuit bearer service (case B). The procedures for access connection control are given in clause 6.

2.2.1 Access connection states at the user side of the interface

The states which may exist on the user side of the user-network interface are defined in this subclause.

2.2.1.1 null state (U0): No access connection exists.

2.2.1.2 call initiated (U1): This state exists for an outgoing access connection when the user requests access connection establishment from the network.

2.2.1.3 outgoing call proceeding (U3): This state exists for an outgoing access connection when the user has received acknowledgement that the network has received all access connection information necessary to effect access connection establishment.

2.2.1.4 call present (U6): This state exists for an incoming access connection when the user has received an access connection establishment request but has not yet responded.

2.2.1.5 call received (U7): This state exists for an incoming access connection when the user has indicated alerting but has not yet answered.

2.2.1.6 connect request (U8): This state exists for an incoming access connection when the user has accepted the access connection and is waiting to be awarded the access connection.

2.2.1.7 incoming call proceeding (U9): This state exists for an incoming access connection when the user has sent acknowledgement that the user has received all access connection information necessary to effect access connection establishment.

2.2.1.8 active (U10): This state exists for an incoming access connection when the user has received an acknowledgement from the network that the user has been awarded the access connection. This state exists for an outgoing access connection when the user has received an indication that the local network has completed the access connection.

2.2.1.9 disconnect request (U11): This state exists when the user has requested the local network to clear the access connection and is waiting for a response.

2.2.1.10 disconnect indication (U12): This state exists when the user has received an invitation to disconnect because the network has disconnected the access connection to end connection (if any).

2.2.1.11 release request (U19): This state exists when the user has requested the network to release the access connection and is waiting for a response.

2.2.2 Access connection states at the network side of the interface

The states which may exist on the network side of the user-network interface are defined in this subclause.

2.2.2.1 null state (N0): No access connection exists.

2.2.2.2 call initiated (N1): This state exists for an outgoing access connection when the network has received an access connection establishment request but has not yet responded.

2.2.2.3 outgoing call proceeding (N3): This state exists for an outgoing access connection when the network has sent acknowledgement that the network has received all access connection information necessary to effect access connection establishment.

2.2.2.4 call present (N6): This state exists for an incoming access connection when the network has sent an access connection establishment request but has not yet received a satisfactory response.

2.2.2.5 call received (N7): This state exists for an incoming access connection when the network has received an indication that the user is alerting but has not yet received an answer.

2.2.2.6 connect request (N8): This state exists for an incoming access connection when the network has received an answer but the network has not yet awarded the access connection.

2.2.2.7 incoming call proceeding (N9): This state exists for an incoming access connection when the network has received acknowledgment that the user has received all access connection information necessary to effect access connection establishment.

2.2.2.8 active (N10): This state exists for an incoming access connection when the network has awarded the access connection to the called user. This state exists for an outgoing access connection when the local network has indicated that the access connection has been completed.

2.2.2.9 disconnect request (N11): This state exists when the network has received a request from the user to clear the access connection.

2.2.2.10 disconnect indication (N12): This state exists when the network has sent an invitation to disconnect the user-network access connection.

2.2.2.11 release request (N19): This state exists when the network has requested the user to release the access connection and is waiting for a response.

2.2.2.12 call abort (N22): This state exists for an incoming access connection for the point-to-multipoint configuration when the access connection is being cleared before any user has been awarded the access connection.

2.3 Temporary signalling connections

This subclause defines the basic call control states for user-to-user signalling not associated with circuit switched calls. The procedures for call control are given in 7.2.

2.3.1 Call states at the user side of the interface

The states which may exist on the user side of the user-network interface are defined in this subclause.

2.3.1.1 null state (U0): No call exists.

2.3.1.2 call initiated (U1): This state exists for an outgoing call when the user requests call establishment from the network.

2.3.1.3 overlap sending (U2): This state exists for an outgoing call when the user has received acknowledgement of the call establishment request which permits the user to send additional call information to the network in overlap mode.

2.3.1.4 outgoing call proceeding (U3): This state exists for an outgoing call when the user has received acknowledgement that the network has received all call information necessary to effect call establishment.

2.3.1.5 call delivered (U4): This state exists for an outgoing call when the calling user has received an indication that remote user alerting has been initiated.

2.3.1.6 call present (U6): This state exists for an incoming call when the user has received a call establishment request but has not yet responded.

2.3.1.7 call received (U7): This state exists for an incoming call when the user has indicated alerting but has not yet answered.

2.3.1.8 connect request (U8): This state exists for an incoming call when the user has answered the call and is awaiting to be awarded the call.

2.3.1.9 incoming call proceeding (U9): This state exists for an incoming call when the user has sent acknowledgement that the user has received all call information necessary to effect call establishment.

2.3.1.10 active (U10): This state exists for an incoming call when the user has received an acknowledgement from the network that the user has been awarded the call. This state exists for an outgoing call when the user has received an indication that the remote user has answered the call.

2.3.1.11 release request (U19): This state exists when the user has requested the network to release and is waiting for a response.

2.3.1.12 overlap receiving (U25): This state exists for an incoming call when the user has acknowledged the call establishment request from the network and is prepared to receive additional call information (if any) in overlap mode.

2.3.2 Network call states

The call states that may exist on the network side of the user-network interface are defined in this subclause.

2.3.2.1 null state (N0): No call exists.

2.3.2.2 call initiated (N1): This state exists for an outgoing call when the network has received a call establishment request but has not yet responded.

2.3.2.3 overlap sending (N2): This state exists for an outgoing call when the network has acknowledged the call establishment request and is prepared to receive additional call information (if any) in overlap mode.

2.3.2.4 outgoing call proceeding (N3): This state exists for an outgoing call when the network has sent acknowledgement that the network has received all call information necessary to effect call establishment.

2.3.2.5 call delivered (N4): This state exists for an outgoing call when the network has indicated that remote user alerting has been initiated.

2.3.2.6 call present (N6): This state exists for an incoming call when the network has sent a call establishment request but has not yet received a satisfactory response.

2.3.2.7 call received (N7): This state exists for an incoming call when the network has received an indication that the user is alerting but has not yet received an answer.

2.3.2.8 connect request (N8): This state exists for an incoming call when the network has received an answer but the network has not yet awarded the call.

2.3.2.9 incoming call proceeding (N9): This state exists for an incoming call when the network has received acknowledgement that the user has received all call information necessary to effect call establishment.

2.3.2.10 active (N10): This state exists for an incoming call when the network has awarded the call to the called user. This state exists for an outgoing call when the network has indicated that the remote user has answered the call.

2.3.2.11 release request (N19): This state exists when the network has requested the user to release and is waiting for a response.

2.3.2.12 call abort (N22): This state exists for an incoming call for the point-to-multipoint configuration when the call is being cleared before any user has been awarded the call.

2.3.2.13 overlap receiving (N25): This state exists for an incoming call when the network has received acknowledgement of the call establishment request which permits the network to send additional call information (if any) in the overlap mode.

2.4 States associated with the global call reference

This subclause defines the states that the protocol may adopt using the global call reference. The procedures for use of the global call reference for RESTART are contained in 5.5.

There is only one global call reference per interface.

2.4.1 Call states at the user side of the interface

The states which may exist on the user side of the user network interface are defined in this subclause.

2.4.1.1 null (Rest 0): No transaction exists.

2.4.1.2 restart request (Rest 1): This state exists for a restart transaction when the user has sent a restart request but has not yet received an acknowledgement response from the network.

2.4.1.3 restart (Rest 2): This state exists when a request for a restart has been received from the network and responses have not yet been received from all locally active call references.

2.4.2 Call states at the network side of the interface

The states which may exist on the network side of the user-network interface are defined in this subclause.

2.4.2.1 null (Rest 0): No transaction exists.

2.4.2.2 restart request (Rest 1): This state exists for a restart transaction when the network has sent a restart request but has not yet received an acknowledgement response from the user.

2.4.2.3 restart (Rest 2): This state exists when a request for a restart has been received from the user and a response has not yet been received from all locally active call references.

3 Message functional definitions and content

This subclause provides an overview of the Q.931 message structure, which highlights the functional definition and information content (i.e. semantics) of each message. Each definition includes:

a) A brief description of the message direction and use, including whether the message has:
   1) local significance, i.e. relevant only in the originating or terminating access;
   2) access significance, i.e. relevant in the originating and terminating access, but not in the network;
   3) dual significance, i.e. relevant in either the originating or terminating access and in the network; or
   4) global significance, i.e. relevant in the originating and terminating access and in the network.

b) A table listing the codeset 0 information elements in the order of their appearance in the message (same relative order for all message types). For each information element, the table indicates:
   1) the clause of this Recommendation describing the information element;
   2) the direction in which it may be sent; i.e. user to network ("u → n"), network to user ("n → u"), or both;

      NOTE 1 – The user-network terminology in [3] refers to the TE-ET, TE-NT2, and NT2-ET interface structures. Annex D contains a description of the information element usage for symmetric NT2-NT2 interfaces.
   3) whether inclusion is mandatory ("M") or optional ("O"), with a reference to Notes explaining the circumstances under which the information element shall be included;
   4) the length of the information element (or permissible range of lengths), in octets, where "*" denotes an undefined maximum length, which may be network or service dependent;

      NOTE 2 – All messages may contain information elements from codesets 5, 6 and 7 and corresponding locking and non-locking shift information elements which comply with the coding rules specified in 4.5.2-4.5.4. None of these information elements, however, are listed in any of the tables in clause 3.

c) further explanatory Notes, as necessary.

3.1 Messages for circuit-mode connection control

Table 3-1 summarizes the messages for circuit-mode connection control.

Table 3-1/Q.931 – Messages for circuit-mode connection control

	Reference (subclauses)
Call establishment messages:	
ALERTING	3.1.1
CALL PROCEEDING	3.1.2
CONNECT	3.1.3
CONNECT ACKNOWLEDGE	3.1.4
PROGRESS	3.1.8
SETUP	3.1.14
SETUP ACKNOWLEDGE	3.1.15
Call information phase messages:	
RESUME	3.1.11
RESUME ACKNOWLEDGE	3.1.12
RESUME REJECT	3.1.13
SUSPEND	3.1.18
SUSPEND ACKNOWLEDGE	3.1.19
SUSPEND REJECT	3.1.20
Call clearing messages:	
DISCONNECT	3.1.5
RELEASE	3.1.9
RELEASE COMPLETE	3.1.10
Miscellaneous messages:	
INFORMATION	3.1.6
NOTIFY	3.1.7
SEGMENT	Annex H (Note 2)
STATUS	3.1.16
STATUS ENQUIRY	3.1.17
NOTE 1 – In Recommendation Q.931 (1988) [53], support of user-user signalling was included for a number of reasons, including support of additional compatibility checking upon bilateral agreement with other users or in accordance with other standards (e.g. Recommendation X.213 [23]). To utilize this capability, the User-user information element can be included in the ALERTING, CONNECT, DISCONNECT, PROGRESS, RELEASE, RELEASE COMPLETE and SETUP messages. Details on this capability (explicit and implicit Type 1 user-user signalling) are given in Recommendation Q.957 [54]. NOTE 2 – The segment message is required if the optional segmentation procedure defined in Annex H is implemented.	

3.1.1 ALERTING

This message is sent by the called user to the network and by the network to the calling user, to indicate that called user alerting has been initiated. See Table 3-2.

Table 3-2/Q.931 – ALERTING message content

Message type: ALERTING
Significance: Global
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	Both	O (Note 1)	4-12
Channel identification	4.5	Both (Note 2)	O (Note 3)	2-*
Progress indicator	4.5	Both	O (Note 4)	2-4
Display	4.5	n → u	O (Note 5)	(Note 6)
Signal	4.5	n → u	O (Note 7)	2-3
High layer compatibility	4.5	Both	O (Note 8)	2-5

NOTE 1 – The Bearer capability information element is included when the procedures of 5.11 for bearer capability selection apply. When present, progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change,* shall also be present.

NOTE 2 – Included in the network-to-user direction for support of the procedures in Annex D.

NOTE 3 – Mandatory if this message is the first message in response to a SETUP, unless the user accepts the B-channel indicated in the SETUP message.

NOTE 4 – Included in the event of interworking. Included in the network-to-user direction in connection with the provision of in-band information/patterns. Included in the user-to-network direction in connection with the provision of in-band information/patterns if Annex K is implemented or in accordance with the procedures of 5.11.3 and 5.12.3.

NOTE 5 – Included if the network provides information that can be presented to the user.

NOTE 6 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 7 – Included if the network optionally provides information describing tones or alerting signals.

NOTE 8 – The High layer compatibility information element is included when the procedures of 5.12 for high layer compatibility selection apply. When present, progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change,* shall also be present.

3.1.2 CALL PROCEEDING

This message is sent by the called user to the network or by the network to the calling user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted. See Table 3-3.

Table 3-3/Q.931 – CALL PROCEEDING message content

Message type: CALL PROCEEDING
Significance: Local
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	Both	O (Note 5)	4-12
Channel identification	4.5	Both	O (Note 1)	2-*
Progress indicator	4.5	Both	O (Note 2)	2-4
Display	4.5	n → u	O (Note 3)	(Note 4)
High layer compatibility	4.5	Both	O (Note 6)	2-5

NOTE 1 – Mandatory in the network-to-user direction if this message is the first message in response to a SETUP message. It is mandatory in the user-to-network direction if this message is the first message in response to a SETUP message, unless the user accepts the B-channel indicated in the SETUP message.

NOTE 2 – Included in the event of interworking. Included in the network-to-user direction in connection with the provision of in-band information/patterns. Included in the user-to-network direction in connection with the provision of in-band information/patterns if Annex K is implemented or in accordance with the procedures of 5.11.3 and 5.12.3.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – The Bearer capability information element is included when the procedures of 5.11 for bearer capability selection apply. When present, progress indication No. 5, *interworking has occurred and has resulted in a telecommunication service change,* shall also be present.

NOTE 6 – The High layer compatibility information element is included when the procedures of 5.12 for high layer compatibility selection apply. When present, progress indication No. 5, *interworking has occurred and has resulted in a telecommunication service change,* shall also be present.

3.1.3 CONNECT

This message is sent by the called user to the network and by the network to the calling user, to indicate call acceptance by the called user. See Table 3-4.

Table 3-4/Q.931 – CONNECT message content

Message type: CONNECT
Significance: Global
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	Both	O (Note 1)	4-12
Channel identification	4.5	Both (Note 2)	O (Note 3)	2-*
Progress indicator	4.5	Both	O (Note 4)	2-4
Display	4.5	n → u	O (Note 5)	(Note 6)
Date/time	4.5	n → u	O (Note 7)	8
Signal	4.5	n → u	O (Note 8)	2-3
Low layer compatibility	4.5	Both	O (Note 9)	2-18
High layer compatibility	4.5	Both	O (Note 10)	2-5

NOTE 1 – The Bearer capability information element is included when the procedures of 5.11 for bearer capability selection apply.

NOTE 2 – Included in the network-to-user direction for support of the procedures in Annex D.

NOTE 3 – Mandatory if this is the first message in response to a SETUP, unless the user accepts the B-channel indicated in the SETUP message.

NOTE 4 – Included in the event of interworking or in connection with the provision of in-band information/patterns.

NOTE 5 – Included if the network provides information that can be presented to the user.

NOTE 6 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 7 – As a network option, it may be included to provide date and time information to the calling user for all calls or for calls involving specific telecommunication services.

NOTE 8 – Included if the network optionally provides additional information describing tones.

NOTE 9 – Included in the user-to-network when the answering user wants to return low layer compatibility information to the calling user. Included in the network-to-user direction if the user awarded the call included a Low layer compatibility information element in the CONNECT message. Optionally included for low layer compatibility negotiation, but some networks may not transport this information element to the calling user (see Annex J).

NOTE 10 – The High layer compatibility information element is included when the procedures of 5.12 for high layer compatibility selection apply.

3.1.4 CONNECT ACKNOWLEDGE

This message is sent by the network to the called user to indicate the user has been awarded the call. It may also be sent by the calling user to the network to allow symmetrical call control procedures. See Table 3-5.

Table 3-5/Q.931 – CONNECT ACKNOWLEDGE message content

Message type: CONNECT ACKNOWLEDGE Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
Signal	4.5	n → u	O (Note 3)	2-3
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets. NOTE 3 – Included if the network optionally provides additional information describing tones.				

3.1.5 DISCONNECT

This message is sent by the user to request the network to clear an end-to-end connection or is sent by the network to indicate that the end-to-end connection is cleared. See Table 3-6.

Table 3-6/Q.931 – DISCONNECT message content

Message type: DISCONNECT Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Progress indicator	4.5	(Note 1)	O (Note 2)	2-4
Display	4.5	n → u	O (Note 3)	(Note 4)
Signal	4.5	n → u	O (Note 5)	2-3

NOTE 1 – Included in the network-to-user direction if the network provides in-band tones. See Annex D for usage in the user-to-network direction.

NOTE 2 – Included by the network if in-band tones are provided. However, the user may include the progress indicator and provide in-band tones. See Annex D. In such cases, the network will ignore this information element and will not convey the in-band tones.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – Included if the network optionally provides additional information describing tones.

3.1.6 INFORMATION

This message is sent by the user or the network to provide additional information. It may be used to provide information for call establishment (e.g. overlap sending) or miscellaneous call-related information. See Table 3-7.

Table 3-7/Q.931 – INFORMATION message content

Message type: INFORMATION
Significance: Local (Note 1)
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M (Note 2)	2-*
Message type	4.4	Both	M	1
Sending complete	4.5	Both	O (Note 3)	1
Display	4.5	n → u	O (Note 4)	(Note 5)
Keypad facility	4.5	u → n	O (Note 6)	2-34
Signal	4.5	n → u	O (Note 7)	2-3
Called party number	4.5	both	O (Note 8)	2-*

NOTE 1 – This message has local significance, but may carry information of global significance.

NOTE 2 – This message may be sent with the dummy call reference defined in 4.3 when feature key management procedures are used (see Recommendation Q.932); otherwise the minimum length is 2 octets.

NOTE 3 – Included if the user optionally indicates completion of overlap sending to the network, or if the network optionally indicates completion of overlap receiving to the user.

NOTE 4 – Included if the network provides information that can be presented to the user.

NOTE 5 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 6 – Either the Called party number or the Keypad facility information element is included by the user to transfer called party number information to the network during overlap sending. The Keypad facility information element may also be included if the user wants to convey other call establishment information to the network, or to convey supplementary service information.

NOTE 7 – Included if the network optionally provides additional information describing tones.

NOTE 8 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network during overlap sending. The Called party number information element is included by the network to convey called party number information to the user during overlap receiving.

3.1.7 NOTIFY

This message is sent by the user or the network to indicate information pertaining to a call, such as user suspended. See Table 3-8.

Table 3-8/Q.931 – NOTIFY message content

Message type: NOTIFY Significance: Access Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	n → u	O (Note 1)	2-12
Notification indicator	4.5	Both	M	3
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Included by the network to indicate a change of the bearer capability (see Annex L). NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.8 PROGRESS

This message is sent by the user or the network to indicate the progress of a call in the event of interworking or in relation with the provision of in-band information/patterns. See Table 3-9.

Table 3-9/Q.931 – PROGRESS message content

Message type: PROGRESS
Significance: Global
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	Both	O (Note 1)	4-12
Cause	4.5	Both	O (Note 2)	2-32
Progress indicator	4.5	Both	M	4
Display	4.5	n → u	O (Note 3)	(Note 4)
High layer compatibility	4.5	Both	O (Note 5)	2-5

NOTE 1 – The Bearer capability information element is included when the procedures of 5.11 for bearer capability selection apply. The Bearer capability information element indicates the bearer service now being used for the call.

NOTE 2 – Included by the user or the network to provide additional information concerning the provision of in-band information/patterns.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – The High layer compatibility information element is included when the optional procedures of 5.12 for high layer compatibility selection apply. The High layer compatibility information element indicates the high layer compatibility now being used for the call.

3.1.9 RELEASE

This message is sent by the user or the network to indicate that the equipment sending the message has disconnected the channel (if any) and intends to release the channel and the call reference. Thus the receiving equipment should release the channel and prepare to release the call reference after sending a RELEASE COMPLETE. See Table 3-10.

Table 3-10/Q.931 – RELEASE message content

Message type: RELEASE
Significance: Local (Note 1)
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)
Signal	4.5	n → u	O (Note 5)	2-3

NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message.

NOTE 2 – Mandatory in the first call clearing message, including when the RELEASE message is sent as a result of an error handling condition.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – Included if the network optionally provides additional information describing tones.

3.1.10 RELEASE COMPLETE

This message is sent by the user or the network to indicate that the equipment sending the message has released the channel (if any) and call reference, the channel is available for reuse, and the receiving equipment shall release the call reference. See Table 3-11.

Table 3-11/Q.931 – RELEASE COMPLETE message content

Message type: RELEASE COMPLETE Significance: Local (Note 1) Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)
Signal	4.5	n → u	O (Note 5)	2-3
NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message. NOTE 2 – Mandatory in the first call clearing message, including when the RELEASE COMPLETE message is sent as a result of an error handling condition. NOTE 3 – Included if the network provides information that can be presented to the user. NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets. NOTE 5 – Included if the network optionally provides additional information describing tones.				

3.1.11 RESUME

This message is sent by the user to request the network to resume a suspended call. See Table 3-12.

Table 3-12/Q.931 – RESUME message content

Message type: RESUME Significance: Local Direction: User-to-network				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	u → n	M	1
Call reference	4.3	u → n	M	2-*
Message type	4.4	u → n	M	1
Call identity	4.5	u → n	O (Note)	2-10
NOTE – Included when the SUSPEND message used to suspend the call included a Call identity information element.				

3.1.12 RESUME ACKNOWLEDGE

This message is sent by the network to the user to indicate completion of a request to resume a suspended call. See Table 3-13.

Table 3-13/Q.931 – RESUME ACKNOWLEDGE message content

Message type: RESUME ACKNOWLEDGE Significance: Local Direction: Network-to-user				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	n → u	M	1
Call reference	4.3	n → u	M	2-*
Message type	4.4	n → u	M	1
Channel identification	4.5	n → u	M	3-*
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.13 RESUME REJECT

This message is sent by the network to the user to indicate failure of a request to resume a suspended call. See Table 3-14.

Table 3-14/Q.931 – RESUME REJECT message content

Message type: RESUME REJECT Significance: Local Direction: Network-to-user				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	n → u	M	1
Call reference	4.3	n → u	M	2-*
Message type	4.4	n → u	M	1
Cause	4.5	n → u	M	4-32
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.14 SETUP

This message is sent by the calling user to the network and by the network to the called user to initiate call establishment. See Table 3-15.

Table 3-15/Q.931 – SETUP message content

Message type: SETUP Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Sending complete	4.5	Both	O (Note 1)	1
Repeat indicator	4.5	Both	O (Note 2)	1
Bearer capability	4.5	Both	M (Note 3)	4-12
Channel identification	4.5	Both	O (Note 4)	2-*
Progress indicator	4.5	Both	O (Note 5)	2-4
Network-specific facilities	4.5	Both	O (Note 6)	2-*
Display	4.5	n → u	O (Note 7)	(Note 8)
Date/Time	4.5	u → n	O (Note 19)	8
Keypad facility	4.5	u → n	O (Note 9)	2-34
Signal	4.5	n → u	O (Note 10)	2-3
Calling party number	4.5	Both	O (Note 11)	2-*
Calling party subaddress	4.5	Both	O (Note 12)	2-23
Called party number	4.5	Both	O (Note 13)	2-*
Called party subaddress	4.5	Both	O (Note 14)	2-23
Transit network selection	4.5	u → n	O (Note 15)	2-*
Repeat indicator	4.5	Both	O (Note 16)	1
Low layer compatibility	4.5	Both	O (Note 17)	2-18
High layer compatibility	4.5	Both	O (Note 18)	2-5

NOTE 1 – Included if the user or the network optionally indicates that all information necessary for call establishment is included in the SETUP message.

NOTE 2 – The Repeat indicator information element is included immediately before the first Bearer capability information element when the bearer capability negotiation procedure is used (see Annex L).

NOTE 3 – May be repeated if the bearer capability negotiation procedure is used (see Annex L). For bearer capability negotiation, two Bearer capability information elements may be included in descending order of priority, i.e. highest priority first. Although support of multiple Bearer capability information elements may not be supported on all networks, on networks that do support it, and through suitable subscription arrangements, three Bearer capability information elements may be included (see 5.11). When they are not preceded by a Repeat indicator information element, they are included in ascending order of priority.

NOTE 4 – Mandatory in the network-to-user direction. Included in the user-to-network direction when a user wants to indicate a channel. If not included, its absence is interpreted as "any channel acceptable".

Table 3-15/Q.931 – SETUP message content *(concluded)*

NOTE 5 – Included in the event of interworking or in connection with the provision of in-band information/patterns.

NOTE 6 – Included by the calling user or the network to indicate network-specific facilities information (see Annex E).

NOTE 7 – Included if the network provides information that can be presented to the user.

NOTE 8 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 9 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network. The Keypad facility information element may also be included by the user to convey other call establishment information to the network.

NOTE 10 – Included if the network optionally provides additional information describing tones.

NOTE 11 – May be included by the calling user or the network to identify the calling user. Not included in the network-to-user direction for basic call control, but may be included for some supplementary services.

NOTE 12 – Included in the user-to-network direction when the calling user wants to indicate the calling party subaddress. Not included in the network-to-user direction for basic call control, but may be included for some supplementary services.

NOTE 13 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network. The Called party number information element is included by the network when called party number information is to be conveyed to the user.

NOTE 14 – Included in the user-to-network direction when the calling user wants to indicate the called party subaddress. Included in the network-to-user direction if the calling user included a Called party subaddress information element in the SETUP message.

NOTE 15 – Included by the calling user to select a particular transit network (see Annex C).

NOTE 16 – Included when two or more Low layer compatibility information elements are included for low layer compatibility negotiation.

NOTE 17 – Included in the user-to-network direction when the calling user wants to pass low layer compatibility information to the called user. Included in the network-to-user direction if the calling user included a Low layer compatibility information element in the SETUP message. Two, three or four information elements may be included in descending order of priority, i.e. highest priority first, if the low layer compatibility negotiation procedures are used (see Annex J).

NOTE 18 – Included in the user-to-network direction when the calling user wants to pass high layer compatibility information to the called user. Included in the network-to-user direction if the calling user included a High layer compatibility information element in the SETUP message. Although support of multiple High layer compatibility information elements may not be supported on all networks, on networks that do support it, and through suitable subscription arrangements, two High layer compatibility information elements may be included (see 5.12). When they are not preceded by a Repeat indicator information element, they are included in ascending order of priority.

NOTE 19 – As a network option, it may be included to provide date and time information to the called user.

3.1.15 SETUP ACKNOWLEDGE

This message is sent by the network to the calling user, or by the called user to the network, to indicate that call establishment has been initiated, but additional information may be required. See Table 3-16.

Table 3-16/Q.931 – SETUP ACKNOWLEDGE message content

Message type: SETUP ACKNOWLEDGE
Significance: Local
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 1)	2-*
Progress indicator	4.5	Both	O (Note 2)	2-4
Display	4.5	n → u	O (Note 3)	(Note 4)
Signal	4.5	n → u	O (Note 5)	2-3

NOTE 1 – Mandatory in all cases, except when the user accepts the specific B-channel indicated in the SETUP message.

NOTE 2 – Included in the event of interworking or in connection with the provision of in-band information/patterns.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – Included if the network optionally provides information describing tones (e.g. activate dial tone).

3.1.16 STATUS

This message sent is by the user or the network in response to a STATUS ENQUIRY message or at any time during a call to report certain error conditions listed in 5.8. See Table 3-17.

Table 3-17/Q.931 – STATUS message content

Message type: STATUS Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Call state	4.5	Both	M	3
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.17 STATUS ENQUIRY

The STATUS ENQUIRY message is sent by the user or the network at any time to solicit a STATUS message from the peer layer 3 entity. Sending a STATUS message in response to a STATUS ENQUIRY message is mandatory. See Table 3-18.

Table 3-18/Q.931 – STATUS ENQUIRY message content

Message type: STATUS ENQUIRY Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.18 SUSPEND

This message is sent by the user to request the network to suspend a call. See Table 3-19.

Table 3-19/Q.931 – SUSPEND message content

Message type: SUSPEND Significance: Local Direction: User-to-network				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	u → n	M	1
Call reference	4.3	u → n	M	2-*
Message type	4.4	u → n	M	1
Call identity	4.5	u → n	O (Note)	2-10
NOTE – Included if the user wants to identify the suspended call explicitly.				

3.1.19 SUSPEND ACKNOWLEDGE

This message is sent by the network to the user to indicate completion of a request to suspend a call. See Table 3-20.

Table 3-20/Q.931 – SUSPEND ACKNOWLEDGE message content

Message type: SUSPEND ACKNOWLEDGE Significance: Local Direction: Network-to-user				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	n → u	M	1
Call reference	4.3	n → u	M	2*
Message type	4.4	n → u	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.1.20 SUSPEND REJECT

This message is sent by the network to the user to indicate failure of a request to suspend a call. See Table 3-21.

Table 3-21/Q.931 – SUSPEND REJECT message content

Message type: SUSPEND REJECT Significance: Local Direction: Network-to-user				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	n → u	M	1
Call reference	4.3	n → u	M	2-*
Message type	4.4	n → u	M	1
Cause	4.5	n → u	M	4-32
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.2 Messages for packet-mode connection control

Table 3-22 summarizes the messages for packet-mode access connection control. The message tables in this subclause should be used for Case B (packet-switched access to an ISDN virtual circuit service) as defined in clause 6. For Case A (circuit-switched access to PSPDN services), the message tables in 3.1 should be used.

Table 3-22/Q.931 – Messages for packet-mode access connection control

	Reference (subclause)
Access connection establishment messages:	
ALERTING	3.2.1
CALL PROCEEDING	3.2.2
CONNECT	3.2.3
CONNECT ACKNOWLEDGE	3.2.4
PROGRESS	3.2.6
SETUP	3.2.9
Access connection clearing messages:	
DISCONNECT	3.2.5
RELEASE	3.2.7
RELEASE COMPLETE	3.2.8
Miscellaneous messages:	
STATUS	3.2.10
STATUS ENQUIRY	3.2.11

3.2.1 ALERTING

This message is sent by the called user to the network to indicate that called user alerting has been initiated. See Table 3-23.

Table 3-23/Q.931 – ALERTING message content

Message type: ALERTING Significance: Local Direction: User-to-network				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	u → n	M	1
Call reference	4.3	u →n	M	2-*
Message type	4.4	u → n	M	1
Channel identification	4.5	u → n	O (Note 1)	2-*
Progress indicator	4.5	u → n	O (Note 2)	2-4
NOTE 1 – Mandatory if this message is the first message in response to SETUP, unless the user accepts the channel indicated in the SETUP message. NOTE 2 – Included in the event of interworking within a private network.				

3.2.2 CALL PROCEEDING

This message is sent by the called user or by the network to the calling user to indicate that requested access connection establishment has been initiated. See Table 3-24.

Table 3-24/Q.931 – CALL PROCEEDING message content

Message type: CALL PROCEEDING
Significance: Local
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 1)	2-*
Progress indicator	4.5	u → n	O (Note 2)	2-4
Display	4.5	n → u	O (Note 3)	(Note 4)

NOTE 1 – Mandatory in the network-to-user direction if this message is the first message in response to a SETUP. Mandatory in the user-to-network direction if this message is the first message in response to SETUP message, unless the user accepts the channel indicated in the SETUP message.

NOTE 2 – Included in the event of interworking. Included in the network-to-user direction in connection with the provision of in-band information/patterns. Included in the user-to-network direction in connection with in-band information/patterns if Annex K is implemented or in accordance with the procedures of 5.11.3 and 5.12.3.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

3.2.3 CONNECT

This message is sent by the called user to the network, and by the network to the calling user, to indicate acceptance of the access connection. See Table 3-25.

Table 3-25/Q.931 – CONNECT message content

Message type: CONNECT Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	u → n	O (Note 1)	2-*
Progress indicator	4.5	u → n	O (Note 4)	2-4
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Mandatory if this message is the first message in response to SETUP, unless the user accepts the channel indicated in the SETUP message. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets. NOTE 4 – Included in the event of interworking within a private network.				

3.2.4 CONNECT ACKNOWLEDGE

This message is sent by the network to the called user to indicate that the user has been awarded the access connection. It may also be sent by the calling user to the network to allow symmetrical access connection control procedures. See Table 3-26.

Table 3-26/Q.931 – CONNECT ACKNOWLEDGE message content

Message type: CONNECT ACKNOWLEDGE Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.2.5 DISCONNECT

This message is sent by the user to request the network to clear an access connection or is sent by the network to the user to indicate clearing of the access connection. See Table 3-27.

Table 3-27/Q.931 – DISCONNECT message contents

Message type: DISCONNECT Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Display	4.5	n → u	O (Note 1)	(Note 2)
User-user	4.5	u → n	O (Note 3)	(Note 4)

NOTE 1 – Included if the network provides information that can be presented to the user.

NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 3 – May be sent if the access connection has not yet reached the active state. However, user-user information is not sent after the access connection has reached the active state since X.25 procedures would be used for this information transfer.

NOTE 4 – The minimum length is 2 octets; the standard default maximum length is 131 octets.

3.2.6 PROGRESS

This message is sent by the called user or the network to indicate the progress of an access connection establishment in the event of interworking within a private network. See Table 3-28.

Table 3-28/Q.931 – PROGRESS message content

Message type: PROGRESS Significance: Local Direction: User-to-network				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	u → n	M	1
Call reference	4.3	u → n	M	2-*
Message type	4.4	u → n	M	1
Cause	4.5	u → n	O (Note)	2-32
Progress indicator	4.5	u → n	M	4

NOTE – Included by the called user to provide additional information.

3.2.7 RELEASE

This message is sent by the user or the network to indicate that the equipment sending the message has disconnected the channel (if any) and intends to release the channel and the call reference, and that the receiving equipment should release the channel and prepare to release the call reference after sending RELEASE COMPLETE. The RELEASE message is sent by the network to the user to indicate that the access connection is awarded on either the D-channel or an existing channel and that the network intends to release the call reference. See Table 3-29.

Table 3-29/Q.931 – RELEASE message content

Message type: RELEASE
Significance: Local (Note 1)
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)
User-user	4.5	u → n	O (Note 5)	(Note 6)

NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message.

NOTE 2 – Mandatory in the first clearing message, including when the RELEASE message is sent as a result of an error handling condition.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – User-user information may be sent if RELEASE is the first clearing message and the access connection has not yet reached the active state and Q.931/X.25 mapping service is provided by the network. However, user-user information is not sent if the access connection has reached the active state since X.25 procedures would be used for this information transfer.

NOTE 6 – The minimum length is 2 octets; the standard default maximum length is 131 octets.

3.2.8 RELEASE COMPLETE

This message is sent by the user or the network to indicate that the equipment sending the message has released the channel (if any) and call reference. The channel is available for reuse, and the receiving equipment shall release the call reference. See Table 3-30.

Table 3-30/Q.931 – RELEASE COMPLETE message content

Message type: RELEASE COMPLETE
Significance: Local (Note 1)
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)
User-user	4.5	u → n	O (Note 5)	(Note 6)

NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message.

NOTE 2 – Mandatory in the first call clearing message, including when the RELEASE COMPLETE message is sent as a result of an error handling condition.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 5 – User-user information may be sent if RELEASE COMPLETE is the first clearing message and the access connection has not yet reached the active state and Q.931/X.25 mapping service is provided by the network. However, user-user information is not sent if the access connection has reached the active state since X.25 [5] procedures would be used for this information transfer.

NOTE 6 – The minimum length is 2 octets; the standard default maximum length is 131 octets.

3.2.9 SETUP

This message is sent by the calling user to the network and by the network to the called user to initiate access connection establishment. See Table 3-31.

Table 3-31/Q.931 – SETUP message content

Message type: SETUP Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Bearer capability	4.5	Both	M (Note 1)	4-12
Channel identification	4.5	Both	O (Note 2)	2-*
Progress indicator	4.5	u → n	O (Note 3)	2-4
Display	4.5	n → u	O (Note 4)	(Note 5)
Information rate	4.6	n → u	O (Note 6)	2-6
End-end transit delay	4.6	n → u	O (Note 8)	2-11
Transit delay selection and indication	4.6	n → u	O (Note 7)	2-5
Packet layer binary parameters	4.6	n → u	O (Note 9)	2-3
Packet layer window size	4.6	n → u	O (Note 10)	2-4
Packet size	4.6	n → u	O (Note 11)	2-4
Closed user group	4.6	n → u	O (Note 12)	4-7
Reverse charging indication	4.6	n → u	O (Note 13)	3
Calling party number	4.5	Both	O (Note 14)	2-*
Calling party subaddress	4.5	Both	O (Note 15)	2-23
Called party number	4.5	n → u	O (Note 16)	2-*
Called party subaddress	4.5	n → u	O (Note 17)	2-23
Redirecting number	4.6	n → u	O (Note 18)	2-*
User-user	4.5	n → u	O (Note 19)	(Note 20)

NOTE 1 – May be used to describe an ITU-T telecommunication service involving packet-mode access connections, if appropriate.

NOTE 2 – Mandatory in the network-to-user direction. Included in the user-to-network direction when the user wants to indicate a channel. If not included, its absence is interpreted as "any channel acceptable".

NOTE 3 – Included in the event of interworking within a private network.

NOTE 4 – Included if the network provides information that can be presented to the user.

NOTE 5 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

Table 3-31/Q.931 – SETUP message content *(concluded)*

NOTE 6 – Included in the network-to-user direction if the network implements X.25 [5]/Q.931 information element mapping and provides indication to the called user of the information rate for the call.

NOTE 7 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the maximum permissible transit delay for the call.

NOTE 8 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the end-end transit delay for the call.

NOTE 9 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the packet layer binary parameters for the call.

NOTE 10 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the packet layer window size for the call.

NOTE 11 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the packet size for the call.

NOTE 12 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called party of the closed user group that belongs for that call.

NOTE 13 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called party of the reverse charging request that applies for that call.

NOTE 14 – Included in the user-to-network direction depending on the user/network identification requirements. Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the calling party number.

NOTE 15 – Included in the user-to-network direction depending on the user/network identification requirements. Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the calling party subaddress.

NOTE 16 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the called party number.

NOTE 17 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the called party subaddress.

NOTE 18 – Included in the network-to-user direction if the network implements X.25/Q.931 information element mapping and provides indication to the called user of the number from which a call diversion or transfer was invoked.

NOTE 19 – Included in the network-to-user direction if the calling user included user information and the network implements X.25/Q.931 information element mapping.

NOTE 20 – The minimum length is 2 octets; the standard default maximum length is 131 octets.

3.2.10 STATUS

This message is sent by the user or the network in response to a STATUS ENQUIRY message or at any time to report certain error conditions listed in 5.8. See Table 3-32.

Table 3-32/Q.931 – STATUS message content

Message Type: STATUS Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Call state	4.5	Both	M	3
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.2.11 STATUS ENQUIRY

The STATUS ENQUIRY message is sent by the user or the network at any time to solicit a STATUS message from the peer layer 3 entity. Sending a STATUS message in response to a STATUS ENQUIRY message is mandatory. See Table 3-33.

Table 3-33/Q.931 – STATUS ENQUIRY message content

Message type: STATUS ENQUIRY Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3 Messages for user signalling bearer service control

Table 3-34 summarizes the messages for the control of non-call associated temporary signalling connections and the transfer of user-user information.

Table 3-34/Q.931 – Messages for temporary signalling connection control

	Reference (subclause)
Call establishment messages:	
ALERTING	3.3.1
CALL PROCEEDING	3.3.2
CONNECT	3.3.4
CONNECT ACKNOWLEDGE	3.3.5
SETUP	3.3.9
SETUP ACKNOWLEDGE	3.3.10
Call information phase messages:	
USER INFORMATION	3.3.13
Call clearing messages:	
RELEASE	3.3.7
RELEASE COMPLETE	3.3.8
Miscellaneous messages:	
CONGESTION CONTROL	3.3.3
INFORMATION	3.3.6
STATUS	3.3.11
STATUS ENQUIRY	3.3.12
NOTE – In Recommendation Q.931 (1988), support of user-user signalling was included for a number of reasons, including support of additional compatibility checking upon bilateral agreement with other users or in accordance with other standards (e.g. Recommendation X.213 [23]). To utilise this capability, the User-user information element can be included in the ALERTING, CONNECT, RELEASE, RELEASE COMPLETE and SETUP messages. Details on this capability (explicit and implicit Type 1 user-user signalling) are given in Recommendation Q.957 [54].	

3.3.1 ALERTING

This message is sent by the called user to the network, and by the network to the calling user, to indicate that called user alerting has been initiated. See Table 3-35.

Table 3-35/Q.931 – ALERTING message content

Message type: ALERTING Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	u → n	O (Note 1)	2-*
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Mandatory if this is the first message in response to SETUP, unless the user accepts the D-channel indicated in the SETUP message. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.2 CALL PROCEEDING

This message is sent by the called user to the network or by the network to the calling user to indicate that requested call establishment has been initiated and no more call establishment information will be accepted. See Table 3-36.

Table 3-36/Q.931 – CALL PROCEEDING message content

Message type: CALL PROCEEDING Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 1)	2-*
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Mandatory in the network-to-user direction if this message is the first message in response to a SETUP. Mandatory in the user-to-network direction if this message is the first message in response to a SETUP, unless the user accepts the D-channel indicated in the SETUP message. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.3 CONGESTION CONTROL

This message is sent by the network or the user to indicate the establishment or termination of flow control on the transmission of USER INFORMATION messages. See Table 3-37.

Table 3-37/Q.931 – CONGESTION CONTROL message content

Message type: CONGESTION CONTROL Significance: Local (Note 1) Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Congestion level	4.5	Both	M	1
Cause	4.5	Both	M	4-32
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – This message has local significance, but may carry information of global significance. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.4 CONNECT

This message is sent by the called user to the network, and by the network to the calling user, to indicate call acceptance by the called user. See Table 3-38.

Table 3-38/Q.931 – CONNECT message content

Message type: CONNECT Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	u → n	O (Note 1)	2-*
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Mandatory if this is the first message in response to a SETUP, unless the user accepts the D-channel indicated in the SETUP message. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 –The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.5 CONNECT ACKNOWLEDGE

This message is sent by the network to the called user to indicate that the user has been awarded the call. It may also be sent by the calling user to the network to allow symmetrical call control procedures. See Table 3-39.

Table 3-39/Q.931 – CONNECT ACKNOWLEDGE message content

Message type: CONNECT ACKNOWLEDGE Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.6 INFORMATION

This message is sent by the user or the network to provide additional information. It may be used to provide information for call establishment (e.g. overlap sending and receiving) or miscellaneous call-related information. See Table 3-40.

Table 3-40/Q.931 – INFORMATION message content

Message type: INFORMATION Significance: Local (Note 1) Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Sending complete	4.5	Both	O (Note 2)	1
Cause	4.5	n → u	O (Note 3)	2-32
Display	4.5	n → u	O (Note 4)	(Note 5)
Keypad facility	4.5	u → n	O (Note 6)	2-34
Called party number	4.5	Both	O (Note 7)	2-*

Table 3-40/Q.931 – INFORMATION message content *(concluded)*

NOTE 1 – This message has local significance, but may carry information of global significance.
NOTE 2 – Included when the user optionally indicates completion of overlap sending to the network, or if the network optionally indicates completion of overlap receiving to the user.
NOTE 3 – Included when the network optionally conveys additional information pertaining to user-user signalling (see clause 7).
NOTE 4 – Included if the network provides information that can be presented to the user.
NOTE 5 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.
NOTE 6 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network during overlap sending.
NOTE 7 – The Called party number information element is included by the network to convey called party number information to the user during overlap receiving.

3.3.7 RELEASE

This message is sent by the user or the network to indicate that the equipment sending the message has disconnected the channel (if any), and intends to release the channel and the call reference, and that the receiving equipment should release the channel and prepare to release the call reference after sending RELEASE COMPLETE. See Table 3-41.

Table 3-41/Q.931 – RELEASE message content

Message type: RELEASE
Significance: Local (Note 1)
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)

NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message.

NOTE 2 – Mandatory in the first call clearing message, including when the RELEASE message is sent as a result of an error handling condition.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

3.3.8 RELEASE COMPLETE

This message is sent by the user or the network to indicate that the equipment sending the message has released the channel (if any) and call reference. The channel is available for reuse, and the receiving equipment shall release the call reference. See Table 3-42.

Table 3-42/Q.931 – RELEASE COMPLETE message content

Message type: RELEASE COMPLETE Significance: Local (Note 1) Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	O (Note 2)	2-32
Display	4.5	n → u	O (Note 3)	(Note 4)
NOTE 1 – This message has local significance; however, it may carry information of global significance when used as the first call clearing message. NOTE 2 – Mandatory in the first call clearing message, including when the RELEASE message is sent as a result of an error handling condition. NOTE 3 – Included if the network provides information that can be presented to the user. NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.9 SETUP

This message is sent by the calling user to the network and by the network to the called user to initiate call establishment. See Table 3-43.

Table 3-43/Q.931 – SETUP message content

Message type: SETUP Significance: Global Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Sending complete	4.5	Both	O (Note 1)	1
Bearer capability	4.5	Both	M (Note 2)	6-8
Channel identification	4.5	Both	M	3-*
Network-specific facility	4.5	Both	O (Note 3)	2-*
Display	4.5	n → u	O (Note 4)	(Note 5)
Keypad facility	4.5	u → n	O (Note 6)	2-34
Calling party number	4.5	Both	O (Note 7)	2-*
Calling party subaddress	4.5	Both	O (Note 8)	2-23
Called party number	4.5	Both	O (Note 9)	2-*
Called party subaddress	4.5	Both	O (Note 10)	2-23

Table 3-43/Q.931 – SETUP message content *(concluded)*

Message type: SETUP
Significance: Global
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Transit network selection	4.5	u → n	O (Note 11)	2-*
Low layer compatibility	4.5	Both	O (Note 12)	2-18
High layer compatibility	4.5	Both	O (Note 13)	2-5

NOTE 1 – Included if the user or the network optionally indicates that all information necessary for call establishment is included in the SETUP message.

NOTE 2 – The Bearer capability and Low layer compatibility information elements may be used to describe an ITU-T telecommunication service, if appropriate.

NOTE 3 – Included by the calling user or the network to indicate network-specific facilities information (see Annex E).

NOTE 4 – Included if the network provides information that can be presented to the user.

NOTE 5 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

NOTE 6 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network during overlap sending. The Keypad facility information element may also be included by the user to convey other call establishment information to the network.

NOTE 7 – May be included by the calling user or the network to identify the calling user.

NOTE 8 – Included in the user-to-network direction when the calling user wants to indicate the calling party subaddress. Included in the network-to-user direction if the calling user included a Calling party subaddress information element in the SETUP message.

NOTE 9 – Either the Called party number or the Keypad facility information element is included by the user to convey called party number information to the network during overlap sending. The Called party number information element is included by the network when called party number information is conveyed to the user.

NOTE 10 – Included in the user-to-network direction when the calling user wants to indicate the called party subaddress. Included in the network-to-user direction if the calling user included a Called party subaddress information element in the SETUP message.

NOTE 11 – Included by the calling user to select a particular transit network (see Annex C).

NOTE 12 – Included in the user-to-network direction when the calling user wants to pass low layer compatibility information to the called user. Included in the network-to-user direction if the calling user included a Low layer compatibility information element in the SETUP message.

NOTE 13 – Included in the user-to-network direction when the calling user wants to pass high layer compatibility information to the called user. Included in the network-to-user direction if the calling user included a High layer compatibility information element in the SETUP message.

3.3.10 SETUP ACKNOWLEDGE

This message is sent by the network to the calling user, or by the called user to the network, to indicate that call establishment has been initiated but additional information may be required. See Table 3-44.

Table 3-44/Q.931 – SETUP ACKNOWLEDGE message content

Message type: SETUP ACKNOWLEDGE Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 1)	2-*
Display	4.5	n → u	O (Note 2)	(Note 3)
NOTE 1 – Mandatory in all cases, except when the user accepts the D-channel indicated in the SETUP message. NOTE 2 – Included if the network provides information that can be presented to the user. NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.11 STATUS

This message is sent by the user or the network in response to a STATUS ENQUIRY message or at any time during a call to report certain error conditions listed in 5.8. See Table 3-45.

Table 3-45/Q.931 – STATUS message content

Message type: STATUS Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Call state	4.5	Both	M	3
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.12 STATUS ENQUIRY

This message is sent by the user or the network at any time to solicit a STATUS message from the peer layer 3 entity. Sending a STATUS message in response to a STATUS ENQUIRY message is mandatory. See Table 3-46.

Table 3-46/Q.931 – STATUS ENQUIRY message content

Message type: STATUS ENQUIRY Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
Display	4.5	n → u	O (Note 1)	(Note 2)
NOTE 1 – Included if the network provides information that can be presented to the user. NOTE 2 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.				

3.3.13 USER INFORMATION

This message is sent by the user to the network to transfer information to the remote user. This message is also sent by the network to the user to deliver information from the other user. See Table 3-47.

Table 3-47/Q.931 – USER INFORMATION message content

Message type: USER INFORMATION Significance: Access Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M	2-*
Message type	4.4	Both	M	1
More data	4.5	Both	O (Note)	1
User-user	4.5	Both	M	2-255
NOTE – Included by the sending user to indicate that another USER INFORMATION message pertaining to the same message block will follow.				

3.4 Messages with the global call reference

Table 3-48 summarizes the messages which may use the global call reference defined in 4.3.

Table 3-48/Q.931 – Messages used with the global call reference

Messages	Reference (subclause)
RESTART	3.4.1
RESTART ACKNOWLEDGE	3.4.2
STATUS	3.4.3

3.4.1 RESTART

This message is sent by the user or network to request the recipient to restart (i.e. return to an idle condition) the indicated channel(s) or interface. See Table 3-49.

Table 3-49/Q.931 – RESTART message content

Message type: RESTART Significance: Local Direction: Both				
Information element	**Reference (subclause)**	**Direction**	**Type**	**Length**
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M (Note 1)	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 2)	2-*
Display	4.5	n → u	O (Note 3)	(Note 4)
Restart indicator	4.5	Both	M	3

NOTE 1 – This message is sent with the global call reference defined in 4.3.

NOTE 2 – Included when necessary to indicate the particular channel(s) to be restarted.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

3.4.2 RESTART ACKNOWLEDGE

This message is sent to acknowledge the receipt of the RESTART message and to indicate that the requested restart is complete. See Table 3-50.

Table 3-50/Q.931 – RESTART ACKNOWLEDGE message content

Message type: RESTART ACKNOWLEDGE
Significance: Local
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M (Note 1)	2-*
Message type	4.4	Both	M	1
Channel identification	4.5	Both	O (Note 2)	2-*
Display	4.5	n → u	O (Note 3)	(Note 4)
Restart indicator	4.5	Both	M	3

NOTE 1 – This message is sent with the global call reference defined in 4.3.

NOTE 2 – Included when necessary to indicate the particular channel(s) which have been restarted. May be repeated in the case of non-associated signalling that controls two or more interfaces.

NOTE 3 – Included if the network provides information that can be presented to the user.

NOTE 4 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

3.4.3 STATUS

This message is sent by the user or the network at any time during a call to report certain error conditions listed in 5.8. See Table 3-51.

Table 3-51/Q.931 – STATUS message content

Message type: STATUS
Significance: Local
Direction: Both

Information element	Reference (subclause)	Direction	Type	Length
Protocol discriminator	4.2	Both	M	1
Call reference	4.3	Both	M (Note 1)	2-*
Message type	4.4	Both	M	1
Cause	4.5	Both	M	4-32
Call state	4.5	Both	M	3
Display	4.5	n → u	O (Note 2)	(Note 3)

NOTE 1 – This message may be sent with the global call reference defined in 4.3.

NOTE 2 – Included if the network provides information that can be presented to the user.

NOTE 3 – The minimum length is 2 octets; the maximum length is network dependent and is either 34 or 82 octets.

4 General message format and information elements coding

The figures and text in this clause describe message contents. Within each octet, the bit designated "bit 1" is transmitted first, followed by bits 2, 3, 4, etc. Similarly, the octet shown at the top of each figure is sent first.

4.1 Overview

Within this protocol, every message shall consist of the following parts:

a) protocol discriminator;

b) call reference;

c) message type;

d) other information elements, as required.

Information elements a), b) and c) are common to all the messages and shall always be present, while information element d) is specific to each message type.

This organization is illustrated in the example shown in Figure 4-1.

A particular message may contain more information than a particular (user or network) equipment needs or can understand. All equipment should be able to ignore any extra information, present in a message, which is not required for the proper operation of that equipment. For example, a user may ignore the calling party number if that number is of no interest to the user when a SETUP message is received.

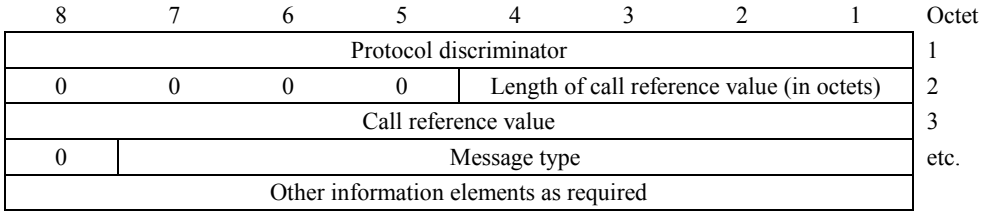


Figure 4-1/Q.931 – General message organization example

Unless specified otherwise, a particular information element may be present only once in a given message.

The term "default" implies that the value defined should be used in the absence of any assignment, or the negotiation of alternative values.

When a field, such as the call reference value, extends over more than one octet, the order of bit values progressively decreases as the octet number increases. The least significant bit of the field is represented by the lowest numbered bit of the highest-numbered octet of the field.

4.2 Protocol discriminator

The purpose of the protocol discriminator is to distinguish messages for user-network call control from other messages (to be defined). It also distinguishes messages of this Recommendation from those OSI network layer protocol units which are coded to other ITU-T Recommendations and other standards.

NOTE – A protocol discriminator field is also included in the User-user information element to indicate the user protocol within the user information; however, the coding of the protocol discriminator in this case is shown in 4.5.30.

The protocol discriminator is the first part of every message. The protocol discriminator is coded according to Table 4-1.

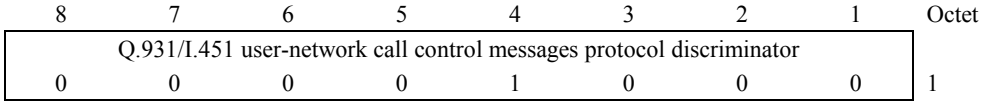


Figure 4-2/Q.931 – Protocol discriminator

Table 4-1/Q.931 – Protocol discriminator

Bits 8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0 through 0 0 0 0 0 1 1 1	Assigned in 4.5.30; not available for use in the message protocol discriminator
0 0 0 0 1 0 0 0	Recommendation Q.931/I.451 user-network call control messages
0 0 0 0 1 0 0 1	Recommendation Q.2931 user-network call control messages
0 0 0 1 0 0 0 0 through 0 0 1 1 1 1 1 1	Reserved for other network layer or layer 3 protocols, including Recommendation X.25 [5] (Note)
0 1 0 0 0 0 0 0 through 0 1 0 0 1 1 1 1	National use
0 1 0 1 0 0 0 0 through 1 1 1 1 1 1 1 0	Reserved for other network layer or layer 3 protocols, including Recommendation X.25 (Note)
All other values are reserved.	
NOTE – These values are reserved to discriminate these protocol discriminators from the first octet of an X.25 packet including general format identifier.	

4.3 Call reference

The purpose of the call reference is to identify the call or facility registration/cancellation request at the local user-network interface to which the particular message applies. The call reference does not have end-to-end significance across ISDNs.

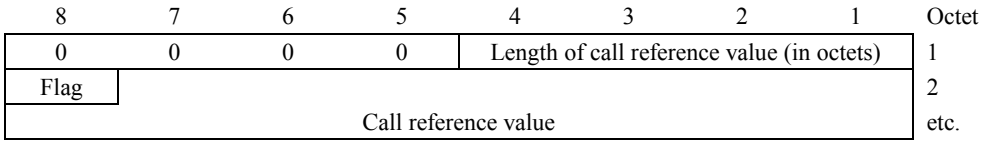
The call reference is the second part of every message. The call reference is coded as shown in Figure 4-3. The length of the call reference value is indicated in octet 1, bits 1-4. The default maximum length of the call reference information element is three octets long. The actions taken by the receiver are based on the numerical value of the call reference and are independent of the length of the call reference information element.

At a minimum, all networks and users must be able to support a call reference value of one octet for a basic user-network interface, and a call reference value of two octets for a primary rate interface.

As a network option for a primary rate interface, the call reference value may be one octet also. In this case, a call reference value up to 127 may be sent in one or two octets.

The call reference information element includes the call reference value and the call reference flag.

Call reference values are assigned by the originating side of the interface for a call. These values are unique to the originating side only within a particular D-channel layer two logical link connection. The call reference value is assigned at the beginning of a call and remains fixed for the lifetime of a call (except in the case of call suspension). After a call ends, or, after a successful suspension, the associated call reference value may be reassigned to a later call. Two identical call reference values on the same D-channel layer two logical link connection may be used when each value pertains to a call originated at opposite ends of the link.



NOTE – For call reference flag (octet 2)

Bit

8/0 The message is sent *from* the side that originates the call reference

1 The message is sent *to* the side that originates the call reference

Figure 4-3/Q.931 – Call reference information element

The call reference flag can take the values "0" or "1". The call reference flag is used to identify which end of the layer two-logical link originated a call reference. The origination side always sets the call reference flag to "0". The destination side always sets the call reference flag to a "1".

Hence the call reference flag identifies who allocated the call reference value for this call, and the only purpose of the call reference flag is to resolve simultaneous attempts to allocate the same call reference value.

The call reference flag also applies to functions which use the global call reference (e.g. restart procedures).

NOTE 1 – The call reference information element containing a dummy call reference is one octet long and is coded "0000 0000". The use of the dummy call reference is specified in Recommendation Q.932.

NOTE 2 – The numerical value of the global call reference is zero. The equipment receiving a message containing the global call reference should interpret the message as pertaining to all call references associated with the appropriate data link connection identifier. See Figure 4-5.

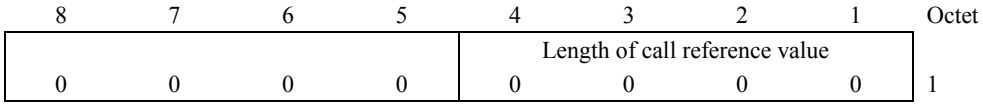


Figure 4-4/Q.931 – Dummy call reference

8	7	6	5	4	3	2	1	Octet
				Length of call reference value				
0	0	0	0	0	0	0	1	1
Flag	Call reference value							
0/1	0	0	0	0	0	0	0	2

a) One-octet call reference value

8	7	6	5	4	3	2	1	Octet
				Length of call reference value				
0	0	0	0	0	0	1	0	1
Flag	Call reference value							
0/1	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	3

b) Two-octet call reference value

Figure 4-5/Q.931 – Examples of the encoding for global call reference

4.4 Message type

The purpose of the message type is to identify the function of the message being sent.

The message type is the third part of every message. The message type is coded as shown in Figure 4-6 and Table 4-2.

Bit 8 is reserved for possible future use as an extension bit.

8	7	6	5	4	3	2	1	Octet
0	Message type							1

Figure 4-6/Q.931 – Message type

Table 4-2/Q.931 – Message types

Bits 8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	0	Escape to nationally specific message type (Note)
0	0	0	-	-	-	-	-	*Call establishment message:*
			0	0	0	0	1	– ALERTING
			0	0	0	1	0	– CALL PROCEEDING
			0	0	1	1	1	– CONNECT
			0	1	1	1	1	– CONNECT ACKNOWLEDGE
			0	0	0	1	1	– PROGRESS
			0	0	1	0	1	– SETUP
			0	1	1	0	1	– SETUP ACKNOWLEDGE
0	0	1	-	-	-	-	-	*Call information phase message:*
			0	0	1	1	0	– RESUME
			0	1	1	1	0	– RESUME ACKNOWLEDGE
			0	0	0	1	0	– RESUME REJECT
			0	0	1	0	1	– SUSPEND
			0	1	1	0	1	– SUSPEND ACKNOWLEDGE
			0	0	0	0	1	– SUSPEND REJECT
			0	0	0	0	0	– USER INFORMATION
0	1	0	-	-	-	-	-	*Call clearing messages:*
			0	0	1	0	1	– DISCONNECT
			0	1	1	0	1	– RELEASE
			1	1	0	1	0	– RELEASE COMPLETE
			0	0	1	1	0	– RESTART
			0	1	1	1	0	– RESTART ACKNOWLEDGE
0	1	1	-	-	-	-	-	*Miscellaneous messages:*
			0	0	0	0	0	– SEGMENT
			1	1	0	0	1	– CONGESTION CONTROL
			1	1	0	1	1	– INFORMATION
			0	1	1	1	0	– NOTIFY
			1	1	1	0	1	– STATUS
			1	0	1	0	1	– STATUS ENQUIRY

NOTE – When used, the message type is defined in the following octet(s), according to the national specification

4.5 Other information elements

4.5.1 Coding rules

The coding of other information elements follows the coding rules described below. These rules are formulated to allow each equipment which processes a message to find information elements important to it, and yet remain ignorant of information elements not important to that equipment.

Two categories of information elements are defined:

a) single-octet information elements [see diagrams a) and b) of Figure 4-7];

b) variable length information elements [see diagram c) of Figure 4-7].

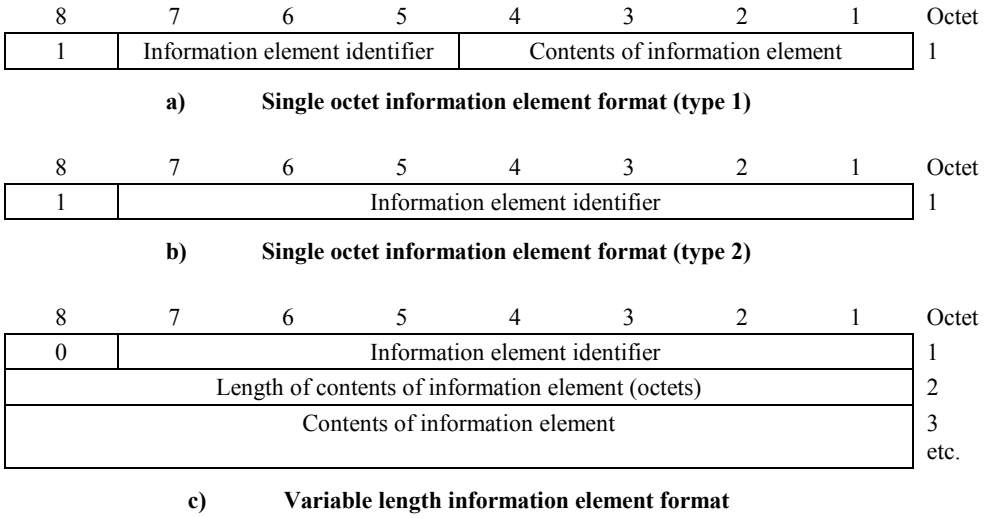


Figure 4-7/Q.931 – Formats of information elements

For the information elements listed below, the coding of the information element identifier bits is summarized in Table 4-3.

Table 4-3/Q.931 – Information element identifier coding

		Reference subclause	Maximum length (octets) (Note 1)
Bits 8 7 6 5 4 3 2 1			
1 : : : - - - -	*Single octet information elements:*		
0 0 0 - - - -	Reserved		
0 0 1 - - - -	Shift (Note 2)	4.5.3/4.5.4	1
0 1 0 0 0 0 0	More data	4.5.20	1
0 1 0 0 0 0 1	Sending complete	4.5.27	1
0 1 1 - - - -	Congestion level	4.5.14	1
1 0 1 - - - -	Repeat indicator	4.5.24	1
0 : : : : : : :	*Variable length information element:*		
0 0 0 0 0 0 0	Segmented message	4.5.26	4
0 0 0 0 1 0 0	Bearer capability (Note 2)	4.5.5	12
0 0 0 1 0 0 0	Cause (Note 2)	4.5.12	32
0 0 1 0 0 0 0	Call identity	4.5.6	10
0 0 1 0 1 0 0	Call state	4.5.7	3
0 0 1 1 0 0 0	Channel identification (Note 2)	4.5.13	(Note 4)
0 0 1 1 1 1 0	Progress indicator (Note 2)	4.5.23	4
0 1 0 0 0 0 0	Network-specific facilities (Note 2)	4.5.21	(Note 4)
0 1 0 0 1 1 1	Notification indicator	4.5.22	3
0 1 0 1 0 0 0	Display	4.5.16	34/82
0 1 0 1 0 0 1	Date/time	4.5.15	8

Table 4-3/Q.931 – Information element identifier coding *(concluded)*

		Reference subclause	Maximum length (octets) (Note 1)
Bits 8 7 6 5 4 3 2 1			
0 1 0 1 1 0 0	Keypad facility	4.5.18	34
0 1 1 0 1 0 0	Signal (Note 2)	4.5.28	3
1 0 0 0 0 0 0	Information rate	4.6.3	6
1 0 0 0 0 1 0	End-to-end transit delay	4.6.2	11
1 0 0 0 0 1 1	Transit delay selection and indication	4.6.9	5
1 0 0 0 1 0 0	Packet layer binary parameters	4.6.4	3
1 0 0 0 1 0 1	Packet layer window size	4.6.5	4
1 0 0 0 1 1 0	Packet size	4.6.6	4
1 0 0 0 1 1 1	Closed user group	4.6.1	7
1 0 0 1 0 1 0	Reverse charging indication	4.6.8	3
1 1 0 1 1 0 0	Calling party number	4.5.10	(Note 4)
1 1 0 1 1 0 1	Calling party subaddress	4.5.11	23
1 1 1 0 0 0 0	Called party number	4.5.8	(Note 4)
1 1 1 0 0 0 1	Called party subaddress	4.5.9	23
1 1 1 0 1 0 0	Redirecting number	4.6.7	(Note 4)
1 1 1 1 0 0 0	Transit network selection (Note 2)	4.5.29	(Note 4)
1 1 1 1 0 0 1	Restart indicator	4.5.25	3
1 1 1 1 1 0 0	Low layer compatibility (Note 2)	4.5.19	18
1 1 1 1 1 0 1	High layer compatibility (Note 2)	4.5.17	5
1 1 1 1 1 1 0	User-user	4.5.30	35/131
1 1 1 1 1 1 1	Escape for extension (Note 3)		
All other values are reserved (Note 5)			

NOTE 1 – The length limits described for the variable length information elements take into account only the present ITU-T standardized coding values. Future enhancements and expansions to this Recommendation will not be restricted to these limits.

NOTE 2 – This information element may be repeated.

NOTE 3 – This escape mechanism is limited to codesets 4, 5, 6 and 7 (see 4.5.2). When the escape for extension is used, the information element identifier is contained in octet-group 3 and the content of the information element follows in the subsequent octets as shown in Figure 4-8.

NOTE 4 – The maximum length is network dependent.

NOTE 5 – The reserved values with bits 5-8 coded "0000" are for future information elements for which comprehension by the receiver is required (see 5.8.7.1).

The descriptions of the information elements below are organized in alphabetical order. However, there is a particular order of appearance for each information element in a message within each codeset (see 4.5.2). The code values of the information element identifier for the variable length formats are assigned in ascending numerical order, according to the actual order of appearance of each information element in a message. This allows the receiving equipment to detect the presence or absence of a particular information element without scanning through an entire message.

Single octet information elements may appear at any point in the message. Two types of single octet information elements have been defined. Type 1 elements provide the information element identification in bit positions 7, 6, 5. The value "010" in these bit positions is reserved for Type 2 single octet elements.

Where the description of information elements in this Recommendation contains spare bits, these bits are indicated as being set to "0". In order to allow compatibility with future implementation, messages should not be rejected simply because a spare bit is set to "1".

The second octet of a variable length information element indicates the total length of the contents of that information element regardless of the coding of the first octet (i.e. the length starting with octet 3). It is the binary coding of the number of octets of the contents, with bit 1 as the least significant bit (2^0).

An optional variable-length information element may be present, but empty. For example, a SETUP message may contain a called party number information element, the content of which is of zero length. This should be interpreted by the receiver as equivalent to that information element being absent. Similarly, an absent information element should be interpreted by the receiver as equivalent to that information element being empty.

The following rules apply for the coding of variable length information elements (octets 3, etc.):

a) The first digit in the octet number identifies one octet or a group of octets.

b) Each octet group is a self-contained entity. The internal structure of an octet group may be defined in alternative ways.

c) An octet group is formed by using some extension mechanism. The preferred extension mechanism is to extend an octet (N) through the next octet(s) (Na, Nb, etc.) by using bit 8 in each octet as an extension bit. The bit value "0" indicates that the octet continues through the next octet. The bit value "1" indicates that this octet is the last octet. If one octet (Nb) is present, also the preceding octets (N and Na) must be present.

   In the format descriptions appearing in 4.5.5 etc., bit 8 is marked "0/1 ext." if another octet follows. Bit 8 is marked "1 ext." if this is the last octet in the extension domain.

   Additional octets may be defined later ("1 ext." changed to "0/1 ext.") and equipments shall be prepared to receive such additional octets although the equipment need not be able to interpret or act upon the content of these octets.

d) In addition to the extension mechanism defined above, an octet (N) may be extended through the next octet(s) (N1, N2 etc.) by indications in bits 7-1 (of octet N).

e) The mechanisms in c) and d) may be combined. Mechanism c) shall take priority in the ordering, such that all octets Na, Nb, etc. shall occur before octets N1, N2, etc. This rule shall apply even where the extension to octets N1, N2, etc. is indicated in one of octet Na, Nb, etc.

f) Similar conventions apply even when mechanism d) is being repeated, i.e. octets N.1 shall occur before octets N.1.1, N.1.2, etc.

g) Optional octets are marked with asterisks (*).

NOTE 1 – It is not possible to use mechanism c) repeatedly, i.e. it is not possible to construct an octet 4a as this would become octet 4b.

NOTE 2 – Protocol designers should exercise care in using multiple extension mechanisms to ensure that a unique interpretation of the resultant coding is possible.

NOTE 3 – For a number of information elements, there is a field that defines the coding standard. When the coding standard defines a national standard, it is recommended that the national standard be structured similar to the information element defined in this Recommendation.

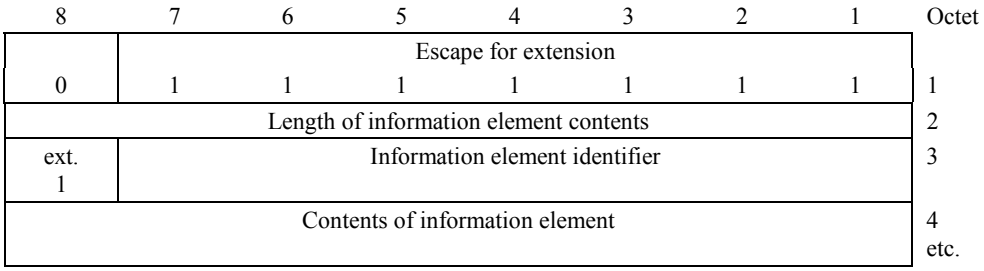


Figure 4-8/Q.931 – Information element format using escape for extension

4.5.2 Extensions of codesets

There is a certain number of possible information element identifier values using the formatting rules described in 4.5.1; 128 from the variable length information element format and at least 8 from the single octet information element format.

One value in the single octet format is specified for shift operations described below. One other value in both the single octet and variable format is reserved. This leaves at least 133 information element identifier values available for assignment.

It is possible to expand this structure to eight codesets of at least 133 information element identifier values each. One common value in the single octet format is employed in each codeset to facilitate shifting from one codeset to another. The contents of this Shift information element identifies the codeset to be used for the next information element or elements. The codeset in use at any given time is referred to as the "active codeset". By convention, codeset 0 is the initially active codeset.

Two codeset shifting procedures are supported: locking shift and non-locking shift.

Codeset 4 is reserved for use by ISO/IEC Standards.

Codeset 5 is reserved for information elements reserved for national use.

Codeset 6 is reserved for information elements specific to the local network (either public or private).

Codeset 7 is reserved for user-specific information elements.

The coding rules specified in 4.5.1 shall apply for information elements belonging to any active codeset.

Transitions from one active codeset to another (i.e. by means of the locking shift procedure) may only be made to a codeset with a higher numerical value than the codeset being left.

An information element belonging to codesets 4, 5, 6, or 7 may appear together with information elements belonging to codeset 0 (being the active codeset) by using the non-locking shift procedure (see 4.5.4).

A user of network equipment shall have the capability to recognize a Shift information element and to determine the length of the following information element, although the equipment need not be able to interpret and act upon the content of the information element. This enables the equipment to determine the start of a subsequent information element.

Codeset 7 information element shall be handled according to the procedures for unrecognized information elements (see 5.8.7.1) by the first exchange in the local network, unless allowed by a future service definition, bilateral agreement, or provision is made to support this across the local network for a specific user.

Codeset 6 is reserved for information elements specific to the local network (either public or private). As such they do not have significance across the boundaries between local networks, or across a national, or international boundary. Therefore, codeset 6 information elements shall be handled according to the procedures for unrecognized information elements (see 5.8.7.1) beyond local network boundary, unless allowed by bilateral agreement.

Codeset 5 is reserved for information elements reserved for national use. As such they do not have significance across an international boundary. Therefore, codeset 5 information elements shall be handled according to the procedures for unrecognized information elements (see 5.8.7.1) at the first exchange beyond the international boundary, unless there are bilateral agreements to the contrary.

Codeset 4 is reserved for information elements specified in ISO/IEC Standards.

4.5.3 Locking shift procedure

The locking shift procedure employs an information element to indicate the new active codeset. The specified codeset remains active until another locking shift information element is encountered which specifies the use of another codeset. For example, codeset 0 is active at the start of message content analysis. If a locking shift to codeset 5 is encountered, the next information elements will be interpreted according to the information element identifiers assigned in codeset 5, until another shift information element is encountered.

This procedure is used only to shift to a higher order codeset than the one being left.

The locking shift is valid only within that message which contains the locking Shift information element. At the start of every message content analysis, the active codeset is codeset 0.

The locking Shift information element uses the single octet information element format and coding shown in Figure 4-9 and Table 4-4.

4.5.4 Non-locking shift procedure

The non-locking shift procedure provides a temporary shift to the specified lower or higher codeset. The non-locking shift procedure uses a single octet information element to indicate the codeset to be used to interpret the next single information element. After the interpretation of the next single information element, the active codeset is again used for interpreting any following information elements. For example, codeset 0 is active at the beginning of message content analysis. If a non-locking shift to codeset 6 is encountered, only the next information element is interpreted according to the information element identifiers assigned in codeset 6. After this information element is interpreted, codeset 0 will again be used to interpret the following information elements. A non-locking Shift information element indicating the current codeset shall not be regarded as an error.

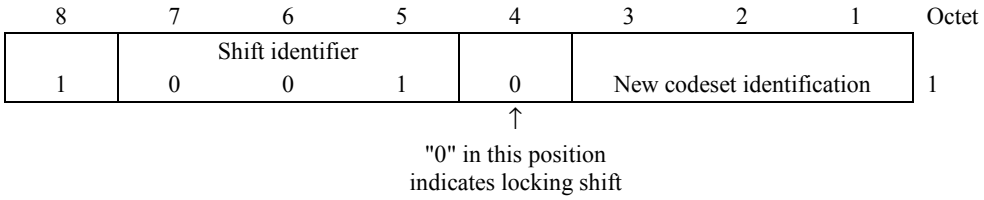


Figure 4-9/Q.931 – Locking Shift information element

Table 4-4/Q.931 – Locking Shift information element

Codeset identification (bits 3 to 1)	
Bits 3 2 1	
0 0 0	Not applicable
0 0 1 to 0 1 1	Reserved
1 0 0	Codeset 4: information elements for ISO/IEC use
1 0 1	Codeset 5: information elements for national use
1 1 0	Codeset 6: information elements specific to the local network (either public or private)
1 1 1	Codeset 7: user-specific information elements

A locking Shift information element shall not follow directly on a non-locking Shift information element. If this combination is received, it shall be interpreted as though a locking Shift information element only had been received.

The non-locking Shift information element uses the single octet information element format and coding shown in Figure 4-10 and Table 4-5.

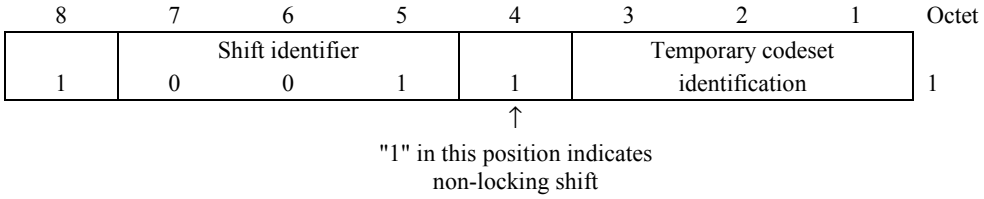


Figure 4-10/Q.931 – Non-locking Shift information element

Table 4-5/Q.931 – Non-locking Shift information element

Codeset identification (bits 3 to 1)	
Bits 3 2 1	
0 0 0	Codeset 0 (initially active): Q.931 information elements
0 0 1 to 0 1 1	Reserved
1 0 0	Codeset 4: information elements for ISO/IEC use
1 0 1	Codeset 5: information elements for national use
1 1 0	Codeset 6: information elements specific to the local network (either public or private)
1 1 1	Codeset 7: user-specific information elements

4.5.5 Bearer capability

The purpose of the Bearer capability information element is to indicate a requested I.231 [6] bearer service to be provided by the network. It contains only information which *may* be used by the network (see Annex I). The use of the Bearer capability information element in relation to compatibility checking is described in Annex B.

The Bearer capability information element is coded as shown in Figure 4-11 and Table 4-6.

No default bearer capability may be assumed by the absence of this information element.

The maximum length of this information element is 12 octets.

NOTE – Future extensions to the codings of the Bearer capability information element should not be in conflict with the initially defined coding of the Low layer compatibility information element. See 4.5.19.

8	7	6	5	4	3	2	1	Octet
	Bearer capability information element identifier							
0	0	0	0	0	1	0	0	1
Length of the bearer capability contents								2
ext. 1	Coding standard		Information transfer capability					3
ext. 1	Transfer mode		Information transfer rate					4
ext. 1	Rate multiplier							4.1* (Note 1)
ext. 0/1	Layer 1 ident. 0 1		User information layer 1 protocol					5*
ext. 0/1	Synch./ asynch	Negot.	User rate					5a* (Note 2)
ext. 0/1	Intermediate rate		NIC on Tx	NIC on Rx	Flow control on Tx	Flow control on Rx	Spare 0	5b* (Note 3)
ext. 0/1	Hdr/ no Hdr	Multiframe	Mode	LLI negot.	Assignor/ee	In-band neg.	Spare 0	5b* (Note 4)
ext. 0/1	Number of stop bits		Number of data bits		Parity			5c* (Note 2)
ext. 1	Duplex mode	Modem type						5d* (Note 2)
ext. 1	Layer 2 ident. 1 0		User information layer 2 protocol					6*
ext. 0	Layer 3 ident. 1 1		User information layer 3 protocol					7*
ext. 0	Spare 0 0 0		Additional layer 3 protocol information (most significant bits)					7a* (Note 5)
ext. 1	Spare 0 0 0		Additional layer 3 protocol information (most significant bits)					7b* (Note 5)

NOTE 1 – This octet is required if octet 4 indicates multirate (64 kbit/s base rate). Otherwise, it shall not be present.

NOTE 2 – This octet may be present if octet 3 indicates *unrestricted digital information* and octet 5 indicates either of the ITU-T standardized rate adaptions V.110, I.460 and X.30 or V.120 [9]. It may also be present if octet 3 indicates 3.1 kHz audio and octet 5 indicates G.711.

NOTE 3 – This structure of octet 5b only applies if octet 5 indicates ITU-T standardized rate adaption (see Recommendations V.110 [7], I.460 [15] and X.30 [8]).

NOTE 4 – This structure of octet 5b only applies if octet 5 indicates ITU-T standardized rate adaption (see Recommendation V.120 [9]).

NOTE 5 – This octet may be included if octet 7 indicates ISO/IEC TR 9577 (Protocol Identification in the network layer).

Figure 4-11/Q.931 – Bearer capability information element

Table 4-6/Q.931 – Bearer capability information element

Coding standard (octet 3)

Bits

7	6	
0	0	ITU-T standardized coding as described below
0	1	ISO/IEC Standard (Note 1)
1	0	National standard (Note 1)
1	1	Standard defined for the network (either public or private) present on the network side of the interface (Note 1)

NOTE 1 – These other coding standards should be used only when the desired bearer capability cannot be represented with the ITU-T-standardized coding.

Information transfer capability (octet 3)

Bits

5	4	3	2	1	
0	0	0	0	0	Speech
0	1	0	0	0	Unrestricted digital information
0	1	0	0	1	Restricted digital information
1	0	0	0	0	3.1 kHz audio
1	0	0	0	1	Unrestricted digital information with tones/announcements (Note 2)
1	1	0	0	0	Video

All other values are reserved.

NOTE 2 – Unrestricted digital information with tones/announcements (UDI-TA) is the new information transfer attribute value that had previously been named "7 kHz audio" in Recommendation Q.931 (1988).

Transfer mode (octet 4)

Bits

7	6	
0	0	Circuit mode
1	0	Packet mode

All other values are reserved.

Information transfer rate (octet 4 , bits 5 to 1)

Bits

5	4	3	2	1	C*ircuit mode*	*Packet-mode*
0	0	0	0	0	–	This code shall be used for packet-mode calls
1	0	0	0	0	64 kbit/s	–
1	0	0	0	1	2 × 64 kbit/s	–
1	0	0	1	1	384 kbit/s	–
1	0	1	0	1	1536 kbit/s	–
1	0	1	1	1	1920 kbit/s	–
1	1	0	0	0	Multirate (64 kbit/s base rate)	

All other values are reserved.

NOTE 3 – When the information transfer rate 2 × 64 kbit/s is used, the coding of octets 3 and 4 refer to both 64 kbit/s channels.

NOTE 4 – Additional attributes are defined in Table 4-7.

Rate multiplier (octet 4.1)

NOTE 5 – Coded as a binary representation of the multiplier to the base rate. The multiplier can take any value from 2 up to the maximum number of B-channels available on the interface.

Table 4-6/Q.931 – Bearer capability information element *(continued)*

User information layer 1 protocol (octet 5)

Bits

5	4	3	2	1	
0	0	0	0	1	ITU-T standardized rate adaption V.110, I.460 and X.30. This implies the presence of octet 5a and optionally octets 5b, 5c and 5d as defined below
0	0	0	1	0	Recommendation G.711 [10] μ-law
0	0	0	1	1	Recommendation G.711 A-law
0	0	1	0	0	Recommendation G.721 [11] 32 kbit/s ADPCM and Recommendation I.460
0	0	1	0	1	Recommendations H.221 and H.242
0	0	1	1	0	Recommendations H.223 [92] and H.245 [93]
0	0	1	1	1	Non-ITU-T standardized rate adaption. This implies the presence of octet 5a and, optionally, octets 5b, 5c and 5d. The use of this codepoint indicates that the user rate specified in octet 5a is defined by the user. Additionally, octets 5b, 5c and 5d, if present, are defined in accordance with the user specified rate adaption
0	1	0	0	0	ITU-T standardized rate adaption V.120 [9]. This implies the presence of octets 5a and 5b as defined below, and optionally octets 5c and 5d
0	1	0	0	1	ITU-T standardized rate adaption X.31 [14] HDLC flag stuffing

All other values are reserved.

NOTE 6 – If the transfer mode is "circuit mode", and if the information transfer capability is "unrestricted digital information" or "restricted digital information", and if the user information layer 1 protocol is to be identified only to the addressed entity octet 5 shall be omitted. If the transfer mode is packet mode, octet 5 may be omitted. Otherwise, octet 5 shall be present.

Synchronous/Asynchronous (octet 5a)

Bit

7	
0	Synchronous data
1	Asynchronous data

NOTE 7 – Octets 5b-5d may be omitted in the case of synchronous user rates.

Negotiation (octet 5a)

Bit

6	
0	In-band negotiation not possible
1	In-band negotiation possible

NOTE 8 – See Recommendations V.110 [7], I.460 [15] and X.30 [8] or modem type Recommendation.

User rate (octet 5a)

Bits

5	4	3	2	1	
0	0	0	0	0	For I.460, rate is specified by bits 7, 6 of octet 5b, intermediate rate. For V.110 and X.30, rate is indicated by E-bits (synchronous data only) or may be negotiated in-band. For V.120, rate is unspecified or may be negotiated in-band.
0	0	0	0	1	0.6 kbit/s Recommendation X.1 [17]
0	0	0	1	0	1.2 kbit/s
0	0	0	1	1	2.4 kbit/s Recommendation X.1
0	0	1	0	0	3.6 kbit/s
0	0	1	0	1	4.8 kbit/s Recommendation X.1
0	0	1	1	0	7.2 kbit/s
0	0	1	1	1	8 kbit/s Recommendation I.460

Table 4-6/Q.931 – Bearer capability information element *(continued)*

0 1 0 0 0 9.6 kbit/s Recommendation X.1
0 1 0 0 1 14.4 kbit/s
0 1 0 1 0 16 kbit/s Recommendation I.460
0 1 0 1 1 19.2 kbit/s
0 1 1 0 0 32 kbit/s Recommendation I.460
0 1 1 0 1 38.4 kbit/s Recommendation V.110 [87]
0 1 1 1 0 48 kbit/s Recommendations X.1
0 1 1 1 1 56 kbit/s
1 0 0 1 0 57.6 kbit/s Recommendation V.14 extended [88]
1 0 0 1 1 28.8 kbit/s Recommendation V.110 [89]
1 0 1 0 0 24 kbit/s Recommendation V.110 [89]
1 0 1 0 1 0.1345 kbit/s Recommendation X.1
1 0 1 1 0 0.100 kbit/s Recommendation X.1
1 0 1 1 1 0.075/1.2 kbit/s Recommendation X.1 (Note 9)
1 1 0 0 0 1.2/0.075 kbit/s Recommendation X.1 (Note 9)
1 1 0 0 1 0.050 kbit/s Recommendation X.1
1 1 0 1 0 0.075 kbit/s Recommendation X.1
1 1 0 1 1 0.110 kbit/s Recommendation X.1
1 1 1 0 0 0.150 kbit/s Recommendation X.1
1 1 1 0 1 0.200 kbit/s Recommendation X.1
1 1 1 1 0 0.300 kbit/s Recommendation X.1
1 1 1 1 1 12 kbit/s

All other values are reserved.

NOTE 9 – The first rate is the transmit rate in the forward direction of the call. The second rate is the transmit rate in the backward direction of the call.

Octet 5b for V.110, I.460 and X.30 rate adaption

Intermediate rate (octet 5b)

Bits 7 6	
0 0	Not used
0 1	8 kbit/s
1 0	16 kbit/s
1 1	32 kbit/s

Network Independent Clock (NIC) on transmission (Tx) (octet 5b) (Note 10)

Bit 5	
0	Not required to send data with network independent clock
1	Required to send data with network independent clock

NOTE 10 – Refers to transmission in the forward direction of the call.

NOTE 11 – See Recommendations V.110 [7], I.460 [15] and X.30 [8].

Table 4-6/Q.931 – Bearer capability information element *(continued)*

Network Independent Clock (NIC) on reception (Rx) (octet 5b) (Note 12)

Bit
4

0 Cannot accept data with network independent clock (i.e. sender does not support this optional procedure).

1 Can accept data with network independent clock (i.e. sender does support this optional procedure).

NOTE 12 – Refers to transmission in the backward direction of the call.
NOTE 13 – See Recommendations V.110 [7], I.460 [15] and X.30 [8].

Flow control on transmission (Tx) (octet 5b) (Note 14)

Bit
3

0 Not required to send data with flow control mechanism

1 Required to send data with flow control mechanism

NOTE 14 – Refers to transmission in the forward direction of the call.
NOTE 15 – See Recommendations V.110, I.460 and X.30.

Flow control on reception (Rx) (octet 5b) (Note 16)

Bit
2

0 Cannot accept data with flow control mechanism (i.e. sender does not support this optional procedure)

1 Can accept data with flow control mechanism (i.e. sender does support this optional procedure)

NOTE 16 – Refers to transmission in the backward direction of the call.
NOTE 17 – See Recommendations V.110, I.460 and X.30.

Octet 5b for V.120 [9] rate adaption

Rate adaption header/no header (octet 5b)

Bit
7

0 Rate adaption header not included

1 Rate adaption header included

Multiple frame establishment support in data link (octet 5b)

Bit
6

0 Multiple frame establishment not supported. Only UI frames allowed

1 Multiple frame establishment supported

Mode of operation (octet 5b)

Bit
5

0 Bit transparent mode of operation

1 Protocol sensitive mode of operation

Logical link identifier negotiation (octet 5b)

Bit
4

0 Default, LLI = 256 only

1 Full protocol negotiation (Note 18)

NOTE 18 – A connection over which protocol negotiation will be executed is indicated in bit 2 of octet 5b.

Table 4-6/Q.931 – Bearer capability information element *(continued)*

Assignor/assignee (octet 5b)

Bit
3

0 Message originator is "Default assignee"

1 Message originator is "Assignor only"

In-band/out-band negotiation (octet 5b)

Bit
2

0 Negotiation is done with USER INFORMATION messages on a temporary signalling connection

1 Negotiation is done in-band using logical link zero

Number of stop bits (octet 5c)

Bits
7 6

0 0 Not used

0 1 1 bit

1 0 1.5 bits

1 1 2 bits

Number of data bits excluding parity Bit if present (octet 5c)

Bits
5 4

0 0 Not used

0 1 5 bits

1 0 7 bits

1 1 8 bits

Parity information (octet 5c)

Bits
3 2 1

0 0 0 Odd

0 1 0 Even

0 1 1 None

1 0 0 Forced to 0

1 0 1 Forced to 1

All other values are reserved.

Mode duplex (octet 5d)

Bit
7

0 Half duplex

1 Full duplex

Table 4-6/Q.931 – Bearer capability information element *(continued)*

Modem type (octet 5d)

Bits
6 5 4 3 2 1

0 0 0 0 0 0 through 0 0 0 1 0 1	National use
0 1 0 0 0 1	Recommendation V.21 [55]
0 1 0 0 1 0	Recommendation V.22 [56]
01 0 0 1 1	Recommendation V.22 *bis* [57]
0 1 0 1 0 0	Recommendation V.23 [58]
0 1 0 1 0 1	Recommendation V.26 [59]
0 1 0 1 1 0	Recommendation V.26 *bis* [60]
0 1 0 1 1 1	Recommendation V.26 *ter* [61]
0 1 1 0 0 0	Recommendation V.27 [62]
0 1 1 0 0 1	Recommendation V.27 *bis* [63]
0 1 1 0 1 0	Recommendation V.27 *ter* [64]
0 1 1 0 1 1	Recommendation V.29 [65]
0 1 1 1 0 1	Recommendation V.32 [66]
0 1 1 1 1 0	Recommendation V.34 [90]
1 0 0 0 0 0 through 1 0 1 1 1 1	National use
1 1 0 0 0 0 through 1 1 1 1 1 1	User specified

All other values reserved.

User information layer 2 protocol (octet 6)

Bits
5 4 3 2 1

0 0 0 1 0	Recommendation Q.921/I.441 [3]
0 0 1 1 0	Recommendation X.25 [5], link layer
0 1 1 0 0	LAN logical link control (ISO/IEC 8802-2) (Note 23)

All other values are reserved.

NOTE 19 – If the transfer mode is "packet mode", octet 6 shall be present. For other cases, if the user layer 2 protocol is to be identified to the network, then octet 6 shall be present; otherwise octet 6 shall be omitted.

User information layer 3 protocol (octet 7)

Bits
5 4 3 2 1

0 0 0 1 0	Recommendation Q.931
0 0 1 1 0	Recommendation X.25, packet layer
0 1 0 1 1	ISO/IEC TR 9577 [82] (Protocol identification in the network layer) (Notes 21 and 23)

All other values are reserved.

NOTE 20 – If the user information layer 3 protocol is to be identified to the network, octet 7 shall be present; otherwise octet 7 shall be omitted.

NOTE 21 – If the user information layer 3 protocol indicates "Network layer protocol identification", octet 7a and 7b may be included to identify the actual user information layer 3 protocol to the network.

Table 4-6/Q.931 – Bearer capability information element *(concluded)*

Octets 7a and 7b (Notes 21 and 22)

Bit 8 (ext.) set to 0 in octet 7a and set to 1 in octet 7b.

Bits 7 to 5 are spare (set to 0) in both octets.

7a Bits 4 3 2 1	*7b* Bits 4 3 2 1	
1 1 0 0	1 1 0 0	Internet Protocol (RFC 791) (ISO/IEC TR 9577 [82])
1 1 0 0	1 1 1 1	Point-to-point Protocol (RFC 1548)

NOTE 22 – If the user information layer 3 protocol indicates "Network layer protocol Identification", octet 7a and 7b may be included to identify the actual user information layer 3 protocol to the network. These codepoints are assigned consistently with ISO/IEC TR 9577 [82].

NOTE 23 – These codings can only be used where transfer mode is "circuit mode".

Table 4-7/Q.931 – Bearer capability attributes

BC attributes		**Additional attributes**			
Transfer mode	**Information transfer capability**	**Structure**	**Configuration**	**Establishment**	**Symmetry**
Circuit	Speech	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Unrestricted data	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Restricted data	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	3.1 kHz audio	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Unrestricted data with tones/ announcements	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Video	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Packet	Unrestricted data	Service data unit integrity	Point-to-point	Demand	Bi-directional symmetric

NOTE 1 – When the information transfer rate 2×64 kbit/s is used, 8 kHz integrity with Restricted Differential Time Delay (RDTD) is offered.

NOTE 2 – When multirate (64 kbit/s base rate) is indicated as the information transfer rate, Time Slot Sequence integrity shall be provided.

4.5.6 Call identity

The purpose of the Call identity information element is to identify the suspended call. The call identity provided by the user is guaranteed by the network to be unique over the user-network interface on which the user resides. The call identity is assigned at the start of the call suspension, and is available for reuse after the resume procedure has completed successfully.

The Call identity information element is coded as shown in Figure 4-12.

The default maximum length of this information element is ten octets.

4.5.7 Call state

The purpose of the Call state information element is to describe the current status of a call, (see 2.1) or an access-connection (see 2.2) or a global interface state (see 2.4).

The Call state information element is coded as shown in Figure 4-13 and Table 4-8.

The length of this information element is three octets.

8	7	6	5	4	3	2	1	Octet
	Call identity information element identifier							
0	0	0	1	0	0	0	0	1
Length of call identity contents								2
Call identity (any bit pattern allowed, e.g. IA5 characters)								3 etc.

Figure 4-12/Q.931 – Call identity information element

8	7	6	5	4	3	2	1	Octet
	Call state information element identifier							
0	0	0	1	0	1	0	0	1
Length of call state contents								2
Coding standard		Call state value / global interface state value (state value is coded in binary)						3

Figure 4-13/Q.931 – Call state information element

Table 4-8/Q.931 – Call state information element

Coding standard (octet 3)

Bits

8 7

0 0 ITU-T standardized coding, as described below

0 1 ISO/IEC standard (Note)

1 0 National standard (Note)

1 1 Standard defined for the network (either public or private) present on the network side of the interface (Note)

NOTE – These other coding standards should only be used only when the desired call states cannot be represented by ITU-T-standardized coding.

Table 4-8/Q.931 – Call state information element *(concluded)*

Call state value (octet 3)

Bits 6 5 4 3 2 1	*User State*		*Network state*	
0 0 0 0 0 0	U0	– Null	N0	– Null
0 0 0 0 0 1	U1	– Call initiated	N1	– Call initiated
0 0 0 0 1 0	U2	– Overlap sending	N2	– Overlap sending
0 0 0 0 1 1	U3	– Outgoing call proceeding	N3	– Outgoing call proceeding
0 0 0 1 0 0	U4	– Call delivered	N4	– Call delivered
0 0 0 1 1 0	U6	– Call present	N6	– Call present
0 0 0 1 1 1	U7	– Call received	N7	– Call received
0 0 1 0 0 0	U8	– Connect request	N8	– Connect request
0 0 1 0 0 1	U9	– Incoming call proceeding	N9	– Incoming call proceeding
0 0 1 0 1 0	U10	– Active	N10	– Active
0 0 1 0 1 1	U11	– Disconnect request	N11	– Disconnect request
0 0 1 1 0 0	U12	– Disconnect indication	N12	– Disconnect indication
0 0 1 1 1 1	U15	– Suspend request	N15	– Suspend request
0 1 0 0 0 1	U17	– Resume request	N17	– Resume request
0 1 0 0 1 1	U19	– Release request	N19	– Release request
0 1 0 1 1 0		– - - - - -	N22	– Call abort
0 1 1 0 0 1	U25	– Overlap receiving	N25	– Overlap receiving

Global interface state value (octet 3)

Bits 6 5 4 3 2 1	*State*
0 0 0 0 0 0	REST0 – Null
1 1 1 1 0 1	REST1 – Restart request
1 1 1 1 1 0	REST2 – Restart

All other values are reserved.

4.5.8 Called party number

The purpose of the Called party number information element is to identify the called party of a call.

The Called party number information element is coded as shown in Figure 4-14 and Table 4-9.

The maximum length of this information element is network dependent.

8	7	6	5	4	3	2	1	Octet
	Called party number information element identifier							
0	1	1	1	0	0	0	0	1
Length of called party number contents								2
ext. 1	Type of number			Numbering plan identification				3
0	Number digits (IA5 characters) (Note)							4 etc.

NOTE – The number digits appear in multiple octet 4s in the same order in which they would be entered, that is, the number digit which would be entered first is located in the first octet 4.

Figure 4-14/Q.931 – Called party number information element

Table 4-9/Q.931 – Called party number information element

Type of number (octet 3) (Note 1)

Bits 7 6 5	
0 0 0	Unknown (Note 2)
0 0 1	International number (Note 3)
0 1 0	National number (Note 3)
0 1 1	Network specific number (Note 4)
1 0 0	Subscriber number (Note 3)
1 1 0	Abbreviated number (Note 5)
1 1 1	Reserved for extension

All other values are reserved.

NOTE 1 – For the definition of international, national and subscriber number, see Recommendation I.330 [18].

NOTE 2 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan, e.g. prefix or escape digits might be present.

NOTE 3 – Prefix or escape digits shall not be included.

NOTE 4 – The type of number "network specific number" is used to indicate administration/service number specific to the serving network, e.g. used to access an operator.

NOTE 5 – The support of this code is network dependent. The number provided in this information element presents a shorthand representation of the complete number in the specified numbering plan as supported by the network.

Numbering plan identification (octet 3)

Numbering plan (applies for type of number = 000, 001, 010 and 100)

Bits 4 3 2 1	
0 0 0 0	Unknown (Note 6)
0 0 0 1	ISDN/telephony numbering plan (Recommendation E.164 [19])
0 0 1 1	Data numbering plan (Recommendation X.121 [21])
0 1 0 0	Telex numbering plan (Recommendation F.69 [22])
1 0 0 0	National standard numbering plan
1 0 0 1	Private numbering plan
1 1 1 1	Reserved for extension

All other values are reserved.

NOTE 6 – The numbering plan "unknown" is used when the user or network has no knowledge of the numbering plan. In this case, the number digits field is organized according to the network dialling plan, e.g. prefix or escape digits might be present.

Number digits (octets 4, etc.)

This field is coded with IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

4.5.9 Called party subaddress

The purpose of the Called party subaddress information element is to identify the subaddress of the called party of the call. The network does not interpret this information. For the definition of subaddress, see Recommendation I.330 [18].

The Called party subaddress information element is coded as shown in Figure 4-15 and Table 4-10.

The maximum length of this information element is 23 octets.

8	7	6	5	4	3	2	1	Octet
	Called party subaddress information element identifier							
0	1	1	1	0	0	0	1	1
Length of called party subaddress contents								2
ext.	Type of subaddress			Odd/even	Spare			3
1				indicator	0	0	0	
Subaddress information								4 etc.

Figure 4-15/Q.931 – Called party subaddress information element

Table 4-10/Q.931 – Called party subaddress information element

Type of subaddress (octet 3)

Bits

7 6 5

0 0 0 NSAP (ITU-T Rec. X.213 [23] and ISO/IEC 8348 Add.2 [24])

0 1 0 User specified

All other values are reserved.

Odd/even indicator (octet 3)

Bit

4

0 Even number of address signals

1 Odd number of address signals

NOTE 1 – The odd/even indicator is used when the type of subaddress is "user specified" and the coding is BCD.

Subaddress information (octets 4, etc.)

The NSAP X.213 and ISO/IEC 8348, Add.2 address shall be formatted as specified by octet 4 which contains the Authority and Format Identifier (AFI). The encoding is made according to the "preferred binary encoding" as defined in ITU-T Rec. X.213 and ISO/IEC 8348, Add.2 except when used for Terminal selection at the S interface (see Note 3). For the definition of this type of subaddress, see Recommendation I.334 [25].

For user specified subaddress, this field is encoded according to the user specification, subject to a maximum length of 20 octets. When interworking with X.25 [5] networks, BCD coding should be applied.

NOTE 2 – It is recommended that users apply the NSAP subaddress type since this subaddress type allows the use of decimal, binary and IA5 syntaxes in a standardized manner.

NOTE 3 – It is recommended that users apply the Local IDI format (the AFI field coded 50 in BCD) when the subaddress is used for terminal selection purposes at the S interface. In this case, the IA5 character syntax using only digits 0 to 9 shall be used for the DSP. Each character is then encoded in one octet according to Recommendation T.50 and ISO/IEC 646, with zero parity in the most significant position.

4.5.10 Calling party number

The purpose of the Calling party number information element is to identify the origin of a call.

The Calling party number information element is coded as shown in Figure 4-16 and Table 4-11.

The maximum length of this information element is network dependent.

8	7	6	5	4	3	2	1	Octet
	Calling party number information element identifier							
0	1	1	0	1	1	0	0	1
Length of calling party number contents								2
ext. 0/1	Type of number			Numbering plan identification				3
ext. 1	Presentation indicator		Spare 0	0	0	Screening indicator		3a*
0	Number digits (IA5 characters)							4*

Figure 4-16/Q.931 – Calling party number information element

Table 4-11/Q.931 – Calling party number information element

Type of number (octet 3) (Note 1)

Bits

7 6 5

0 0 0 Unknown (Note 2)
0 0 1 International number (Note 3)
0 1 0 National number (Note 3)
0 1 1 Network specific number (Note 4)
1 0 0 Subscriber number (Note 3)
1 1 0 Abbreviated number (Note 5)
1 1 1 Reserved for extension

All other values are reserved.

NOTE 1 – For the definition of international, national and subscriber number, (see Recommendation I.330 [18]).

NOTE 2 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

NOTE 3 – Prefix or escape digits shall not be included.

NOTE 4 – The type of number "network specific number" is used to indicate administration/service number specific to the serving network, e.g. used to access an operator.

NOTE 5 – The support of this code is network dependent. The number provided in this information element presents a shorthand representation of the complete number in the specified numbering plan as supported by the network.

Numbering plan identification (octet 3)

Numbering plan (applies for type of number = 000, 001, 010 and 100)

Bits

4 3 2 1

0 0 0 0 Unknown (Note 6)
0 0 0 1 ISDN/telephony numbering plan (Recommendation E.164 [19])
0 0 1 1 Data numbering plan (Recommendation X.121 [21])
0 1 0 0 Telex numbering plan (Recommendation F.69 [22])
1 0 0 0 National standard numbering plan
1 0 0 1 Private numbering plan
1 1 1 1 Reserved for extension

All other values are reserved.

Table 4-11/Q.931 – Calling party number information element *(concluded)*

NOTE 6 – The numbering plan "unknown" is used when the user or network has no knowledge of the numbering plan. In this case, the number digits field is organized according to the network dialling plan, e.g. prefix or escape digits might be present.

Presentation indicator (octet 3a)

Bits 7 6	Meaning
0 0	Presentation allowed
0 1	Presentation restricted
1 0	Number not available due to interworking
1 1	Reserved

NOTE 7 – The meaning and the use of this field is defined in clause 3/Q.951 and clause 4/Q.951.

Screening indicator (octet 3a)

Bits 2 1	
0 0	User-provided, not screened
0 1	User-provided, verified and passed
1 0	User-provided, verified and failed
1 1	Network provided

NOTE 8 – The meaning and the use of this field is defined in clause 3/Q.951 and clause 4/Q.951.

Number digits (octets 4, etc.)

This field is coded with IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

4.5.11 Calling party subaddress

The purpose of the Calling party subaddress information element is to identify a subaddress associated with the origin of a call. For the definition of subaddress, see Recommendation I.330 [18].

The Calling party subaddress information element is coded as shown in Figure 4-17 and Table 4-12.

The maximum length of this information element is 23.

8	7	6	5	4	3	2	1	Octet
	Calling party subaddress information element identifier							
0	1	1	0	1	1	0	1	1
Length of calling party subaddress contents								2
ext.	Type of subaddress			Odd/even	Spare			3
1				indicator	0	0	0	
Subaddress information								4
								etc.

Figure 4-17/Q.931 – Calling party subaddress information element

Table 4-12/Q.931 – Calling party subaddress information element

Type of subaddress (octet 3)

Bits

7 6 5

0 0 0 NSAP (ITU-T Rec. X.213 [23] and ISO/IEC 8348, Add.2 [24])

0 1 0 User specified

All other values are reserved.

Odd/even indicator (octet 3)

Bit

4

0 Even number of address signals

1 Odd number of address signals

NOTE 1 – The odd/even indicator is used when the type of subaddress is "user specified" and the coding is BCD.

Subaddress information (octets 4, etc.)

The NSAP ITU-T Rec. X.213 and ISO/IEC 8348, Add.2 address shall be formatted as specified by octet 4 which contains the Authority and Format Identifier (AFI). The encoding is made according to the "preferred binary encoding" as defined in ITU-T Rec. X.213and ISO/IEC 8348, Add.2 except when used for Terminal selection at the S interface (see Note 3). For the definition of this type of subaddress, see Recommendation I.334 [25].

For user specified subaddress, this field is encoded according to the user specification, subject to a maximum length of 20 octets. When interworking with X.25 [5] networks, BCD coding should be applied.

NOTE 2 – It is recommended that users apply the NSAP subaddress type since this subaddress type allows the use of decimal, binary and IA5 syntaxes in a standardized manner.

NOTE 3 – It is recommended that users apply the Local IDI format (the AFI field coded 50 in BCD) when the subaddress is used for terminal selection purposes at the S interface. In this case, the IA5 character syntax using only digits 0 to 9 shall be used for the DSP. Each character is then encoded in one octet according to Recommendation T.50 [49] and ISO/IEC 646, with zero parity in the most significant position.

4.5.12 Cause

The content and use of the Cause information element is defined in Recommendation Q.850 [67].

4.5.13 Channel identification

The purpose of the Channel identification information element is to identify a channel within the interface(s) controlled by these signalling procedures.

The Channel identification information element is coded as shown in Figures 4-18 and 4-19 and Table 4-13. The channel identification element may be repeated in a message, e.g. to list several acceptable channels during channel negotiation.

The default maximum length for this information element is network dependent.

8	7	6	5	4	3	2	1	Octet
Channel identification information element identifier								
0	0	0	1	1	0	0	0	1
Length of channel identification contents								2
ext. 1	Int. id. present	Int. type	Spare 0	Pref./Excl.	D-channel ind	Info. channel selection		3
ext. 0/1	Interface identifier							3.1*, etc. (Note 1)
ext. 1	Coding standard		Number/ Map	Channel type/Map element type				3.2* (Note 2) (Note 5)
Channel number/Slot map (Note 3)								3.3* (Note 2) (Note 4) (Note 5)

NOTE 1 – When the "interface identifier present" field in octet 3 indicates "interface implicitly identified" octet 3.1 is omitted. When octet 3.1 is present, it may be extended by using the extension bit (bit 8).

NOTE 2 – When the "interface type" field in octet 3 indicates "basic interface", octets 3.2 and 3.3 are functionally replaced by the "information channel selection" field in octet 3, and thus omitted.

NOTE 3 – When channel number is used and a single channel is indicated, bit 8 shall be set to "1". When channel number is used and multiple channels are indicated, bit 8 shall be used as an extension bit to indicate an extension to subsequent channels and coded according to the rules specified in 4.5.1.

NOTE 4 – When channel number is used, this octet may be repeated to indicate multiple channels.

NOTE 5 – These octets shall be omitted when the entire interface is to be identified.

Figure 4-18/Q.931 – Channel identification information element

Table 4-13/Q.931 – Channel identification information element

Interface identifier present (octet 3)

Bit
7
0 Interface implicitly identified (Note 1)
1 Interface explicitly identified in one or more octets, beginning with octet 3.1.

NOTE 1 – The interface which includes the D-channel carrying this information element is indicated.

Interface type (octet 3)

Bit
6
0 Basic interface
1 Other interface, e.g. primary rate (Note 2)

NOTE 2 – The type of interface should be understood because the interface is identified by the "interface identifier present" field (octet 3, bit 7) and the interface identifier field (octet 3.1), if any.

Preferred/Exclusive (octet 3)

Bit
4
0 Indicated channel is preferred
1 Exclusive; only the indicated channel is acceptable

NOTE 3 – Preferred/exclusive has significance only for B-channel selection.

Table 4-13/Q.931 – Channel identification information element *(continued)*

D-channel indicator (octet 3)

Bit 3	
0	The channel identified is not the D-channel
1	The channel identified is the D-channel

NOTE 4 – D-channel indication has significance in D-channel use. No other information affects D-channel use.

Information channel selection (octet 3) (Note 5)

Bits 2 1	*Basic interface*	*Other interfaces*
0 0	No channel	No channel
0 1	B1 channel	As indicated in following octets
1 0	B2 channel	Reserved
1 1	Any channel (Note 6)	Any channel

NOTE 5 – The information channel selection does not apply to the D-channel.

NOTE 6 – This value shall be used on a basic access when both B-channels are to be identified, e.g. multirate (64 kbit/s base rate). This shall not be used for restart procedures.

Interface identifier (octet 3.1)

Binary code assigned to the interface at subscription time. At subscription time, the binary code for the interface will specify the number of octets to be used and the content of each octet.

NOTE 7 – When the 'information channel selection' field of octet 3 (bit 2-1) indicates 'any channel', if present the interface 'identifier' field is set to all '0'.

Coding standard (octet 3.2)

Bits 7 6	
0 0	ITU-T standardized coding, as described below
0 1	ISO/IEC Standard (Note 8)
1 0	National standard (Note 8)
1 1	Standard defined for the network (either public or private) present on the network side of the interface (Note 8)

NOTE 8 – These other coding standards should only be used when the desired call states cannot be represented by ITU-T-standardized coding.

Number/map (octet 3.2)

Bit 5	
0	Channel is indicated by the number in the following octet
1	Channel is indicated by the slot map (Map) in the following octet(s)

NOTE 9 – When the information transfer rate is 64 kbit/s, the channel number shall be used unless there exists a bilateral agreement between the user and the network to use the slot map.

NOTE 10 – Slot map shall be used when supporting the multirate (64 kbit/s base rate) bearer capability on a primary rate access.

Table 4-13/Q.931 – Channel identification information element *(concluded)*

Channel type/map element type (octet 3.2)

Bits

4 3 2 1

0 0 1 1	B-channel units (Note 11)
0 1 1 0	H0-channel units
1 0 0 0	H11-channel units
1 0 0 1	H12-channel units

All other values are reserved.

NOTE 11 – This value shall be used for multirate (64 kbit/s base rate) bearer capability.

Channel number (octet 3.3)

Binary number assigned to the channel. For B-channels, the channel number equals the time slot number. See Recommendation I.431 [27].

NOTE 12 – Either "Channel Number" or "Slot map" is used exclusively, depending on the "Number/Map" information.

Slot map (octet 3.3)

Bit position(s) in slot map corresponding to time slot(s) used by the channel is set to 1, see Figure 4-19. The remaining bits are set to 0.

NOTE 13 – The length of the slot map (in bits) is defined by the capacity of the interface type (e.g. 1534 kbit/s or 2048 kbit/s for a primary rate interface) divided by the capacity of the channel type/map-element type (e.g. 64 kbit/s for a B-channel). The length of the slot map is the smallest number of complete octets that contain the length in bits.

8	7	6	5	4	3	2	1	Octet
24	23	22	21	20	19	18	17	3.3.1
16	15	14	13	12	11	10	9	3.3.2
8	7	6	5	4	3	2	1	3.3.3

1544 kbit/s

8	7	6	5	4	3	2	1	Octet
31	30	29	28	27	26	25	24	3.3.1
23	22	21	20	19	18	17	16	3.3.2
15	14	13	12	11	10	9	8	3.3.3
7	6	5	4	3	2	1	0	3.3.4

2048 kbit/s

a) Primary rate interface, map element type = B-channel

8	7	6	5	4	3	2	1	Octet
				d(4)	c(3)	b(2)	a(1)	3.3

1544 kbit/s

8	7	6	5	4	3	2	1	Octet
			e(5)	d(4)	c(3)	b(2)	a(1)	3.3

2048 kbit/s

b) Primary rate interface, map element type = H_0 channel

NOTE 1 – See Annex A/I.431 [27], concerning meaning of a-e.

NOTE 2 – Number within ( ) indicates the associated H_0-channel number used when corresponding H_0-channel is represented by channel number in octet 3.3.

8	7	6	5	4	3	2	1	Octet
							H11(1)	3.3

1544 kbit/s

8	7	6	5	4	3	2	1	Octet
							H12(1)	3.3

2048 kbit/s

c) Primary rate interface, map element type = H_1-channel

NOTE 3 – Number within ( ) indicates the associated H_1-channel number used when corresponding H_1-channel is represented by channel number in octet 3.3.

NOTE 4 – For 2048 kbit/s interface, H_{11} slot will be indicated by the same format.

Figure 4-19/Q.931 – Slot map field

4.5.14 Congestion level

The purpose of the Congestion level information element is to describe the congestion status of the call. It is a single octet information element coded as shown in Figure 4-20 and Table 4-14.

8	7	6	5	4	3	2	1	Octet
	Congestion level information element identifier							
1	0	1	1	Congestion level				1

Figure 4-20/Q.931 – Congestion level information element

Table 4-14/Q.931 – Congestion level information element

Congestion level (octet 1)	
Bits	
4 3 2 1	
0 0 0 0	Receiver ready
1 1 1 1	Receiver not ready
All other values are reserved.	

4.5.15 Date/time

The purpose of the Date/time information element is to provide the date and time to the user. It indicates the point in time when the message has been generated by the network.

NOTE – It is a network dependent matter whether the time indicated is local time or Coordinated Universal Time (UTC) and which calendar is used for referencing the date.

The Date/time information element is coded as shown in Figure 4-21.

8	7	6	5	4	3	2	1	Octet
	Date/time information element identifier							
0	0	1	0	1	0	0	1	1
Length of date/time contents								2
Year								3
Month								4
Day								5
Hour								6*
Minute								7*
Second								8*

NOTE – Octets 3-8 are binary coded (bit 1 being the least significant bit).

Figure 4-21/Q.931 – Date/time information element

4.5.16 Display

The purpose of the Display information element is to supply display information that may be displayed by the user. The information contained in this element is coded in IA5 characters.

The display information element is coded as shown in Figure 4-22.

The Display information element has a network dependent default maximum length of 34 or 82 octets. The evolution to a single maximum value of 82 octets is an objective. If a user receives a Display information element with a length exceeding the maximum length which the user can handle, the information element should be truncated by the user.

8	7	6	5	4	3	2	1	Octet
	Display information element identifier							
0	0	1	0	1	0	0	0	1
Length of display contents								2
0	Display information (IA5 characters)							3
								etc.

Figure 4-22/Q.931 – Display information element

4.5.17 High layer compatibility

The purpose of the High layer compatibility information element is to provide a means which should be used by the remote user for compatibility checking. See Annex B.

The High layer compatibility information element is coded as shown in Figure 4-23 and Table 4-15.

The High layer compatibility information element can be repeated in the SETUP message to indicate dual high layer capabilities for selection. By default, if the High layer compatibility information element is repeated without the Repeat indicator information element, it shall be interpreted as increasing order of priority.

The maximum length of this information element is five octets.

NOTE – The High layer compatibility information element is transported transparently by an ISDN between a call originating entity, e.g. a calling user and the addressed entity, e.g. a remote user or a high layer function network node addressed by the call originating entity. However, if explicitly requested by the user (at subscription time), a network which provides some capabilities to realize teleservices may interpret this information to provide a particular service.

8	7	6	5	4	3	2	1	Octet
	High layer compatibility information element identifier							
0	1	1	1	1	1	0	1	1
Length of high layer compatibility contents								2
ext. 1	Coding standard		Interpretation			Presentation method of protocol profile		3
ext. 0/1	High layer characteristics identification							4
ext. 1	Extended high layer characteristics identification							4a* (Note 1)
ext. 1	Extended videotelephony characteristics identification							4a* (Note 2)

NOTE 1 – This octet may be present when octet 4 indicates Maintenance or Management.
NOTE 2 – This octet may be present when octet 4 indicates audio visual.

Figure 4-23/Q.931 – High layer compatibility information element

Table 4-15/Q.931 – High layer compatibility information element

Coding standard (octet 3)

Bits 7 6	
0 0	ITU-T standardized coding, as described below
0 1	ISO/IEC standard (Note 1)
1 0	National standard (Note 1)
1 1	Standard defined for the network (either public or private) present on the network side of the interface (Note 1)

NOTE 1 – These other coding standards should only be used only when the desired high layer compatibility cannot be represented by ITU-T standardized coding.

Interpretation (octet 3)

Bits 5 4 3	
1 0 0	First (primary or only) high layer characteristics identification (in octet 4) to be used in the call

All other values are reserved.

NOTE 2 – "Interpretation" indicates how the "High layer characteristics identification" (in octet 4) should be interpreted.

NOTE 3 – Currently, "Interpretation" has only a single value. However, "Interpretation", when enhanced, will be able to indicate how the "High layer characteristics identification" in the same information element shall be interpreted when multiple "High layer characteristics identifications" are used and exact relationship among them needs to be indicated (e.g. sequential usage, alternative list, simultaneous usage). Such enhancements in conjunction with the possible negotiation procedures are left for further study.

Presentation method of protocol profile (octet 3)

Bits 2 1	
0 1	High layer protocol profile (without specification of attributes)

All other values are reserved.

NOTE 4 – Currently, "Presentation method of protocol profile" has only a single value, i.e. a "profile value" is used to indicate a service to be supported by high layer protocols as required. Necessity of other presentation methods, e.g. service indications in the forum of layer-by-layer indication of protocols to be used in high layers, is left for further study.

High layer characteristics identification (octet 4)

Bits 7 6 5 4 3 2 1	
0 0 0 0 0 0 1	Telephony
0 0 0 0 1 0 0	Facsimile Group 2/3 (Recommendation F.182 [68])
0 1 0 0 0 0 1	Facsimile Group 4 Class I (Recommendation F.184 [69])
0 1 0 0 1 0 0	Facsimile service Group 4, Classes II ad III (Recommendation F.184)
0 1 0 1 0 0 0	(Note 7)
0 1 1 0 0 0 1	(Note 7)
0 1 1 0 0 1 0	Syntax based Videotex (Recommendation F.300 [73] and T.102 [74])
0 1 1 0 0 1 1	International Videotex interworking via gateways or interworking units (Recommendation F.300 and T.101 [75])
0 1 1 0 1 0 1	Telex service (Recommendation F.60 [76])
0 1 1 1 0 0 0	Message Handling Systems (MHS) (X.400-series Recommendation [77])

Table 4-15/Q.931 – High layer compatibility information element *(continued)*

1 0 0 0 0 0 1	OSI application (Note 6) (X.200-series Recommendations [78])
1 0 0 0 0 1 0	FTAM application (ISO 8571)
1 0 1 1 1 1 0	Reserved for maintenance (Note 8)
1 0 1 1 1 1 1	Reserved for management (Note 8)
1 1 0 0 0 0 0	Videotelephony (Recommendations F.720 [91] and F.721 [79]) and F.731 profile 1a) (Note 9)
1 1 0 0 0 0 1	Videoconferencing Recommendation F.702 [94] and F.731 [97] Profile 1b (Note 9)
1 1 0 0 0 1 0	Audiographic conferencing Recommendations F.702 [94] and F.731 [97] (including at least profile 2a2 and optionally 2a1, 2a3, 2b1, 2b2, and 2bc) (Notes 9 and 10)
1 1 0 0 0 1 1 through 1 1 0 0 1 1 1	Reserved for audiovisual service (F.700-series Recommendations [80])
1 1 0 1 0 0 0	Multimedia services F.700-series Recommendations [80] (Note 9)
1 1 0 0 0 1 1 through 1 1 0 1 1 1 1	Reserved for audiovisual services (F.700-series Recommendations [80])
1 1 1 1 1 1 1	Reserved
All other values are reserved	

Extended high layer characteristics identification (octet 4a for maintenance or management)

Bits 7 6 5 4 3 2 1	
0 0 0 0 0 0 1	Telephony
0 0 0 0 1 0 0	Facsimile Group 2/3 (Recommendation F.182)
0 1 0 0 0 0 1	Facsimile Group 4 Class I (Recommendation F.184)
0 1 0 0 1 0 0	Facsimile service Group 4, Classes II and III (Recommendation F.184)
0 1 0 1 0 0 0	(Note 11)
0 1 1 0 0 0 1	(Note 11)
0 1 1 0 0 1 0	Syntax based Videotex (Recommendations F.300 and T.102)
0 1 1 0 0 1 1	International Videotex interworking via gateways or interworking units (Recommendations F.300 and T.101)
0 1 1 0 1 0 1	Telex service (Recommendation F.60)
0 1 1 1 0 0 0	Message Handling Systems (MHS) (X.400-series Recommendations [77])
1 0 0 0 0 0 1	OSI application (Note 6) (X.200-series Recommendations [78])
1 0 0 0 0 1 0	FTAM application (ISO 8571)
1 0 1 1 1 1 0	Not available for assignment
1 0 1 1 1 1 1	Not available for assignment
1 1 0 0 0 0 0	Videotelephony (Recommendations F.720 [91] and F.721 [79] and F.731, profile 1a)
1 1 0 0 0 0 1	Videoconferencing Recommendations F.702 [94] and F.731 [97], profile 1b
1 1 0 0 0 1 0	Audiographic conferencing Recommendations F.702 [94] and F.731 [97] (including at least profile 2a2 and optionally 2a1, 2a3, 2b1, 2b2 and 2bc)
1 1 0 0 0 1 1 through 1 1 0 0 1 1 1	Reserved for audiovisual services (F.700-series Recommendations [80])
1 1 0 1 0 0 0	Multimedia services (F.700-series Recommendations [80])

Table 4-15/Q.931 – High layer compatibility information element *(concluded)*

1 1 0 1 0 0 1 through 1 1 0 1 1 1 1	Reserved for audiovisual services (F.700-series Recommendations [80])
1 1 1 1 1 1 1	Reserved
All other values are reserved	
Extended audiovisual characteristics identification (octet 4a for videotelephony)	
Bits 7 6 5 4 3 2 1	
0 0 0 0 0 0 1	Capability set of initial channel of H.221
0 0 0 0 0 1 0	Capability set of subsequent channel of H.221
0 1 0 0 0 0 1	Capability set of initial channel associated with an active 3.1 kHz audio or speech call.
NOTE 5 – The coding above applies in case of "Coding standard" = "ITU-T Standard" and "Presentation method of protocol profile" = "High layer protocol profile".	
NOTE 6 – Further compatibility checking will be executed by the OSI high layer protocol.	
NOTE 7 – Codepoints are added only to those services for which ITU-T Recommendations are available. See also the I.241-series of Recommendations [34].	
NOTE 8 – When this coding is included, octet 4 may be followed by octet 4a.	
NOTE 9 – When this coding is used, octet 4 may be followed by octet 4a.	
NOTE 10 – The multimedia services identified by this codepoint mush have a mandatory common core functionality of speech that will ensure a minimum capability to communicate.	
NOTE 11 – This codepoint was previously allocated for an F.200-series Recommendation that has been deleted.	

4.5.18 Keypad facility

The purpose of the Keypad facility information element is to convey IA5 characters, e.g. entered by means of a terminal keypad.

The Keypad facility information element is coded as shown in Figure 4-24.

The maximum length of this information element is 34 octets.

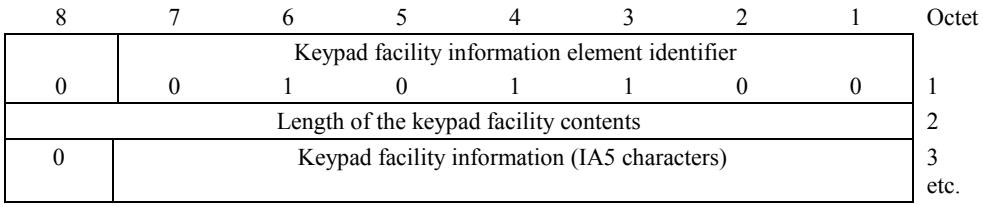


Figure 4-24/Q.931 – Keypad facility information element

4.5.19 Low layer compatibility

The purpose of the Low layer compatibility information element is to provide a means which should be used for capability checking by an addressed entity (e.g. a remote user or an interworking unit or a high layer function network node addressed by the calling user). The Low layer compatibility information element is transferred transparently by an ISDN between the call originating entity (e.g. the calling user) and the addressed entity. See Annexes B and I.

If low layer compatibility negotiation is allowed by the network (see Annex J), the Low layer compatibility information element is also passed transparently from the addressed entity to originating entity.

The Low layer compatibility information element is coded as shown in Figure 4-25 and Table 4-16. The maximum length of this information element is 18 octets.

NOTE – Some networks conforming to Recommendation Q.931 (1988) may support a maximum information element length of only 16 octets.

8	7	6	5	4	3	2	1	Octet
	Low layer compatibility information element identifier							
0	1	1	1	1	1	0	0	1
Length of the low layer compatibility contents								2
ext. 0/1	Coding standard		Information transfer capability					3
ext. 1	Negot. indic.	Spare 0	0	0	0	0	0	3a*
ext. 1	Transfer mode		Information transfer rate					4
ext. 1	Rate multiplier							4.1* (Note 1)
ext. 0/1	Layer 1 ident. 0	1	User information layer 1 protocol					5*
ext. 0/1	Synch./ asynch.	Negot.	User rate					5a* (Note 2)
ext. 0/ 1	Intermediate rate		NIC on Tx	NIC on Rx	Flow control on Tx	Flow control on Rx	Spare 0	5b* (Note 3)
ext. 0/1	Hdr/no Hdr	Multifra me	Mode	Negot. LLI	Assignor/ Assignor ee	In-band negot.	Spare 0	5b* (Note 4)
ext. 0/1	Number of stop bits		Number of data bits		Parity			5c* (Note 2)
ext. 1	Duplex mode	Modem type						5d* (Note 2)
ext. 0/1	layer 2 ident. 1	0	User information layer 2 protocol					6*
ext. 0/1	Mode		Spare 0	0	0	Q.933 use		6a* (Note 5)
ext. 1	User specified layer 2 protocol information							6a* (Note 6)
ext. 1	Window size (k)							6b* (Note 5)
ext. 0/1	layer 3 ident. 1	1	User information layer 3 protocol					7
ext. 1	Optional layer 3 protocol information							7a* (Note 8)
ext. 0/1	Mode		Spare 0	0	0	0	0	7a* (Note 7)
ext. 0/1	Spare 0	0	0	Default packet size				7b* (Note 7)
ext. 1	Packet window size							7c* (Note 7)
ext. 0	Spare 0	0	0	Additional layer 3 protocol information (most significant bits)				7a* (Note 9)
ext. 1	Spare 0	0	0	Additional layer 3 protocol information (least significant bits)				7b* (Note 9)

Figure 4-25/Q.931 – Low layer compatibility information element

NOTES to Figure 4-25

NOTE 1 – This octet is required if octet 4 indicates multirate (64 kbit/s base rate). Otherwise, it shall not be present.

NOTE 2 – This octet may be present if octet 3 indicates *unrestricted digital information* and octet 5 indicates either of the ITU-T standardized rate adaptions V.110, I.460 and X.30 or V.120 [9]. It may also be present if octet 3 indicates 3.1 kHz audio and octet 5 indicates G.711.

NOTE 3 – This structure of octet 5b only applies if octet 5 indicates ITU-T standardized rate adaption (V.110 [7], I.460 [15] and X.30 [8]).

NOTE 4 – This structure of octet 5b only applies if octet 5 indicates ITU-T standardized rate adaption V.120 [9].

NOTE 5 – This octet may be present only if octet 6 indicates certain acknowledged mode HDLC elements of procedure as indicated in Table 4-16.

NOTE 6 – This octet may be present only if octet 6 indicates user specified layer 2 protocol.

NOTE 7 – This octet may be present only if octet 7 indicates a layer 3 protocol based on ITU-T X.25 [5], ISO/IEC 8208 [41] or ITU-T Rec. X.223 [96] and ISO/IEC 8878 [81] as indicated in Table 4-16.

NOTE 8 – This octet may be present only if octet 7 indicates user specified layer 3 protocol.

NOTE 9 – This may be included if octet 7 indicates ISO/IEC TR 9577.

Table 4-16/Q.931 – Low layer compatibility information element

Coding standard (octet 3)

Bits 7 6	
0 0	ITU-T standardized coding, as described below
0 1	ISO/IEC Standard (Note 1)
1 0	National standard (Note 1)
1 1	Standard defined for the network (either public or private) present on the network side of the interface (Note 1)

NOTE 1 – These other coding standards should only be used only when the desired low layer compatibility cannot be represented by ITU-T-standardized coding.

Information transfer capability (octet 3)

Bits 5 4 3 2 1	
0 0 0 0 0	Speech
0 1 0 0 0	Unrestricted digital information
0 1 0 0 1	Restricted digital information
1 0 0 0 0	3.1 kHz audio
1 0 0 0 1	Unrestricted digital information with tones/announcements (Note 2)
1 1 0 0 0	Video

All other values are reserved.

NOTE 2 – Unrestricted digital information with tones/announcements (UDI-TA) is the new information transfer attribute value that had previously been named "7 kHz audio" in Recommendation Q.931 (1988).

Negotiation indicator (octet 3a)

Bit 7	
0	Out-band negotiation not possible
1	Out-band negotiation possible

NOTE 3 – See Annex J for description of low layer compatibility negotiation.

NOTE 4 – When octet 3a is omitted, "out-band negotiation not possible" shall be assumed.

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

Transfer mode (octet 4)

Bits

7 6

0 0 Circuit mode

1 0 Packet-mode

All other values are reserved.

Information transfer rate (octet 4)

Bits

5 4 3 2 1	Circuit mode	Packet-mode
0 0 0 0 0	–	This code shall be used for all packet calls
1 0 0 0 0	64 kbit/s	–
1 0 0 0 1	2 × 64 kbit/s	–
1 0 0 1 1	384 kbit/s	–
1 0 1 0 1	1536 kbit/s	–
1 0 1 1 1	1920 kbit/s	–
1 1 0 0 0	Multirate (64 kbit/s base rate)	

All other values are reserved.

NOTE 5 – When the information transfer rate 2 × 64 kbit/s is used, the coding of octets 3 and 4 refer to both 64 kbit/s channels.

NOTE 6 – Additional attributes are defined in Table 4-17.

Rate multiplier (octet 4.1)

Coded as a binary representation of the multiplier to the base rate. The multiplier can take any value from 2 up to the maximum number of B-channels available on the interface.

User information layer 1 protocol (octet 5)

Bits

5 4 3 2 1	
0 0 0 0 1	ITU-T standardized rate adaption V.110 [7], I.460 [15] and X.30 [8]. This implies the presence of octet 5a and optionally octets 5b, 5c and 5d as defined below.
0 0 0 1 0	Recommendation G.711 [10] μ-law.
0 0 0 1 1	Recommendation G.711 A-law.
0 0 1 0 0	Recommendation G.721 [11] 32 kbit/s ADPCM and Recommendation I.460 [15].
0 0 1 0 1	Recommendations H.221 and H.242.
0 0 1 1 0	Recommendations H.223 [92] and H.245 [93]
0 0 1 1 1	Non-ITU-T standardized rate adaption. This implies the presence of octet 5a and, optionally, octets 5b, 5c and 5d. The use of this codepoint indicates that the user rate specified in octet 5a is defined by the user. Additionally, octets 5b, 5c and 5d, if present, are defined in accordance with the user specified rate adaption.
0 1 0 0 0	ITU-T standardized rate adaption V.120 [9]. This implies the presence of octets 5a and 5b as defined below, and optionally octets 5c and 5d.
0 1 0 0 1	ITU-T standardized rate adaption X.31 [14] HDLC flag stuffing.

All other values are reserved.

NOTE 7 – If the transfer mode is "circuit mode" and if the information transfer capability is "unrestricted digital information" or "Restricted digital information", and if a specific user information layer 1 protocol is to be identified to the addressed entity, octet 5 shall be present. If the transfer mode is packet mode, octet 5 may be omitted.

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

Synchronous/Asynchronous (octet 5a)

Bit 7	
0	Synchronous data
1	Asynchronous data

NOTE 8 – Octets 5b-5d may be omitted in the case of synchronous user rates.

Negotiation (octet 5a)

Bit 6	
0	In-band negotiation not possible
1	In-band negotiation possible

NOTE 9 – See Recommendations V.110 [7], I.460 [15] and X.30 [8] or modem type Recommendations.

User rate (octet 5a)

Bits 5 4 3 2 1	
0 0 0 0 0	For I.460, rate is specified by bits 7, 6 of octet 5b, Intermediate rate. For V.110 and X.30, rate is indicated by E-bits (synchronous data only) or may be negotiated in-band. For V.120, rate is unspecified or may be negotiated in-band.
0 0 0 0 1	0.6 kbit/s Recommendation X.1 [17]
0 0 0 1 0	1.2 kbit/s
0 0 0 1 1	2.4 kbit/s Recommendation X.1
0 0 1 0 0	3.6 kbit/s
0 0 1 0 1	4.8 kbit/s Recommendation X.1
0 0 1 1 0	7.2 kbit/s
0 0 1 1 1	8 kbit/s Recommendation I.460
0 1 0 0 0	9.6 kbit/s Recommendation X.1
0 1 0 0 1	14.4 kbit/s
0 1 0 1 0	16 kbit/s Recommendation I.460
0 1 0 1 1	19.2 kbit/s
0 1 1 0 0	32 kbit/s Recommendation I.460
0 1 1 0 1	38.4 kbit/s Recommendation V.110 [7]
0 1 1 1 0	48 kbit/s Recommendation X.1
0 1 1 1 1	56 kbit/s
1 0 0 0 0	64 kbit/s Recommendation X.1
1 0 0 1 0	57.6 kbit/s Recommendation V.14 [88] extended
1 0 0 1 1	28.8 kbit/s Recommendation V.110 [89]
1 0 1 0 0	24 kbit/s Recommendation V.110 [89]
1 0 1 0 1	0.1345 kbit/s Recommendation X.1
1 0 1 1 0	0.100 kbit/s Recommendation X.1
1 0 1 1 1	0.075/1.2 kbit/s Recommendation X.1 (Note 10)
1 1 0 0 0	1.2/0.075 kbit/s Recommendation X.1 (Note 10)

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

1 1 0 0 1	0.050 kbit/s Recommendation X.1
1 1 0 1 0	0.075 kbit/s Recommendation X.1
1 1 0 1 1	0.110 kbit/s Recommendation X.1
1 1 1 0 0	0.150 kbit/s Recommendation X.1
1 1 1 0 1	0.200 kbit/s Recommendation X.1
1 1 1 1 0	0.300 kbit/s Recommendation X.1
1 1 1 1 1	12 kbit/s

All other values are reserved.

NOTE 10 – The first rate is the transmit rate in the forward direction of the call. The second rate is the transmit rate in the backward direction of the call.

Octet 5b for V.110 [7], I.460 [15] and X.30 [8] rate adaption

Intermediate rate (octet 5b)

Bits 7 6	
0 0	Not used
0 1	8 kbit/s
1 0	16 kbit/s
1 1	32 kbit/s

Network Independent Clock (NIC) on transmission (Tx) (octet 5b) (Note 11)

Bit 5	
0	Not required to send data with network independent clock
1	Required to send data with network independent clock

NOTE 11 – Refers to transmission in the forward direction of the call.

NOTE 12 – See Recommendations V.110, I.460 [15] and X.30.

Network Independent Clock (NIC) on reception (Rx) (octet 5b) (Note 13)

Bit 4	
0	Cannot accept data with Network Independent Clock (i.e. sender does not support this optional procedure)
1	Can accept data with Network Independent Clock (i.e. sender does support this optional procedure)

NOTE 13 – Refers to transmission in the backward direction of the call.

NOTE 14 – See Recommendations V.110 [7], I.460 [15] and X.30 [8].

Flow control on transmission (Tx) (octet 5b) (Note 15)

Bit 3	
0	Not required to send data with flow control mechanism
1	Required to send data with flow control mechanism

NOTE 15 – Refers to transmission in the forward direction of the call.

NOTE 16 – See Recommendations V.110, I.460 and X.30.

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

Flow control on reception (Rx) (octet 5b) (Note 17)

Bit
2
0 Cannot accept data with flow control mechanism (i.e. sender does not support this optional procedure)
1 Can accept data with flow control mechanism (i.e. sender does support this optional procedure)

NOTE 17 – Refers to transmission in the backward direction of the call.

NOTE 18 – See Recommendations V.110, I.460 and X.30.

Octet 5b for V.120 [9] Rate adaption

Rate adaption header/no header (octet 5b)

Bit
7
0 Rate adaption header not included
1 Rate adaption header included

Multiple frame establishment support in data link (octet 5b)

Bit
6
0 Multiple frame establishment not supported. Only UI frames allowed
1 Multiple frame establishment supported

Mode of operation (octet 5b)

Bit
5
0 Bit transparent mode of operation
1 Protocol sensitive mode of operation

Logical link identifier negotiation (octet 5b)

Bit
4
0 Default, LLI = 256 only
1 Full protocol negotiation (Note 19)

NOTE 19 – A connection over which protocol negotiation will be executed is indicated in bit 2 of octet 5b.

Assignor/assignee (octet 5b)

Bit
3
0 Message originator is "default assignee"
1 Message originator is "assignor only"

In-band/out-band negotiation (octet 5b)

Bit
2
0 Negotiation is done with USER INFORMATION messages on a temporary signalling connection
1 Negotiation is done in-band using logical link zero

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

Number of stop bits (octet 5c)

Bits 7 6	
0 0	Not used
0 1	1 bit
1 0	1.5 bits
1 1	2 bits

Number of data bits excluding parity bit if present (octet 5c)

Bits 5 4	
0 0	Not used
0 1	5 bits
1 0	7 bits
1 1	8 bits

Parity information (octet 5c)

Bits 3 2 1	
0 0 0	Odd
0 1 0	Even
0 1 1	None
1 0 0	Forced to 0
1 0 1	Forced to 1

All other values are reserved.

Duplex mode (octet 5d)

Bit 7	
0	Half duplex
1	Full duplex

Modem type (octet 5d)

Bits 6 5 4 3 2 1	
0 0 0 0 0 0 through 0 0 0 1 0 1	National use
0 1 0 0 0 1	Recommendation V.21
0 1 0 0 1 0	Recommendation V.22
0 1 0 0 1 1	Recommendation V.22 *bis*
0 1 0 1 0 0	Recommendation V.23
0 1 0 1 0 1	Recommendation V.26
0 1 0 1 1 0	Recommendation V.26 *bis*
0 1 0 1 1 1	Recommendation V.26 *ter*
0 1 1 0 0 0	Recommendation V.27
0 1 1 0 0 1	Recommendation V.27 *bis*
0 1 1 0 1 0	Recommendation V.27 *ter*
0 1 1 0 1 1	Recommendation V.29
0 1 1 1 0 0	Recommendation V.32

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

0 1 1 1 1 0	Recommendation V.34 [90]
1 0 0 0 0 0 through 1 0 1 1 1 1	National use
1 1 0 0 0 0 through 1 1 1 1 1 1	User specified

All other values reserved.

User information layer 2 protocol (octet 6)

Bits 5 4 3 2 1	
0 0 0 0 1	Basic mode ISO 1745 [36]
0 0 0 1 0	Recommendation Q.921/I.441 [3] (Note 23)
0 0 1 1 0	Recommendation X.25 [5], link layer (Notes 20, 23)
0 0 1 1 1	Recommendation X.25 Multilink (Note 23)
0 1 0 0 0	Extended LAPB; for half duplex operation (Recommendation T.71 [37])
0 1 0 0 1	HDLC ARM (ISO/IEC 4335 [38]) (Note 23)
0 1 0 1 0	HDLC NRM (ISO/IEC 4335) (Note 23)
0 1 0 1 1	HDLC ABM (ISO/IEC 4335) (Note 23)
0 1 1 0 0	LAN logical link control (ISO/IEC 8802-2 [39])
0 1 1 0 1	Recommendation X.75 [40]. Single Link Procedure (SLP) (Note 23)
0 1 1 1 0	Recommendation Q.922 (Note 23)
0 1 1 1 1	Core aspects of Recommendation Q.922
1 0 0 0 0	User specified (Note 21)
1 0 0 0 1	ISO/IEC 7776 DTE-DCE operation (Notes 22, 23)

All other values are reserved.

NOTE 20 – This Recommendation is compatible with ISO/IEC 7776 DTE-DCE operation.

NOTE 21 – When this coding is included, octet 6a will include user coding for the user specified Layer 2 protocol.

NOTE 22 – This Standard is compatible with Recommendation X.75 modified by the application rules defined in Recommendation T.90.

NOTE 23 – When this coding is included, octets 6a and 6b with ITU-T encoding may be included.

Octet 6a for ITU-T codings

Mode of operation (octet 6a)

Bits 7 6	
0 1	Normal mode of operation
1 0	Extended mode of operation

All other values are reserved.

Q.933 use (octet 6a)

Bits 2 1	
0 0	For use when the coding defined in Recommendation Q.933 is not used

All other values are reserved.

Octet 6a for user protocol

User specified layer 2 protocol information (octet 6a)

The use and coding of octet 6a is according to user defined requirements.

Window size (k) (octet 6b)

Bits 7-1 binary coding of *k* parameter value in the range from 1 to 127.

Table 4-16/Q.931 – Low layer compatibility information element *(continued)*

User information layer 3 protocol (octet 7)

Bits 5 4 3 2 1	
0 0 0 1 0	Recommendation Q.931/I.451
0 0 1 1 0	Recommendation X.25, packet layer (Note 25)
0 0 1 1 1	ISO/IEC 8208 [41] (X.25 packet level protocol for data terminal equipment) (Note 25)
0 1 0 0 0	ITU-T Rec. X.223 and ISO/IEC 8878 [81] (use of ISO/IEC 8208 [41] and Recommendation X.25 to provide the OSI-CONS) (Note 25)
0 1 0 0 1	ISO/IEC 8473 [43] (OSI connectionless mode protocol)
0 1 0 1 0	Recommendation T.70 [32] minimum network layer
0 1 0 1 1	ISO/IEC TR 9577 [82] (Protocol identification in the network layer) (Note 26)
1 0 0 0 0	User specified (Note 24)

All other values are reserved.

NOTE 24 – When this coding is included, octet 7a will include user coding for the user specified layer 3 protocol.

NOTE 25 – When this coding is included, octets 7a, 7b and 7c with ITU-T X.25, X.223 and ISO/IEC TR 9577 encoding may be included.

NOTE 26 – When this coding is included, octets 7a and 7b with ITU-T X.25, X.223 [96] and ISO/IEC TR 9577 encoding may be included.

Octet 7a, 7b and 7c X.25 packet layer, ISO/IEC 8208 and X.223 codings

Mode of operation (octet 7a)

Bits 7 6	
0 1	Normal packet sequence numbering
1 0	Extended packet sequence numbering

All other values are reserved.

Default packet size (octet 7b)

Bits 4 3 2 1	
0 1 0 0	Default packet size 16 octets
0 1 0 1	Default packet size 32 octets
0 1 1 0	Default packet size 64 octets
0 1 1 1	Default packet size 128 octets
1 0 0 0	Default packet size 256 octets
1 0 0 1	Default packet size 512 octets
1 0 1 0	Default packet size 1024 octets
1 0 1 1	Default packet size 2048 octets
1 1 0 0	Default packet size 4096 octets

All other values are reserved.

Packet window size (octet 7c)

Bits 7-1 binary coding of packet window size value in the range from 1 to 127.

Octet 7a for user protocol

User specified layer 3 protocol information (octet 7a)

The use and coding of octet 7a depends on user defined requirements

Octets 7a and 7b for ISO/IEC TR 9577 coding (Notes 26 and 27)

Bit 8 (ext.) set to 0 in octet 7a and set to 1 in octet 7b.

Bits 7 to 5 are spare (set to 0) in both octets.

Table 4-16/Q.931 – Low layer compatibility information element *(concluded)*

7a Bits 4 3 2 1	*7b* Bits 4 3 2 1	
1 1 0 0	1 1 0 0	Internet Protocol (RFC 791) (Annex C of ISO/IEC TR 9577 [82])
1 1 0 0	1 1 1 1	Point-to-Point Protocol (RFC 1548)

All other values are reserved.

NOTE 27 – If the user information layer 3 protocol indicates "Network layer protocol identification", it may be included to identify the actual user information layer 3 protocol to the addressed entity (see Annex I). Any Network layer Protocol Identifier code defined in ISO/IEC TR 9577 [82] may be included. Octet 7c shall not be included.

Table 4-17/Q.931 – Low layer compatibility attributes

LLC attributes		**Additional attributes**			
Transfer mode	**Information transfer capability**	**Structure**	**Configuration**	**Establishment**	**Symmetry**
Circuit	Speech	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Unrestricted data	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Restricted data	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	3.1 kHz audio	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Unrestricted data with tones/announcements	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Circuit	Video	8 kHz integrity	Point-to-point	Demand	Bi-directional symmetric
Packet	Unrestricted data	Service data unit integrity	Point-to-point	Demand	Bi-directional symmetric

NOTE 1 – When the information transfer rate 2 × 64 kbit/s is used, 8 kHz integrity with Restricted Differential Time Delay (RDTD) is offered.

NOTE 2 – When multirate (64 kbit/s base rate) is indicated as the information transfer rate, Time Slot Sequence integrity shall be provided.

4.5.20 More data

The More data information element is sent by the user to the network in a USER INFORMATION message, and delivered by the network to the destination user(s) in the corresponding USER INFORMATION message. The presence of the More data information element indicates to the destination user that another USER INFORMATION message will follow, containing information belonging to the same block.

The use of the More data information element is not supervised by the network.

The More data information element is coded as shown in Figure 4-26.

The length of this information element is one octet.

8	7	6	5	4	3	2	1	Octet
	More data information element identifier							
1	0	1	0	0	0	0	0	1

Figure 4-26/Q.931 – More data information element

4.5.21 Network-specific facilities

The purpose of the Network-specific facilities information element is to indicate which network facilities are to be invoked. The Network-specific facilities information element is coded as shown in Figure 4-27 and Table 4-18. No more than four Network-specific facilities information elements may be included in a single message.

The maximum length of this information element is network dependent.

8	7	6	5	4	3	2	1	Octet
	Network-specific facilities information element identifier							
0	0	1	0	0	0	0	0	1
Length of network-specific facilities contents								2
Length of network identification								3
ext. 1	Type of network identification			Network identification plan				3.1*
Spare 0	Network identification (IA5 characters)							3.2*
Network-specific facility specification								4

NOTE 1 – Octets 3.1 and 3.2 are only present when the length in octet 3 is non-zero.

NOTE 2 – Octet 3.2 may be repeated as appropriate.

Figure 4-27/Q.931 – Network-specific facilities information element

Table 4-18/Q.931 – Network-specific facilities information element

Length of network identification (octet 3)

This field contains the length, in octets, of the network identification found in octet 3.1 and the repetition of octet 3.2. If the value is "0000 0000", then the default provider (see E.1) is assumed and octets 3.1 and 3.2 are omitted.

Type of network identification (octet 3.1)

Bits 7 6 5	
0 0 0	User specified
0 1 0	National network identification (Note 1)
0 1 1	International network identification

All other values are reserved.

NOTE 1 – In the case that "type of network identification" is coded as 010, "national network identification", "national identification plan" is coded according to national specification.

Network identification plan (octet 3.1)

Bits 4 3 2 1	
0 0 0 0	Unknown
0 0 0 1	Carrier Identification Code (Note 2)
0 0 1 1	Data network identification code (Recommendation X.121 [21])

All other values are reserved.

NOTE 2 – Carrier Identification Codes may be an appropriate method of identifying the network serving the remote user.

Network identification (octets 3.2, etc.)

These IA5 characters are organized according to the network identification plan specified in octet 3.1.

Network-specific facilities (octets 4, etc.)

This field is encoded according to the rules specified by the identified network.

4.5.22 Notification indicator

The purpose of the Notification indicator information element is to indicate information pertaining to a call.

The Notification indicator information element is coded as shown in Figure 4-28 and Table 4-19.

The maximum length of this information element is three octets.

8	7	6	5	4	3	2	1	Octet
	Notification indicator information element identifier							
0	0	1	0	0	1	1	1	1
Length of notification indicator contents								2
ext. 1	Notification description							3

Figure 4-28/Q.931 – Notification indicator information element

Table 4-19/Q.931 – Notification indicator information element

Notification description (octet 3)	
Bits	
7 6 5 4 3 2 1	
0 0 0 0 0 0 0	User suspended
0 0 0 0 0 0 1	User resumed
0 0 0 0 0 1 0	Bearer service change
All other values are reserved.	

4.5.23 Progress indicator

The purpose of the Progress indicator information element is to describe an event which has occurred during the life of a call. The information element may occur two times in a message.

The Progress indicator information element is coded as shown in Figure 4-29 and Table 4-20.

The maximum length of this information element is four octets.

8	7	6	5	4	3	2	1	Octet
Progress indicator information element identifier								
0	0	0	1	1	1	1	0	1
Length of progress indicator contents								2
ext. 1	Coding standard		Spare 0	Location				3
ext. 1	Progress description							4

Figure 4-29/Q.931 – Progress indicator information element

Table 4-20/Q.931 – Progress indicator information element

Coding standard (octet 3)

Bits 7 6	
0 0	ITU-T standardized coding, as described below
0 1	ISO/IEC Standard (Note 1)
1 0	National standard (Note 1)
1 1	Standard specific to identified location (Note 1)

NOTE 1 – These other coding standards should be used only when the desired progress indication can not be represented with the ITU-T-standardized coding.

Location (octet 3)

Bits 4 3 2 1	
0 0 0 0	User
0 0 0 1	Private network serving the local user
0 0 1 0	Public network serving the local user
0 0 1 1	Transit network (Note 2)
0 1 0 0	Public network serving the remote user
0 1 0 1	Private network serving the remote user
1 0 1 0	Network beyond the interworking point

All other values are reserved.

NOTE 2 – This value may be generated by some networks.

NOTE 3 – Depending on the location of the users, the local public network and remote public network may be the same network.

Progress description (octet 4)

Bits 7 6 5 4 3 2 1	No.	
0 0 0 0 0 0 1	1.	Call is not end-to-end ISDN; further call progress information may be available in-band
0 0 0 0 0 1 0	2.	Destination address is non-ISDN
0 0 0 0 0 1 1	3.	Origination address is non-ISDN
0 0 0 0 1 0 0	4.	Call has returned to the ISDN
0 0 0 0 1 0 1	5.	Interworking has occurred and has resulted in a telecommunication service change (Note 5)
0 0 0 1 0 0 0	8.	In-band information or an appropriate pattern is now available

All other values are reserved.

NOTE 4 – The use of different progress descriptions is further explained in Annex G.

NOTE 5 – This progress description value shall be used only in the case of interworking in a full ISDN environment, e.g. when bearer capability selection is not supported or when resource or route of the preferred capability is not available. In case of interworking with a non-ISDN environment, a progress description No. 1 shall be used. If the destination address is non-ISDN, the progress description No. 2 shall be used.

4.5.24 Repeat indicator

The purpose of the Repeat indicator information element is to indicate how repeated information elements shall be interpreted, when included in a message. The Repeat indicator information element is included before the first occurrence of the information element which will be repeated in a message. The Repeat indication information element is coded as shown in Figure 4-30 and Table 4-21.

The length of this information element is one octet.

NOTE – Use of the Repeat indicator information element in conjunction with an information element that occurs only once in a message shall not of itself constitute an error.

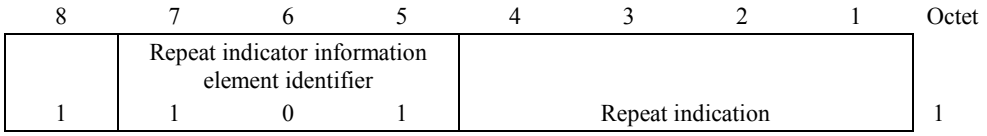


Figure 4-30/Q.931 – Repeat indicator information element

Table 4-21/Q.931 – Repeat indicator information element

Repeat indication (octet 1)

Bits
4 3 2 1
0 0 1 0 Prioritized list for selecting one possibility (Note)
All other values are reserved.

NOTE – Used for Bearer service change procedures (see Annex L).

4.5.25 Restart indicator

The purpose of the Restart indicator information element is to identify the class of the facility (i.e. channel or interface) to be restarted.

The Restart indicator information element is coded as shown in Figure 4-31 and Table 4-22.

The maximum length of this information element is three octets.

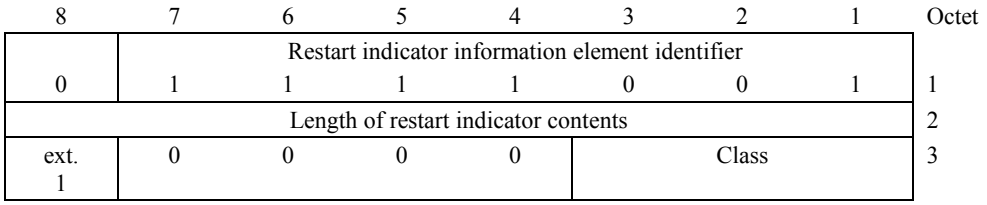


Figure 4-31/Q.931 – Restart indicator information element

Table 4-22/Q.931 – Restart indicator information element

Class (octet 3)

Bits
3 2 1
0 0 0 Indicated channels (Note 1)
1 1 0 Single interface (Note 2)
1 1 1 All interfaces (Note 3)

All other values are reserved.

NOTE 1 – The channel identification information element must be included and indicates which channels are to be restarted.

NOTE 2 – If non-associated signalling is used, the channel identification information element must be included to indicate the interface to be restarted if it is other than the one on which the D-channel is present.

NOTE 3 – May be used when there are two or more interfaces controlled by the D-channel. The channel identification information element must not be included with this coding.

4.5.26 Segmented message

The purpose of the Segmented message information element is to indicate that the transmission in which it appears is part of a segmented message, in addition to the use of message type SEGMENT. When included in a message segment, it appears directly after the Message type information element (see Annex H).

The Segmented message information element is coded as shown in Figure 4-32 and Table 4-23.

The length of this information element is four octets.

8	7	6	5	4	3	2	1	Octet
	Segmented message information element identifier							
0	0	0	0	0	0	0	0	1
Length of segmented message contents								2
First segment indicator	Number of segments remaining							3
0	Segmented message type							4

Figure 4-32/Q.931 – Segmented message information element

Table 4-23/Q.931 – Segmented message information element

First segment indicator (octet 3)

Bit
8
0 Subsequent segment to first segment
1 First segment of segmented message

Number of segments remaining (octet 3)

Binary number indicating the number of remaining segments within the message to be sent.

Segmented message type (octet 4)

Type of message being segmented coded as per 4.4.

NOTE – Bit 8 is reserved for possible future use as an extension bit.

4.5.27 Sending complete

The purpose of the Sending complete information element is to optionally indicate completion of called party number, see 5.1.3, 5.2.1 and 5.2.4.

It is a single octet information element coded as shown in Figure 4-33.

8	7	6	5	4	3	2	1	Octet
Sending complete information element identifier								
1	0	1	0	0	0	0	1	1

Figure 4-33/Q.931 – Sending complete information element

4.5.28 Signal

The purpose of the Signal information element is to allow the network to optionally convey information to a user regarding tones and alerting signals. (See clause 7).

The Signal information element is coded as shown in Figure 4-34 and Table 4-24.

The length of this information element is three octets.

The Signal information element may be repeated in a message.

8	7	6	5	4	3	2	1	Octet
Signal information element identifier								
0	0	1	1	0	1	0	0	1
Length of signal contents								
0	0	0	0	0	0	0	1	2
Signal value								3

Figure 4-34/Q.931 – Signal information element

Table 4-24/Q.931 – Signal information element

Signal value (octet 3)

Bits 8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0	Dial tone on
0 0 0 0 0 0 0 1	Ring back tone on
0 0 0 0 0 0 1 0	Intercept tone on
0 0 0 0 0 0 1 1	Network congestion tone on
0 0 0 0 0 1 0 0	Busy tone on
0 0 0 0 0 1 0 1	Confirm tone on
0 0 0 0 0 1 1 0	Answer tone on
0 0 0 0 0 1 1 1	Call waiting tone
0 0 0 0 1 0 0 0	Off-hook warning tone
0 0 0 0 1 0 0 1	Pre-emption tone on
0 0 1 1 1 1 1 1	Tones off
0 1 0 0 0 0 0 0	Alerting on – pattern 0 (Note 1)
0 1 0 0 0 0 0 1	Alerting on – pattern 1 (Note 1)
0 1 0 0 0 0 1 0	Alerting on – pattern 2 (Note 2)
0 1 0 0 0 0 1 1	Alerting on – pattern 3 (Note 1)
0 1 0 0 0 1 0 0	Alerting on – pattern 4 (Note 1)
0 1 0 0 0 1 0 1	Alerting on – pattern 5 (Note 1)
0 1 0 0 0 1 1 0	Alerting on – pattern 6 (Note 1)
0 1 0 0 0 1 1 1	Alerting on – pattern 7 (Note 1)
0 1 0 0 1 1 1 1	Alerting off

All other values are reserved.

NOTE 1 – The use of these patterns is network dependent.

NOTE 2 – Used for special/priority alerting.

4.5.29 Transit network selection

The purpose of the Transit network selection information element is to identify one requested transit network. The Transit network selection information element may be repeated in a message to select a sequence of transit networks through which a call must pass (see Annex C).

The Transit network selection information element is coded as shown in Figure 4-35 and Table 4-25. The maximum length of this information element is network dependent.

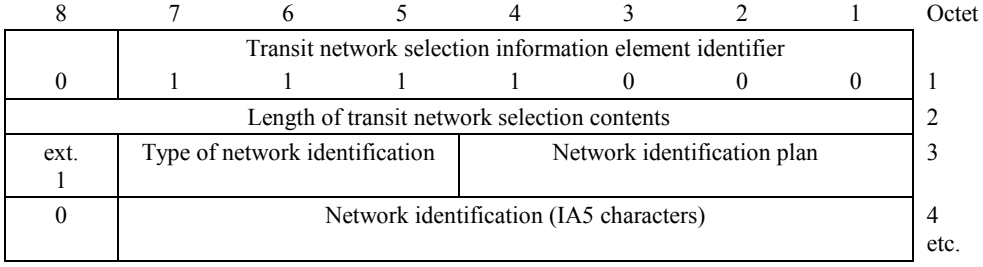


Figure 4-35/Q.931 – Transit network selection information element

Table 4-25/Q.931 – Transit network selection information element

Type of network identification (octet 3)

Bits
7 6 5
0 0 0 User specified
0 1 0 National network identification (Note 1)
0 1 1 International network identification

All other values are reserved.

NOTE 1 – In the case that "type of network identification" is coded as 010, "national network identification", "national identification plan" is coded according to national specification.

Network identification plan (octet 3)

Bits
4 3 2 1
0 0 0 0 Unknown
0 0 0 1 Carrier Identification Code (Note 2)
0 0 1 1 Data network identification code (Recommendation X.121 [21])

All other values are reserved.

NOTE 2 – Carrier Identification Codes may be an appropriate method of identifying the network serving the remote user.

Network identification (octet 4)

These IA5 characters are organized according to the network identification plan specified in octet 3.

4.5.30 User-user

The purpose of the User-user information element is to convey information between ISDN users. This information is not interpreted by the network, but rather is carried transparently and delivered to the remote user(s).

The User-user information element is coded as shown in Figure 4-36 and Table 4-26. There are no restrictions on content of the user information field.

In SETUP, ALERTING, CONNECT, DISCONNECT, RELEASE and RELEASE COMPLETE messages, the User-user information element has a network dependent maximum size of 35 or 131 octets. The evolution to a single maximum value is the long term objective; the exact maximum value is the subject of further study.

In USER INFORMATION messages sent in association with a circuit-mode connection, the User-user information element has a network dependent maximum size of 35 or 131 octets. For USER INFORMATION messages sent in a temporary or permanent user-user signalling connection, the user information field contained inside this information element has a maximum size equal to the maximum size of messages defined in clause 3, that is 260 octets.

NOTE – The User-user information element is transported transparently by an ISDN between a call originating entity, e.g. a calling user and the addressed entity, e.g. a remote user or a high layer function network node addressed by the call originating entity.

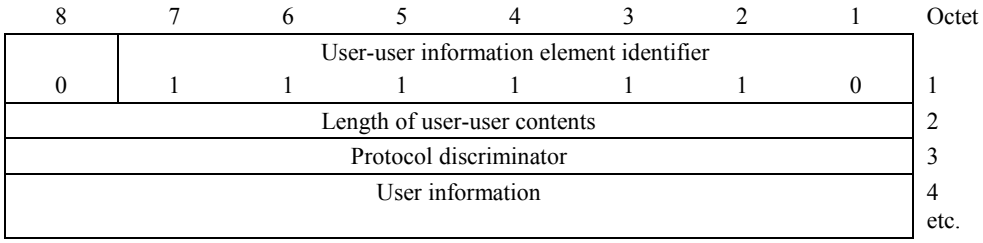


Figure 4-36/Q.931 – User-user information element

Table 4-26/Q.931 – User-user information element

Protocol discriminator (octet 3)

Bits 8 7 6 5 4 3 2 1	
0 0 0 0 0 0 0 0	User-specific protocol (Note 1)
0 0 0 0 0 0 0 1	OSI high layer protocols
0 0 0 0 0 0 1 0	Recommendation X.244 [44] (Note 2)
0 0 0 0 0 0 1 1	Reserved for system management convergence function
0 0 0 0 0 1 0 0	IA5 characters (Note 4)
0 0 0 0 0 1 0 1	X.208 and X.209 coded user information (Note 5)
0 0 0 0 0 1 1 1	Recommendation V.120 [9] rate adaption
0 0 0 0 1 0 0 0	Q.931/I.451 user-network call control messages
0 0 0 1 0 0 0 0 through 0 0 1 1 1 1 1 1	Reserved for other network layer or layer 3 protocols, including Recommendation X.25 [5] (Note 3)
0 1 0 0 0 0 0 0 through 0 1 0 0 1 1 1 1	National use
0 1 0 1 0 0 0 0 through 1 1 1 1 1 1 1 0	Reserved for other network layer or layer 3 protocols, including Recommendation X.25 (Note 3)

All other values are reserved.

NOTE 1 – The user information is structured according to user needs.

NOTE 2 – The user information is structured according to Recommendation X.244 which specifies the structure of X.25 call user data.

NOTE 3 – These values are reserved to discriminate these protocol discriminators from the first octet of an X.25 packet including general format identifier.

NOTE 4 – The user information consists of IA5 characters.

NOTE 5 – The number of X.208 and X.209 components contained in a User-user information element as well as their semantics and use are user-application dependent and may be subject to other Recommendations.

4.6 Information element for packet communications

The information elements defined below are intended to be used in the support of packet communications as described in clause 6 and Recommendation X.31 [14].

The use of these information elements for out-of-band call control for packet calls is for further study.

4.6.1 Closed user group

The purpose of the Closed user group information element is to indicate the closed user group to be used for that call. It may be used for X.25 packet-mode calls when either an X.25 CUG selection facility or an X.25 CUG with Outgoing Access selection facility is received in an X.25 Incoming Call packet and X.25 and Q.931 mapping applies.

The Closed user group information element is coded as shown in Figure 4-37 and Table 4-27.

The maximum length of this information element is 7 octets.

8	7	6	5	4	3	2	1	Octet
	Closed user group information element identifier							
0	1	0	0	0	1	1	1	1
Length of information element contents								2
ext.	Spare							
1	0	0	0	0	CUG indication			3
Spare	CUG index code (IA5) characters)							4
0								etc.

Figure 4-37/Q.931 – Closed user group information element

Table 4-27/Q.931 – Closed user group information element

CUG indication (octet 3)

Bits

3 2 1

0 0 1 Closed user group selection

0 1 0 Closed user group with outgoing access selection and indication

All other values are reserved.

CUG index code (octet 4)

Bits

7 6 5 4 3 2 1	CUG index
0 1 1 0 0 0 0	0
0 1 1 0 0 0 1	1
0 1 1 0 0 1 0	2
0 1 1 0 0 1 1	3
0 1 1 0 1 0 0	4
0 1 1 0 1 0 1	5
0 1 1 0 1 1 0	6
0 1 1 0 1 1 1	7
0 1 1 1 0 0 0	8
0 1 1 1 0 0 1	9

All other values are reserved.

NOTE – The CUG index code should be represented by up to four IA5 characters, encoded as shown above.

4.6.2 End-to-end transit delay

The purpose of the End-to-end transit delay information element is to request and indicate the nominal maximum permissible transit delay applicable on a per call basis to that virtual call.

The End-to-end transit delay information element is coded as shown in Figure 4-38 and Table 4-28.

The maximum length of this information element is 11 octets.

8	7	6	5	4	3	2	1	Octet
	End-to-end transit delay information element identifier							
0	1	0	0	0	0	1	0	1
Length of end-to-end transit delay contents								2
ext.	Spare					Cumulative transit		
0	0	0	0	0	0	delay value		3
ext.	Cumulative transit delay value (cont.)							3a
0								
ext.	Cumulative transit delay value (cont.)							3b
1								
ext.	Spare					Requested end-to-		4*
0	0	0	0	0	0	end transit delay value		(Note 1)
ext.	Requested end-to-end transit delay value (cont.)							4a*
0								
ext.	Requested end-to-end transit delay value (cont.)							4b*
1								
ext.	Spare					Maximum end-to-		5*
0	0	0	0	0	0	end transit delay value		(Note 2)
ext.	Maximum transit delay value (cont.)							5a*
0								
ext.	Maximum transit delay value (cont.)							5b*
1								

NOTE 1 – Octets 4, 4a and 4b are optional. If present, these octets are always interpreted as Requested end-to-end transit delay.

NOTE 2 – Octets 5, 5a and 5b are optional. If present, octets 4, 4a and 4b must also be present.

Figure 4-38/Q.931 – End-to-end transit delay information element

Table 4-28/Q.931 – End-to-end transit delay information element

Cumulative transit delay value [octet 3 (bits 1-2) octets 3a and 3b]

Cumulative transit delay value binary encoded in milliseconds. Bit 2 of octet 3 is the highest order bit and bit 1 of octet 3b is the lowest order bit. The cumulative transit delay value occupies 16 bits total.

Requested end-to-end transit delay value [octet 4 (bits 1-2) octets 4a and 4b]

Requested end-to-end transit delay value binary encoded in milliseconds. Bit 2 of octet 4 is the highest order bit and bit 1 of octet 4b is the lowest order bit. The requested end-to-end transit delay value occupies 16 bits total.

Maximum end-to-end transit delay value [octet 5 (bits 1-2) octets 5a and 5b]

Maximum end-to-end transit delay value binary encoded in milliseconds. Bit 2 of octet 5 is the highest order bit and bit 1 of octet 5b is the lowest order bit. The maximum end-to-end transit delay value occupies 16 bits total.

NOTE – For an X.31 type of access to an ISDN, the procedures only apply in the notification phase at the terminating exchange. At the terminating exchange, if the End-to-End Transit Delay facility is present in the X.25 [5] incoming call request packet, the contents should be copied into End-to-end transit delay information element as follows:

i) The cumulative transit delay field (octets 3 and 4) of the X.25 end-to-end transit delay facility should be copied into octets 3, 3a and 3b. The bit order should be preserved as described above in the description.

ii) If octets 5 and 6 are present in the X.25 end-to-end transit delay facility, they should be interpreted as the requested end-to-end transit delay value. The value present should be copied into octets 4, 4a and 4b. The bit order should be preserved as described above in the description.

iii) If octets 7 and 8 are present in the X.25 end-to-end transit delay facility, the value present is the minimum end-to-end transit delay allowed. Octets 7 and 8 should be copied into octets 5, 5a and 5b. The bit order should be preserved as described above in the description.

4.6.3 Information rate

The purpose of the Information rate information element is to notify the terminating user of the throughput indicated by the incoming X.25 call request packet.

The Information rate information element is coded as shown in Figure 4-39 and Tables 4-29 and 4-30.

The maximum length of this information element is 6 octets.

8	7	6	5	4	3	2	1	Octet
	Information rate information element identifier							
0	1	1	0	0	0	0	0	1
Length of information rate contents								2
ext.	Spare		Incoming information rate					
1	0	0						3
ext.	Spare		Outgoing information rate					
1	0	0						4
ext.	Spare							
1	0	0	Minimum incoming information rate					5
ext.	Spare							
1	0	0	Minimum outgoing information rate					6

NOTE – This information element applies only in the notification phase at the terminating exchange. If the throughput class facility/minimum throughput class facility is present in the X.25 incoming call packet, the contents may be copied into the Information rate information element. The Information rate for the direction of data transmission from the calling user is copied into octet 3/5. The information rate for the direction of data transmission from the called user is copied into octet 4/6. The bit order should be preserved as described in Table 4-30.

Figure 4-39/Q.931 – Information rate information element

Table 4-29/Q.931 – Information rate information element

Incoming/outgoing information rate (octets 3 and 4)

The incoming outgoing information rate fields are used to indicate the information rate in the direction network-to-user, and user-to-network, respectively.

The information rate for the direction of data transmission from the calling DTE is indicated in bits 5, 4, 3, 2 and 1 of octet 3. The information rate for the direction of data transmission from the called DTE is indicated in bits 5, 4, 3, 2 and 1 of octet 4. The bits are coded as specified in Table 4-30.

Minimum incoming/outgoing information rate (octets 5 and 6)

The minimum information rate for the direction of data transmission from the calling DTE is indicated in bits 5, 4, 3, 2 and 1 of octet 5. The minimum information rate for the direction of data transmission from the called DTE is indicated in bits 5, 4, 3, 2 and 1 of octet 6. The bits are encoded as specified in Table 4-30.

Table 4-30/Q.931 – Throughput class coding

Bits					Throughput class
5	**4**	**3**	**2**	**1**	**(bit/s)**
0	0	0	0	0	Reserved
0	0	0	0	1	Reserved
0	0	0	1	0	Reserved
0	0	0	1	1	75
0	0	1	0	0	150
0	0	1	0	1	300
0	0	1	1	0	600
0	0	1	1	1	1 200
0	1	0	0	0	2 400
0	1	0	0	1	4 800
0	1	0	1	0	9 600
0	1	0	1	1	19 200
0	1	1	0	0	48 000
0	1	1	0	1	64 000
0	1	1	1	0	Reserved
0	1	1	1	1	Reserved

4.6.4 Packet layer binary parameters

The purpose of the Packet layer binary parameters information element is to indicate requested layer 3 parameter values to be used for the call.

The Packet layer binary parameters information element is coded as shown in Figure 4-40 and Table 4-31.

The maximum length of this information element is 3 octets.

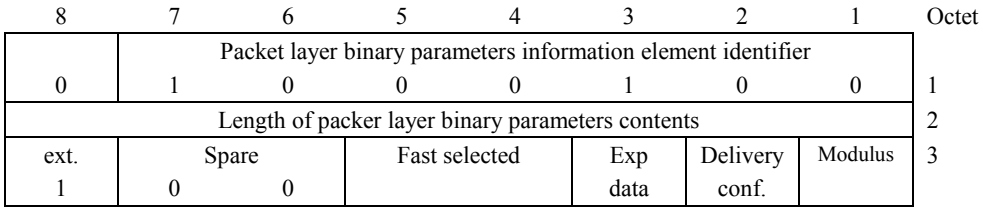


Figure 4-40/Q.931 – Packet layer binary parameters information element

Table 4-31/Q.931 – Packet layer binary parameters information element

Fast select (octet 3)

Bits 5 4	
0 0 0 1	Fast select not requested
1 0	Fast select requested with no restriction of response
1 1	Fast select requested with restrictions of response

Expedited data (octet 3)

Bit 3	
0	No request/request denied
1	Request indicated/request accepted

Delivery confirmation (octet 3)

Bit 2	
0	Link-by-link confirmation
1	End-to-end confirmation

Modulus (octet 3)

Bit 1	
0	Modulus 8 sequencing
1	Modulus 128 sequencing

4.6.5 Packet layer window size

The purpose of the Packet layer window size information element is to indicate the requested layer 3 window size value to be used for the call. The values are binary encoded.

The Packet layer window size information element is coded as shown in Figure 4-41.

The maximum length of this information element is 4 octets.

8	7	6	5	4	3	2	1	Octet
Packet layer window size information element identifier								
0	1	0	0	0	1	0	1	1
Length of packer layer window size contents								2
ext. 1	Forward value							3
ext. 1	Backward value							4* (Note)

NOTE – This octet may be omitted. When omitted, it indicates a request for the default value.

Figure 4-41/Q.931 – Packet layer window size information element

4.6.6 Packet size

The purpose of the Packet size information element is to indicate the requested packet size values to be used for the call. The values are encoded log 2.

The Packet size information element is coded as shown in Figure 4-42.

The maximum length of this information element is 4 octets.

8	7	6	5	4	3	2	1	Octet
	Packet size information element identifier							
0	1	0	0	0	1	1	0	1
Length of packet size contents								2
ext. 1	Forward value (Note 2)							3
ext. 1	Backward value (Note 2)							4* (Note 1)

NOTE 1– This octet may be omitted. When omitted, it indicates a request for the default value.

NOTE 2 – 000 0000 is reserved.

Figure 4-42/Q.931 – Packet size information element

4.6.7 Redirecting number

The purpose of the Redirecting number information element is to identify the number from which a call diversion or transfer was invoked.

The Redirecting number information element is coded as shown in Figure 4-43 and Table 4-32.

The maximum length of this information element is network dependent.

8	7	6	5	4	3	2	1	Octet
	Redirecting number information element identifier							
0	1	1	1	0	1	0	0	1
Length of redirecting number contents								2
ext. 0/1	Type of number			Numbering plan identification				3
ext. 0/1	Presentation indicator		Spare 0		0	Screening indicator		3a* 1
ext. 1	Spare 0	0	0	Reason for redirection				3b* 1
Spare 0	Number digits (IA5 characters)							4 etc.

Figure 4-43/Q.931 – Redirecting number information element

Table 4-32/Q.931 – Redirecting number information element

Type of number (octet 3) (Note 1)

Bits

7 6 5

0 0 0 Unknown (Note 2)
0 0 1 International number (Note 3)
0 1 0 National number (Note 3)
0 1 1 Network specific number (Note 4)
1 0 0 Subscriber number (Note 3)
1 1 0 Abbreviated number
1 1 1 Reserved for extension

All other values are reserved.

NOTE 1 – For the definition of international, national and subscriber number, see Recommendation I.330 [18].

NOTE 2 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number of digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

NOTE 3 – Prefix or escape digits shall not be included.

NOTE 4 – The type of number "network specific number" is used to indicate administration/service number specific to the serving network, e.g. used to access an operator.

Numbering plan identification (octet 3)

Numbering plan (applies for type of number = 000, 001, 010 and 100)

Bits

4 3 2 1

0 0 0 0 Unknown (Note 5)
0 0 0 1 ISDN/telephony numbering plan (Recommendation E.164 [19])
0 0 1 1 Data numbering plan (Recommendation X.121 [21])
0 1 0 0 Telex numbering plan (Recommendation F.69 [22])
1 0 0 0 National standard numbering plan
1 0 0 1 Private numbering plan
1 1 1 1 Reserved for extension

All other values are reserved.

NOTE 5 – The numbering plan "unknown" is used when the user on network has no knowledge of the numbering plan. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

Presentation indicator (octet 3a)

Bits

7 6

0 0 Presentation allowed
0 1 Presentation restricted
1 0 Number not available due to interworking
1 1 Reserved

NOTE 6 – The meaning and use of this field is defined in clause 3/Q.951 and clause 4/Q.951

Table 4-32/Q.931 – Redirecting number information element *(concluded)*

Screening indicator (octet 3a)

Bits
2 1
0 0 User-provided, not screened
0 1 User-provided, verified and passed
1 0 User-provided, verified and failed
1 1 Network provided

NOTE 7 – If octet 3a is omitted, "00 – user-provided, not screened" is assumed.

Reason for redirection (octet 3b)

Bits
4 3 2 1
0 0 0 0 Unknown
0 0 0 1 Call forwarding busy or called DTE busy
0 0 1 0 Call forwarding no reply
0 1 0 0 Call deflection
1 0 0 1 Called DTE out of order
1 0 1 0 Call forwarding by the called DTE
1 1 1 1 Call forwarding unconditional or systematic call redirection
All other values are reserved.

Number digits (octets 4, etc.)

This field is coded with IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

4.6.8 Reverse charging indication

The purpose of the Reverse charging information element is to indicate that reverse charging has been requested for that call. It may be used for X.25 packet-mode calls when either an X.25 Reverse charging facility is received in an X.25 Incoming Call packet and X.25 and Q.931 mapping applies.

The Reverse charging information element is coded as shown in Figure 4-44 and Table 4-33.

The maximum length of this information element is 3 octets.

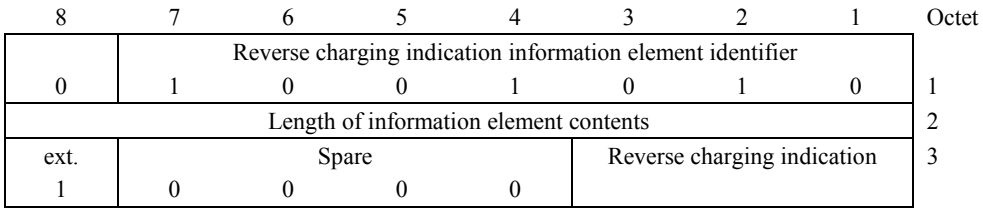


Figure 4-44/Q.931 – Reverse charging indication information element

Table 4-33/Q.931 – Reverse charging indication information element

Reverse charging indication (octet 3)

Bits
3 2 1
0 0 1 Reverse charging requested
All other values are reserved.

4.6.9 Transit delay selection and indication

The purpose of the Transit delay selection and indication information element is to request the nominal maximum permissible transit delay applicable on a per call basis to that virtual call.

The Transit delay selection and indication information element is coded as shown in Figure 4-45 and Table 4-34.

The maximum length of this information element is 5 octets.

8	7	6	5	4	3	2	1	Octet
Transit delay selection and indication information element identifier								
0	1	0	0	0	0	1	1	1
Length of transit delay selection and indication contents								2
ext.	Spare					Transit delay selection		3
0	0	0	0	0	0	and indication value		
ext. 0	Transit delay selection and indication value (cont.)							3a
ext. 1	Transit delay selection and indication value (cont.)							3b

Figure 4-45/Q.931 – Transit delay selection and indication information element

Table 4-34/Q.931 – Transit delay selection and indication information element

Transit delay selection and indication value [octet 3 (bits 1-2), octets 3a and 3b] Transit delay value binary encoded in milliseconds. Bit 2 of octet 3 is the highest order bit and bit 1 of octet 3b is the lowest order bit. The transit delay value occupies 16 bits total. NOTE – For an X.31 [14] type of access to an ISDN, the procedures only apply in the notification phase at the terminating exchange. At the terminating exchange, if the Transit Delay Selection and Indication facility is present in the X.25 [5] incoming call request packet, the two-octet value should be copied into octets 3, 3a and 3b with the highest order bit contained in bit 2 of octet 3 and the lowest order bit contained in bit 1 of octet 3b.

5 Circuit-switched call control procedures

This clause provides the D-channel signalling procedures in support of circuit-mode bearer capabilities other than multirate (64 kbit/s base rate).

Extensions to this basic protocol and exceptions that apply in the case of packet-mode connections or of circuit-mode multirate (64 kbit/s base rate) or supplementary services are described elsewhere in this Recommendation.

The call states referred to in this clause cover the states perceived by the network, states perceived by the user, and states which are common to both user and network. Unless specifically qualified, all states described in the following text should be understood as common (see 2.1.1 and 2.1.2 for user and network call states respectively). An overview diagram of call states is given in Figures A.2 and A.3 (see Annex A).

Detailed Specification and Description Language (SDL) diagrams for the procedures specified in this clause are contained in Figures A.4 through A.6. When there is an ambiguity in the narrative text, the SDL diagrams in Figures A.4 through A.6 should be used to resolve the conflict. Where the text and the SDLs are in disagreement, the text should be used as the prime source.

NOTE 1 – This clause describes the sequences of messages associated with the control of circuit-switched connections. Optional extensions to this basic protocol and exceptions that apply in the case of packet-mode connections or supplementary services are described elsewhere in this Recommendation, in Recommendation Q.932 [4] or the Q.95x-series Recommendations [83]. Annex D also contains extensions to the basic call establishment procedures defined in clause 5 for symmetric signalling.

All messages in this Recommendation may contain two types of information elements, functional and/or stimulus. Functional information elements are characterized as requiring a degree of intelligent processing by the terminal in either their generation or analysis. Stimulus information elements, on the other hand, are either generated as a result of a single event at the user/terminal interface or contain a basic instruction from the network to be executed by the terminal.

As a general principle, all the messages sent by the network to the user may contain a Display information element whose contents may be displayed by the terminal; the content of this information element shall be network dependent.

NOTE 2 – Keypad facility information elements shall be conveyed only in the direction user-to-network. Display information elements shall be conveyed only in the direction network-to-user.

In addition to the messages exchanged as described in the following subclauses, INFORMATION messages for call control may be sent by the user or by the network only after the first response to a SETUP message has been sent or received, and before the clearing of the call reference is initiated. An INFORMATION message received in the Release Request state may be ignored.

In order to accommodate the transfer of Layer 3 messages which exceeds the data link layer maximum frame length (i.e. defined in Recommendation Q.921 [3]), a method of message segmentation and reassembly may optionally be implemented as described in Annex H. Message segmentation shall only be used where all the information comprising the unsegmented message is available at the time of sending the first message segment.

NOTE 3 – Message segmentation is not used to replace existing procedures where information is yet to be provided by call control, e.g. digit-by-digit sending in overlap mode, although this may be used in addition. Message segmentation shall only be used when the message length exceeds the value of the N201 parameter defined in Recommendation Q.921 [3].

5.1 Call establishment at the originating interface

Before these procedures are invoked, a reliable data link connection must be established between the user (TE/NT2) and the network. All layer 3 messages shall be sent to the data link layer using a DL-DATA request primitive. The data link services described in Recommendations Q.920/I.440 [45] and Q.921 [3] are assumed.

5.1.1 Call request

A user initiates call establishment by transferring a SETUP message across the user-network interface. Following the transmission of the SETUP message, the call shall be considered by the user to be in the call initiated state. The message shall always contain a call reference, selected according to the procedures given in 4.3. In selecting a call reference, the dummy call reference value shall not be used in association with the basic call. The bearer capability information element is mandatory in the SETUP message, even in the case of overlap sending.

If the user knows all appropriate channels controlled by the D-channel are in use, it shall not transfer a SETUP message across the user-network interface. If the user does not monitor the status of channels in use, it may send a SETUP during an all channels busy condition. In this case, the network returns a RELEASE COMPLETE message with cause No. 34, *no circuit/channel available*.

Furthermore, the SETUP message may also contain all or part of the call information (i.e. address and facility requests) necessary for call establishment depending on whether *en bloc* or overlap procedures are being used respectively (see 5.1.3).

If *en bloc* sending is used, the SETUP message shall contain all the information required by the network to process the call, and, in particular, the called party address information if present, is contained as follows:

a) in the called party number information element possibly completed by the called party subaddress information element; or

b) the Keypad facility information element which may also be used to convey other call information.

NOTE – The support of a) is mandatory in all networks. Whether the support of b) is mandatory or optional requires further study.

If *en bloc* sending is used, the SETUP message may contain the sending complete indication (i.e. either the Sending complete information element or the "#" character within the Called party number information element). It is mandatory for the network to recognize at least one of the sending complete indications; however, the recognition of the sending complete IE is preferred.

For overlap sending, see 5.1.3.

Called party subaddress information, if present, shall be given in the Called party subaddress information element and, in the case of overlap sending, shall be sent only in the SETUP message.

5.1.2 B-channel selection – Originating

In the SETUP message, the user will indicate one of the following:

a) channel is indicated, no acceptable alternative [i.e. channel is indicated by the information channel selection field of octet 3 (bits 2-1) and, if applicable, octet 3.3, and the preferred/exclusive field (bit 4 of octet 3) is set to "1" in the Channel identification information element]; or

b) channel is indicated, any alternative is acceptable [i.e. channel is indicated by the information channel selection field of octet 3 (bits 2-1) and, if applicable, octet 3.3, and the preferred/exclusive field (bit 4 of octet 3) is set to "0" in the Channel identification information element]; or

c) any channel is acceptable [i.e. either the information channel selection field of octet 3 (bits 2-1) of the Channel identification information element indicate "any channel" or the Channel identification information element is not present].

If no indication is included, alternative c) is assumed. In cases a) and b), if the indicated channel is available, the network selects it for the call.

In case b), if the network cannot grant the preferred channel, it selects any other available B-channel associated with the D-channel. In case c), the network selects any available B-channel associated with the D-channel.

NOTE – It is recommended that TEs connected to the ISDN basic access in a point-to-multipoint configuration use alternative c) for basic circuit-switched call control unless the TE is already using a given B-channel.

The selected B-channel is indicated in the Channel identification information element coded as "channel is indicated, no acceptable alternative" in the first message returned by the network in response to the SETUP message (i.e. a SETUP ACKNOWLEDGE or CALL PROCEEDING message). After transmitting this message, the network shall activate the B-channel connection.

The user need not attach until receiving a CALL PROCEEDING/SETUP ACKNOWLEDGE/PROGRESS/ALERTING message with the progress indicator No. 8, *in-band information or appropriate pattern is now available*, or progress indicator No. 1, *call is not end-to-end ISDN; further call progress information may be available in-band*. Prior to this time, the network cannot assume that the user has attached to the B-channel. After this time, the user shall be connected to the B-channel, provided the equipment does not generate local tone. Upon receipt of the CONNECT message, the user shall attach to the B-channel (if it has not already done so).

In case a) if the specified channel is not available, and in cases b) and c) if no channel is available, a RELEASE COMPLETE message with cause No. 44, *requested circuit/channel not available* or cause No. 34, *no circuit/channel available*, respectively, is sent by the network as described in 5.3.

In case a), if the specified channel does not exist, cause No. 82, *identified channel does not exist*, is included in the RELEASE COMPLETE message.

5.1.3 Overlap sending

If overlap sending is used, the SETUP message contains either:

a) no called number information; or

b) incomplete called number information; or

c) called number information which the network cannot determine to be complete.

On receipt of such a SETUP message, the network starts timer T302 (the value of timer T302 is specified in 9.1), sends a SETUP ACKNOWLEDGE message to the user and enters the overlap sending state. In case a), the network will return dial tone, if required by the tone option. In this case, it may include progress indicator No. 8, *in-band information or appropriate pattern is now available,* in the SETUP ACKNOWLEDGE message.

NOTE 1 – Some networks which systematically provide the conventional telephone dial tone will not generate the progress indicator when providing the dial tone.

When the SETUP ACKNOWLEDGE message is received, the user enters the overlap sending state and optionally starts timer T304 (the value of timer T304 is specified in 9.2).

After receiving the SETUP ACKNOWLEDGE message, the user sends the remainder of the call information (if any) in one or more INFORMATION messages.

The called party number information may be provided by the user as follows:

a) in the called party number information element; or

b) in the Keypad facility information element, exclusively.

The called party number must be sent in a unique way.

NOTE 2 – The support of a) is mandatory in all networks. Whether the support of b) is mandatory or optional requires further study.

NOTE 3 – Besides the possible called party number [conveyed by method a) or b) as described above], the INFORMATION messages may contain additional call information (i.e. for supplementary services). The interpretation of the contents of Keypad facility information elements is network-specific, and in accordance with the dialling plan provided to that user. It should be noted that the user shall transfer all the additional call information (contained within the Keypad facility information element) before the network determines that the called party number (contained within the Called party number information element or the Keypad facility information element) is complete, and terminates the overlap sending procedure using the CALL PROCEEDING message as recommended in 5.1.5.2.

If, for symmetry purposes, the user employs timer T304, the user restarts timer T304 when each INFORMATION message is sent.

The call information in the message which completes the information sending may contain a *sending complete* indication, (e.g. the # character or, as a network option, the sending complete information element) appropriate to the dialling plan being used. The network shall restart timer T302 on the receipt of every INFORMATION message not containing a sending complete indication.

5.1.4 Invalid call information

If, following the receipt of a SETUP message or during overlap sending, the network determines that the call information received from the user is invalid, (e.g. invalid number), then the network shall follow the procedures described in 5.3 with a cause such as one of the following:

No. 1 – *Unallocated (unassigned) number;*

No. 3 – *No route to destination;*

No. 22 – *Number changed;*

No. 28 – *Invalid number format (address incomplete).*

5.1.5 Call proceeding

5.1.5.1 Call proceeding, *en bloc* sending

If *en bloc* sending is used, (i.e. the network can determine that the SETUP message contains all the information required from the user to establish the call), and if the network can determine that access to the requested service is authorized and available, the network shall send a CALL PROCEEDING message to the user. This acknowledges the SETUP message and indicates that the call is being processed. The network will then enter the Outgoing Call Proceeding state. When the user receives the CALL PROCEEDING message, the user shall also enter the Outgoing Call Proceeding state.

Similarly, if the network determines that a requested service is not authorized or is not available, the network shall initiate call clearing in accordance with 5.3, with one of the following causes:

No. 57 – *Bearer capability not authorized*;

No. 58 – *Bearer capability not presently available*;

No. 63 – *Service or option not available, unspecified*; or

No. 65 – *Bearer capability not implemented*.

NOTE 1 – If a supplementary service is not authorized or is not available, the procedure to be used is defined in the supplementary service control procedures.

NOTE 2 – When the network cannot assign a channel due to congestion, the procedures of 5.1.2 shall be followed.

5.1.5.2 Call proceeding, overlap sending

If overlap sending is used, then following the occurrence of one of these conditions:

a) the receipt by the network of a sending complete indication which the network understands; or

b) analysis by the network that all call information necessary to effect call establishment has been received,

and if the network can determine that access to the requested service and supplementary service is authorized and available, the network shall: send a CALL PROCEEDING message to the user; stop timer T302; and enter the Outgoing Call Proceeding state. Similarly, if the network determines that a requested service or supplementary service is not authorized or is not available, the network shall initiate call clearing in accordance with 5.3, with one of the following causes:

No. 57 – *Bearer capability not authorized*;

No. 58 – *Bearer capability not presently available*;

No. 63 – *Service or option not available, unspecified*; or

No. 65 – *Bearer capability not implemented*.

NOTE 1 – The CALL PROCEEDING message is sent to indicate that the requested call establishment has been initiated, and no more call establishment information will be accepted.

NOTE 2 – If a supplementary service is not authorized or is not available, the procedure to be used is defined in the supplementary service control procedures.

NOTE 3 – When the network cannot assign a channel due to congestion, the procedures of 5.1.2 shall be followed.

When the user receives the CALL PROCEEDING message, the user shall enter the outgoing call proceeding state. If, for symmetry purposes, the calling user employs timer T304, the user shall stop timer T304 when the CALL PROCEEDING message is received. If, for symmetry purposes, the calling user employs timer T304 then, on expiry of T304, the user shall initiate call clearing in accordance with 5.3 with cause No. 102, *recovery on timer expiry*.

An alerting or connect indication received from the called party will stop timer T302 and cause an ALERTING or CONNECT message respectively to be sent to the calling user. No CALL PROCEEDING message shall be sent by the network. If, for symmetry purposes, the calling user employs timer T304, the user shall stop timer T304 on receiving the ALERTING or CONNECT message.

At the expiration of timer T302, the network shall:

i) initiate call clearing in accordance with 5.3, with cause No. 28, *invalid number format* (address incomplete) sent to the calling user and with cause No. 102, *recovery on timer expiry,* sent towards the called user, if the network determines that the call information is definitely incomplete; otherwise

ii) send a CALL PROCEEDING message and enter the outgoing call proceeding state.

5.1.6 Notification of interworking at the originating interface

During call establishment, the call may leave the ISDN environment, e.g. because of interworking with another network, with a non-ISDN user, or with non-ISDN equipment within the called user's premises. When such situations occur, a progress indication shall be returned to the calling user either:

a) in an appropriate call control message when a state change is required: CALL PROCEEDING, ALERTING, SETUP ACKNOWLEDGE or CONNECT; or

b) in the PROGRESS message when no state change is appropriate.

One of the following progress description values shall be included in the Progress indicator information element in the messages sent to the user, (for further information, see Annex G).

1) No. 1 – *Call is not end-to-end ISDN; further call progress information may be available in-band*;

2) No. 2 – *Destination address is non-ISDN*;

3) No. 4 – *Call has returned to the ISDN*.

If the Progress indicator information element is included in a call control message, the procedures as described in the rest of 5.1 apply but T310 shall not be started if progress indicator 1 or 2 has been delivered in the CALL PROCEEDING message or in a previous PROGRESS message. If the Progress indicator information element is included in the Progress message, no state change will occur but any supervisory timers shall be stopped except user timer T301 and network timer T302. In

both cases, if Progress description No 1 is received by the user, the user shall connect to (if not connected already) and then monitor the B channel for further in-band information.

If the interface at which the progress indication originates is the point at which a call enters the ISDN environment from a non-ISDN environment, one or more of the following progress indicator information elements shall be included in the SETUP message sent to the network:

i) No. 1 – *Call is not end-to-end ISDN; further call progress information may be available in-band*;

ii) No. 3 – *Origination address is non-ISDN*.

5.1.7 Call confirmation indication

Upon receiving an indication that user alerting has been initiated at the called address, the network shall send an ALERTING message across the user-network interface of the calling address, and shall enter the call delivered state. When the user receives the ALERTING message, the user may begin an internally-generated alerting indication and shall enter the call delivered state.

5.1.8 Call connected

Upon the network receiving an indication that the call has been accepted, the network shall send a CONNECT message across the user-network interface to the calling user, and shall enter the Active state. As a network option, the Date/time information element may be included in the CONNECT message.

This message indicates to the calling user that a connection has been established through the network and stops a possible local indication of alerting.

On receipt of the CONNECT message, the calling user shall stop any user-generated alerting indication, may optionally send a CONNECT ACKNOWLEDGE message and shall enter the active state. The network shall not take any action on receipt of a CONNECT ACKNOWLEDGE message when it perceives the call to be in the active state.

5.1.9 Call rejection

Upon receiving an indication that the network or the called user is unable to accept the call, the network shall initiate call clearing at the originating user-network interface as described in 5.3, using the cause provided by the terminating network or the called user.

5.1.10 Transit network selection

When the transit network selection information element is present, the call shall be processed according to Annex C.

5.2 Call establishment at the destination interface

This procedure assumes that a data link connection providing services described in Recommendation Q.920/I.440 [45] may not exist before the first layer 3 message (SETUP) is transferred across the interface. However, reliable data link connections must be established by each of the users (terminals and/or NT2s) at the interface before they respond to the SETUP message.

Data link connections may be established by the TA, TE or NT2 as soon as a TEI is assigned (either locally or by automatic assignment procedure), and may be retained indefinitely. This may be recommended as a network option.

The SETUP message offered on a point-to-point data link shall be delivered to layer 2 using a DL-DATA request primitive. No use shall be made of the DL-UNIT-DATA request primitive other than for operation using the broadcast capability of the data link layer.

The call reference contained in all messages exchanged across the user-network interface shall contain the call reference value specified in the SETUP message delivered by the network. In selecting a call reference, the dummy call reference shall not be used in association with the basic call.

5.2.1 Incoming call

The network will indicate the arrival of a call at the user-network interface by transferring a SETUP message across the interface. This message is sent if the network can select an idle B-channel. In some circumstances (e.g. provision of other bearer services, see 6.1), the SETUP message may also be sent when no B-channel is idle. The number of calls presented in these circumstances may be limited.

In addition to the mandatory information elements, the SETUP message may include, as required, the information elements described in 3.1.14 (e.g. Display, Low layer compatibility).

If a multipoint terminal configuration exists at the user-network interface, this message shall be sent using a broadcast capability at the data link layer. In this case, the SETUP message should contain the appropriate part of the called party number as required and/or subaddress if provided. However, if the network has knowledge that a single-point configuration exists at the interface, a point-to-point data link shall be used to carry the SETUP message. The knowledge that a point-to-point configuration exists may be based on information entered at the time of configuration of the access. After sending the SETUP message, the network starts timer T303. If the SETUP message is sent via a broadcast data link, timer T312 shall also be started. (The values of timers T303 and T312 are specified in 9.1.) The network then enters the call present state.

NOTE 1 – Timer T312 is used to supervise the retention of the call reference when the SETUP message was transmitted by a broadcast data link. The value of timer T312 is such that if a network disconnect indication is received during the call establishment phase, it maximizes the probability that all responding users will be released prior to release of the call reference. Refer to 5.3.2 e) and 5.2.5.3 (Case 1) for procedures to be followed on expiry of timer T312.

If *en bloc* receiving is used, the SETUP message shall contain all the information required by the called user to process the call. In this case, the SETUP message may contain the Sending complete information element.

Upon receipt of a SETUP message, the user will enter the Call present state.

Depending on the contents of the received message, either *en bloc* receiving procedure (see 5.2.5.1) or overlap receiving procedure (see 5.2.4) follows. However, if the SETUP message includes the Sending complete information element, *en bloc* receiving procedure shall follow. Therefore, those users who support overlap receiving procedure shall recognize the Sending complete information element.

NOTE 2 – Users supporting only the *en bloc* receiving procedure need not recognize the Sending complete information element and may directly analyze the received SETUP message on the assumption that all the call information is contained in the message.

If no response to the SETUP message is received by the network before the first expiry of timer T303, the SETUP message will be retransmitted and timers T303 and T312 restarted.

NOTE 3 – In the case of overlap sending within the network, the appropriate part of the called party number as required (e.g. for supplementary services) may also be conveyed by means of INFORMATION messages to the called user on a point-to-point data link (see 5.2.4).

5.2.2 Compatibility checking

A user receiving a SETUP message shall perform compatibility checking before responding to that SETUP message. Any reference to user in 5.2.3 through 5.2.7 implicitly refers to a compatible user equipment. Annex B defines compatibility checking to be performed by users upon receiving a SETUP message.

When the SETUP message is delivered via a broadcast data link, an incompatible user shall either:

a) ignore the incoming call; or

b) respond by sending a RELEASE COMPLETE message with cause No. 88, *incompatible destination*, and enter the Null state. The network processes this RELEASE COMPLETE message in accordance with 5.2.5.3.

When the SETUP message is delivered via a point-to-point data link, an incompatible user shall respond with RELEASE COMPLETE message with cause No. 88, *incompatible destination*, and enter the Null state. The network shall process this RELEASE COMPLETE message in accordance with 5.2.5.3.

5.2.3 B-channel selection – Destination

5.2.3.1 SETUP message delivered by point-to-point data link

When the SETUP message is delivered by a point-to-point data link, negotiation for the selection of a B-channel will be permitted between the network and the user. Only B-channels controlled by the same D-channel will be the subject of the selection procedure. The selection procedure is as follows:

a) In the SETUP message, the network will indicate one of the following:

   1) channel is indicated, no acceptable alternative [i.e. channel is indicated by the information channel selection field of octet 3 (bits 2-1) and, if applicable, octet 3.3, and the preferred/exclusive field (bit 4 of octet 3) is set to "1" in the Channel identification information element]; or

   2) channel is indicated, any alternative is acceptable [i.e. channel is indicated by the information channel selection field of octet 3 (bits 2-1) and, if applicable, octet 3.3, and the preferred/exclusive field (bit 4 of octet 3) is set to "0" in the Channel identification information element]; or

   3) any channel is acceptable [i.e. either the information channel selection field of octet 3 (bits 2-1) of the Channel identification information element indicate "any channel" or the Channel identification information element is not present]; or

   4) no B-channel available [i.e. the information channel field (bits 2 and 1 of octet 3) of the Channel identification information element set to "00"].

   NOTE – Not all networks will support the *no B-channel available* condition.

b) In cases 1) and 2), if the indicated channel is acceptable and available, the user selects it for the call.

   In case 2), if the user cannot grant the indicated channel, it selects any other available B-channel associated with the D-channel and identifies that channel in the Channel identification information element as "channel is indicated, no acceptable alternative" in the first message sent in response to the SETUP message.

   In case 3), the user selects any available B-channel associated with the D-channel and identifies that channel in the first message sent in response to the SETUP message.

   If in case 1), the B-channel indicated in the first response message is not the channel offered by the network, or in cases 2) and 3) the B-channel indicated in the first response message is

unacceptable to the network, it will clear the call by sending a RELEASE message with cause No. 6, *channel unacceptable*.

In case 4), the user rejects the call by sending RELEASE COMPLETE message with cause No. 34, *no circuit/channel available*, unless it is able to proceed with the call. Unless the procedures of the Call Waiting supplementary service (see Q.953-series Recommendations [84]) are followed, the user wishing to re-use a B-channel it has already allocated to another call (e.g. by multiplexing packet calls) shall send the appropriate message containing the channel identification information element, coded as channel is indicated, no alternative acceptable.

c) If no Channel identification information element is present in the first response message, the B-channel indicated in the SETUP message will be assumed.

d) When a B-channel has been selected by the user, that channel may be connected by the user.

e) In case 1) if the indicated B-channel is not available, or in cases 2), 3), and 4) if no B-channel is available and the user cannot proceed with the offered call, the user returns a RELEASE COMPLETE message with cause No. 44, *requested circuit/channel not available*, or No. 34, *no circuit/channel available*, respectively, and returns to the Null state.

See 5.2.4 and 5.2.5 for the appropriate first response to the SETUP message.

5.2.3.2 SETUP message delivered by broadcast data link

When the SETUP message is delivered by a broadcast data link, the channel selection procedure, provided in 5.2.3.1, is not applicable. The network sends a SETUP message with the Channel identification information element indicating one of the following:

a) channel indicated, no alternative is acceptable [i.e. channel is indicated by the information channel selection field of octet 3 (bits 2-1) and, if applicable, octet 3.3, and the preferred/exclusive field (bit 4 of octet 3) is set to "1" in the Channel identification information element]; or

b) no channel available [i.e. the information channel field (bits 2 and 1 of octet 3) of the Channel identification information element set to "00"].

In case a), if the user can accept the call on the indicated channel, the user shall send the appropriate message (see 5.2.4 and 5.2.5). If the user cannot accept the call on the indicated channel, the user shall send a RELEASE COMPLETE message with cause No. 44, *requested circuit/channel not available*.

The user, in any case, must not connect to the channel until a CONNECT ACKNOWLEDGE message has been received.

In case b), the user not controlling any channel shall send a RELEASE COMPLETE message with cause No. 34, *no circuit/channel available*. Unless the procedures of the Call Waiting supplementary service (see Q.953-series Recommendations [84]) are followed, the user wishing to reuse a B-channel it has already allocated to another call (e.g. by multiplexing packet calls) shall send the appropriate message containing the Channel identification information element, coded as, *channel is indicated, no alternative acceptable*.

5.2.4 Overlap receiving

When a user determines that a received SETUP message contains either:

a) no called number information; or

b) incomplete called number information; or

c) called number information which the user cannot determine to be complete; and

when the user:

d) is compatible with other call characteristics (see Annex B); and

e) implements overlap receiving,

the user shall start timer T302; send a SETUP ACKNOWLEDGE message to the network and enter the Overlap receiving state.

When the SETUP ACKNOWLEDGE message is received, the network shall: stop timer T303; start timer T304; enter the Overlap receiving state; and send the remainder of the call information (if any) in one or more INFORMATION messages, starting timer T304 when each INFORMATION message is sent.

The called party number information is provided in the Called party number information element.

The call address information may contain a *sending complete* indication (e.g. No. or, as a network option, the Sending complete information element) appropriate to the dialling plan being used.

NOTE 1 – If the network can determine that sufficient call set-up information will be received by the called user by sending the next INFORMATION message, it is recommended that the INFORMATION message contains the Sending complete information element.

The user shall start timer T302 on receipt of every INFORMATION message not containing a sending complete indication.

Following the receipt of a sending complete indication which the user understands, or the determination that sufficient call information has been received, the user shall stop timer T302 (if implemented) and send a CALL PROCEEDING message to the network. Alternatively, depending on internal events, the user may send an ALERTING or a CONNECT message to the network.

NOTE 2 – The CALL PROCEEDING message in this case will cause the originating exchange to send a CALL PROCEEDING message to the originating user, if not already sent.

At the expiration of timer T302, the user shall:

a) initiate clearing in accordance with 5.3, with cause No. 28, *invalid number format (address incomplete)* if it determines that the call information is definitely incomplete; or

b) if sufficient information has been received, send a CALL PROCEEDING, ALERTING or CONNECT message as appropriate.

At the expiration of timer T304 the network initiates call clearing in accordance with 5.3, with cause No. 28, *invalid number format (address incomplete)*, sent to the calling user, and cause No. 102, *recovery on timers expiry*, sent to the called user.

If, following the receipt of a SETUP message or during overlap receiving, the user determines that the received call information is invalid (e.g. invalid called party number), it shall initiate call clearing in accordance with 5.3 with a cause such as one of the following:

No. 1 – *Unallocated (unassigned) number*;

No. 3 – *No route to destination*;

No. 22 – *Number changed*;

No. 28 – *Invalid number format (incomplete number)*.

Upon receipt of the complete call information, the user may further perform some compatibility checking functions, as outlined in Annex B.

When the call is offered on a point-to-point data link, only one SETUP ACKNOWLEDGE message can be received in response to the call offering.

When the call is offered to the user on a broadcast data link, multiple SETUP ACKNOWLEDGE messages may be received by the network which shall then complete as many overlap receiving procedures as these SETUP ACKNOWLEDGE messages are received. It is the network responsibility to limit the number of overlap receiving procedures to be completed for a given call. The default maximum is fixed to eight. Some networks will limit the call offering completion in overlap receiving to single data link and will therefore clear the subsequent responding users after the first SETUP ACKNOWLEDGE message has been received, in accordance with the non-selected user clearing procedures described in 5.2.9.

5.2.5 Call confirmation

5.2.5.1 Response to *en bloc* SETUP or completion of overlap receiving

When the user determines that sufficient call set-up information has been received and compatibility requirements (see Annex B) have been satisfied, the user responds with either a CALL PROCEEDING, ALERTING, or CONNECT message (see Note), and enters the Incoming Call Proceeding, Call Received, or Connect Request state respectively.

NOTE – A Progress indicator information element may be included in CALL PROCEEDING, ALERTING and CONNECT messages (e.g. when an analogue terminal is connected to an ISDN PABX). The CALL PROCEEDING message may be sent by the user which cannot respond to a SETUP message with an ALERTING, CONNECT, or RELEASE COMPLETE message before expiration of timer T303.

When the SETUP message was delivered via a broadcast data link, an incompatible user shall either:

a) ignore the incoming call; or

b) respond by sending a RELEASE COMPLETE message with cause No. 88, *incompatible destination*, and enter the Null state. The network processes this RELEASE COMPLETE message in accordance with 5.2.5.3.

When the SETUP message was delivered via a point-to-point data link, an incompatible user shall respond with a RELEASE COMPLETE message with cause No. 88, *incompatible destination*. The network processes this RELEASE COMPLETE message in accordance with 5.2.5.3.

A busy user which satisfies the compatibility requirements indicated in the SETUP message shall respond with a RELEASE COMPLETE message with cause No. 17, *user busy*. The network processes this RELEASE COMPLETE message in accordance with 5.2.5.3.

If the user wishes to refuse the call, a RELEASE COMPLETE message shall be sent with the cause No. 21, *call rejected*, and the user returns to the Null state. The network processes this RELEASE COMPLETE message in accordance with 5.2.5.3.

5.2.5.2 Receipt of CALL PROCEEDING and ALERTING

When the SETUP message is delivered on a broadcast data link, the network shall maintain a state machine that tracks the overall progression of the incoming call. The network shall also maintain an associated call state for each responding user as determined by the data link on which a message is received.

Upon receipt of the first CALL PROCEEDING message from a user (assuming no other user had previously responded with an ALERTING or CONNECT message when the SETUP message has been delivered on a broadcast data link), the network shall stop timer T303 (or, in the case of overlap receiving, timer T304 for that user); start timer T310, and enter the incoming call proceeding state, Timer T310 shall not be restarted upon receipt of subsequent CALL PROCEEDING messages.

When the SETUP message has been delivered on a broadcast data link, the network shall (at a minimum) associate the incoming call proceeding state with each called user that sends a CALL

PROCEEDING message as a first response to the broadcast SETUP message prior to expiration of timer T312. Actions to be taken when a user sends a first response to an incoming call after the expiration of timer T312 are described in 5.2.5.4.

Upon receipt of the first ALERTING message from a user (assuming no other user has previously responded with a CONNECT message when the SETUP message has been delivered on a broadcast data link), the network shall stop timer T304 for that user (in the case of overlap receiving); stop timer T303 or T310 (if running); start timer T301 (unless another internal alerting supervision timer function exists, e.g. incorporated in call control); enter the call received state, and send a corresponding ALERTING message to the calling user. Timer T301 shall not be restarted upon receipt of subsequent CALL PROCEEDING messages.

When the SETUP message has been delivered on a broadcast data link, the network shall (at a minimum) associate the call received state with each called user that sends an ALERTING message either as a first response to the broadcast SETUP message or following a CALL PROCEEDING message.

5.2.5.3 Called user clearing during incoming call establishment

If the SETUP message has been delivered on a point-to-point data link and a RELEASE COMPLETE or DISCONNECT message is received before a CONNECT message has been received, the network shall stop timer T303, T304, T310 or T301 (if running); continue to clear the user as described in 5.3.3, and clear the call to the calling user with the cause received in the RELEASE COMPLETE or DISCONNECT message.

If the SETUP message has been delivered on a broadcast data link and a RELEASE COMPLETE message is received whilst timer T303 is running, the cause value received in the RELEASE COMPLETE message shall be retained by the network. If timer T303 expires (i.e. if no valid message such as CALL PROCEEDING, ALERTING or CONNECT has been received), the cause previously retained when a RELEASE COMPLETE message was received is sent back to the calling user in a DISCONNECT message and the network shall enter the Call Abort state. When multiple RELEASE COMPLETE messages are received with different causes, the network shall:

1) ignore any cause No. 88, *incompatible destination*; and

2) give preference to the following causes (if received) in the order listed below:

   (highest) No. 17 *user busy;*

   No. 21 *call rejected;*

3) any other received cause may also be included in the clearing message sent to the originating user (see 5.3).

If the SETUP message has been delivered on a broadcast data link and a user which has previously sent a SETUP ACKNOWLEDGE, CALL PROCEEDING or ALERTING message sends a DISCONNECT message to the network, the actions taken by the network depend on whether timer T312 is running and whether other called users have responded to the SETUP message.

Case 1 – DISCONNECT received prior to expiry of timer T312

If timer T312 is running and the network receives a DISCONNECT message after having received a SETUP ACKNOWLEDGE, CALL PROCEEDING or ALERTING message from a called user (but before receiving a CONNECT message), timer T312, as well as timer T310 or T301 (if running), should continue to run. The network shall retain the cause in the DISCONNECT message and shall continue to clear the user as described in 5.3.3. The network shall stop timer T304 (if running) for this user.

Upon expiration of timer T312, if either:

a) no other users have responded to the incoming call; or

b) all users that have responded to the incoming call have been cleared or are in the process of being cleared,

the network shall stop timer T310 or T301 (if running) and shall clear the call to the calling user. If an ALERTING message has been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority): No. 21, *call rejected*; any other cause sent by a called user. If only SETUP ACKNOWLEDGE, or CALL PROCEEDING messages have been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority): No. 17, *user busy*; No. 21, *call rejected*; any other appropriate cause sent by a called user.

Case 2 – DISCONNECT received after expiry of timer T312

If timer T312 has expired and the network receives a DISCONNECT message from the called user after having received a SETUP ACKNOWLEDGE, CALL PROCEEDING or ALERTING message (but before receiving a CONNECT message), the network shall continue to clear the user as described in 5.3.3. The network shall stop timer T304 (if running) for this user.

If other called users have responded to the SETUP message with a SETUP ACKNOWLEDGE, CALL PROCEEDING or ALERTING message, and still have the opportunity to accept the call by sending a CONNECT message, the network shall retain the cause in the DISCONNECT message. The network shall continue to process the incoming call for the remaining responding users (T310 or T301, if running, shall continue to run).

If either:

a) no other users have responded to the incoming call; or

b) all users that have responded to the incoming call have been cleared or are in the process of being cleared,

the network shall stop timer T310 or T301 (if running) and shall clear the call to the calling user. If an ALERTING message has been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority): No. 21, *call rejected*; or any other cause sent by a called user. If only SETUP ACKNOWLEDGE, or CALL PROCEEDING message have been received, the cause sent to the calling user shall be a cause received from the called user, giving preference to (in order of priority): No. 17, *user busy*; No. 21, *call rejected*; any other appropriate cause sent by a called user.

5.2.5.4 Call failure

If the network does not receive any response to the retransmitted SETUP message prior to the expiration of timer T303, then the network shall initiate clearing procedures towards the calling user with cause No. 18, *no user responding*.

a) If the SETUP message was delivered by a broadcast data link, the network shall enter the Call Abort state.

b) If the SETUP message was delivered on a point-to-point data link, the network shall also initiate clearing procedures towards the called user in accordance with 5.3.4, using cause No. 102, *recovery on timer expiry*.

If the network receives a user's first response to SETUP when in the Call Abort state but before timer T312 has expired, the network shall initiate clearing to the called user as described in 5.3.2 b), except that the cause No. 102, *recovery on timer expiry*, shall be sent. If the network receives a message that

is a user's first response to an incoming call after timer T312 has expired, the network will interpret this message as a message received with an invalid call reference value, as described in 5.8.3.2.

If the network has received a CALL PROCEEDING message, but does not receive an ALERTING, CONNECT, or DISCONNECT message prior to the expiration of timer T310, then the network shall initiate clearing procedures towards the calling user with cause No. 18, *no user responding*, and initiate clearing procedures towards the called user.

1) If the SETUP message is delivered by a broadcast data link, the called user shall be cleared in accordance with 5.3.2 e), except that cause No. 102, *recovery on timer expiry,* shall be sent.

2) If the SETUP message was delivered on a point-to-point data link, the called user shall be cleared in accordance with 5.3.4 using cause No. 102, *recovery on timer expiry*.

If the network has received an ALERTING message, but does not receive a CONNECT or DISCONNECT message prior to the expiry of timer T301 (or a corresponding internal alerting supervision timing function), then the network shall initiate clearing procedures towards the calling user with cause No. 19, *No answer from user (user alerted)*, and initiate clearing procedures towards the called user.

i) If the SETUP message was delivered by a broadcast data link, the called user shall be cleared in accordance with 5.3.2 e), except that cause No. 102, *recovery on timer expiry,* shall be sent.

ii) If the SETUP message was delivered on a point-to-point data link, the called user shall be cleared in accordance with 5.3.4, using cause No. 102, *recovery on timer expiry*.

5.2.6 Notification of interworking at the terminating interface

During call establishment the call may enter an ISDN environment, e.g. because of interworking with another network, with a non-ISDN user, or with non-ISDN equipment within the calling or called user's premises. When this occurs, the point at which the call enters an ISDN environment shall cause a Progress indicator information element to be included in the SETUP message to be sent to the called user:

a) No. 1 – *Call is not end-to-end ISDN, further call progress information may be available in-band*;

NOTE – On receipt of progress indicator No. 1, the called user shall connect to the B-channel in accordance with the procedures of 5.2.8.

b) No. 3 – *Origination address is non-ISDN*.

In addition, the user shall notify the calling party if the call has left the ISDN environment within the called user's premises, or upon the availability of in-band information/patterns. When such situations occur, a progress indication shall be sent by the user to the network either:

a) in an appropriate call control message when a state change is required (SETUP ACKNOWLEDGE, CALL PROCEEDING, ALERTING, or CONNECT); or

b) in the PROGRESS message when no state change is appropriate.

One of the following progress description values shall be included in the Progress indicator information element in the message sent to the network (for further information, see Annex G):

i) No. 1 – *Call is not end-to-end ISDN, further call progress information may be available in-band*.

ii) No. 2 – *Destination address is non-ISDN*.

iii) No. 4 – *Call has returned to the ISDN*.

If the Progress indicator information element is included in a call control message, the procedures as described in the rest of 5.2 apply. If the Progress indicator information element is included in the PROGRESS message, no state change will occur and all supervisory timers running shall be continued.

NOTE – If progress description No. 8 is received, it has no impact on any supervisory timers and shall be ignored by the network except when the Annex K procedures are applied.

5.2.7 Call accept

A user indicates acceptance of an incoming call by sending a CONNECT message to the network. Upon sending the CONNECT message, the user shall start timer T313 (specified in 9.2) and enter the Connect Request state. If an ALERTING message had previously been sent to the network, the CONNECT message may contain only the call reference.

If a call can be accepted using the B-channel indicated in the SETUP message, and no user alerting is required, a CONNECT message may be sent without a previous ALERTING message.

5.2.8 Active indication

On receipt of the first CONNECT message, the network shall stop (if running) timers T301, T303, T304 and T310; complete the circuit-switched path to the selected B-channel; send a CONNECT ACKNOWLEDGE message to the user which first accepted the call; initiate procedures to send a CONNECT message towards the calling user and enter the active state.

The CONNECT ACKNOWLEDGE message indicates completion of the circuit-switched connection. There is no guarantee of an end-to-end connection until a CONNECT message is received at the calling user. Upon receipt of the CONNECT ACKNOWLEDGE message, the user shall stop timer T313 and enter the active state.

When timer T313 expires prior to receipt of a CONNECT ACKNOWLEDGE message, the user shall initiate clearing in accordance with 5.3.3.

A user which has received the SETUP via the broadcast data link, and has been awarded the call, shall connect to the B-channel only after it has received the CONNECT ACKNOWLEDGE message. Only the user that is awarded the call will receive the CONNECT ACKNOWLEDGE message.

A user which has received the SETUP via a point-to-point data link may connect to the B-channel as soon as channel selection has been completed.

5.2.9 Non-selected user clearing

In addition to sending the CONNECT ACKNOWLEDGE message to the user selected for the call, the network shall send RELEASE message [as described in 5.3.2 b)] to all other users at the interface that have sent SETUP ACKNOWLEDGE, CALL PROCEEDING, ALERTING or CONNECT messages in response to the SETUP message. These RELEASE messages are used to notify the users that the call is no longer offered to them. The procedures described in 5.3.4 are then followed. Any user which having previously sent a CONNECT message and started timer T313, and which subsequently receives a RELEASE message, shall stop timer T313 and follow the procedures of 5.3.4.

5.3 Call clearing

5.3.1 Terminology

The following terms are used in this Recommendation in the description of clearing procedures:

– A channel is *connected* when the channel is part of a circuit-switched ISDN connection established according to this Recommendation.

– A channel is *disconnected* when the channel is no longer part of a circuit-switched ISDN connection, but is not yet available for use in a new connection.

– A channel is *released* when the channel is not part of a circuit-switched ISDN connection and is available for use in a new connection. Similarly, a call reference that is *released* is available for reuse.

5.3.2 Exception conditions

Under normal conditions, call clearing is usually initiated when the user or the network sends a DISCONNECT message and follows the procedures defined in 5.3.3 and 5.3.4 respectively. The only exceptions to the above rule are as follows:

a) In response to a SETUP message, the user or network can reject a call (e.g. because of the unavailability of a suitable B-channel) by responding with a RELEASE COMPLETE message provided no other response has previously been sent (e.g. the SETUP ACKNOWLEDGE message in the case of overlap sending), by releasing the call reference and entering the Null state.

b) In the case of a multipoint terminal configuration, non-selected user call clearing will be initiated with RELEASE message(s) from the network (see 5.2.9). The RELEASE message shall contain cause No. 26, *non-selected user clearing*.

c) Clearing of temporary signalling connections will be initiated by sending a RELEASE message as described in 5.3.3 and 5.3.4.

d) Unsuccessful termination of the B-channel selection procedure (see 5.2.3.1 and 5.1.2) by the side offering the call is accomplished by sending a RELEASE message. The RELEASE message shall contain cause No. 6, *channel unacceptable*. The network and user shall subsequently follow the procedures of 5.3.3 and 5.3.4.

e1) In the case of a SETUP message sent via the broadcast data link, if a network disconnect indication is received during call establishment, and prior to the expiry of timer T312, timer T303 is stopped (if running) and the network enters the Call Abort state. Any user which has responded, or subsequently responds before timer T312 expires, will be cleared by a RELEASE message [with the cause code(s) contained in the network disconnect indication] and the procedures of 5.3.4 are then followed for that user. Upon expiry of timer T312, the network shall treat any subsequent responses according to the procedures defined in 5.8.3.2. The network shall enter the Null state upon completion of clearing procedures for all responding users.

e2) In the case of a SETUP message sent via the broadcast data link, if a network disconnect indication is received during call establishment after expiry of timer T312, any user which has responded shall be cleared by a RELEASE message [with the cause code(s) contained in the network disconnect indication] and the procedures of 5.3.4 are then followed for that user. The network enters the Null state upon completion of clearing procedures for all responding users.

NOTE – A separate state machine exists for each responding user.

f) When timer T318 expires, the user initiates call clearing by sending a RELEASE message with cause No. 102, *recovery on timer expiry* starting timer T308 and continuing as described in 5.3.3.

5.3.3 Clearing initiated by the user

Apart from the exceptions identified in 5.3.2 and 5.8, the user shall initiate clearing by sending a DISCONNECT message, starting timer T305 (the value of timer T305 is specified in 9.2), disconnecting the B-channel and entering the disconnect request state.

NOTE 1 – When a user initiates call clearing by sending a RELEASE message, the procedures described in 5.3.4 are then followed.

The network shall enter the Disconnect Request state upon receipt of a DISCONNECT message. This message then prompts the network to disconnect the B-channel, and to initiate procedures for clearing the network connection to the remote user. Once the B-channel used for the call has been disconnected, the network shall send a RELEASE message to the user; start timer T308 (the value of a timer T.308 is specified in 9.1) and enter the Release Request state.

NOTE 2 – The RELEASE message has only local significance and does not imply an acknowledgement of clearing from the remote user.

On receipt of the RELEASE message the user shall cancel timer T305; release the B-channel, send a RELEASE COMPLETE message, release the call reference and return to the Null state. Following the receipt of a RELEASE COMPLETE message from the user, the network shall stop timer T308; release both the B-channel and the call reference and return to the Null state.

If timer T305 expires, the user shall send a RELEASE message to the network with the cause number originally contained in the DISCONNECT message; start timer T308 and enter the Release Request state. In addition, the user may indicate a second Cause information element with cause No. 102, *recovery on timer expiry*.

If timer T308 expires for the first time, the network shall retransmit the RELEASE message and timer T308 shall be restarted. In addition, the network may indicate a second Cause information element with cause No. 102, *recovery on timer expiry*. If no RELEASE COMPLETE message is received from the user before timer T308 expires a second time, the network shall place the B-channel in a maintenance condition, release the call reference and return to the Null state.

NOTE 3 – The restart procedures contained in 5.5 may be used on B-channels in the maintenance condition.

NOTE 4 – Other actions which could be taken by the network upon receipt of a DISCONNECT message are for further study.

The actions to be taken with regard to the maintenance condition are network dependent.

5.3.4 Clearing initiated by the network

Apart from the exception conditions identified in 5.3.2 and 5.8, the network shall initiate clearing by sending a DISCONNECT message, and entering the Disconnect Indication state. The DISCONNECT message is a local invitation to clear and does not imply that the B-channel has been disconnected at the user-network interface.

NOTE – When the network initiates clearing by sending a RELEASE message, the procedures described in 5.3.3 are followed.

5.3.4.1 Clearing when tones/announcements provided

When in-band tones/announcements are provided (see 5.4), the DISCONNECT message contains progress indicator No. 8, *in-band information or appropriate pattern is now available*. The network shall: start timer T306 and enter the Disconnect Indication state.

On receipt of the DISCONNECT message with progress indicator No. 8, the user may connect (if not already connected) to the B-channel to receive the in-band tone/announcement; and enter the disconnect indication state. Alternatively, to continue clearing without connecting to the in-band tone/announcement, the user shall disconnect the B-channel; send a RELEASE message, start timer T308 and enter the Release Request state.

If the user connects to the provided in-band tone/announcement, the user may subsequently continue clearing (before the receipt of a RELEASE from the network) by disconnecting from the B-channel, sending a RELEASE message, starting timer T308 and entering the Release Request state.

On receipt of the RELEASE message, the network shall stop timer T306; disconnect and release the B-channel; send a RELEASE COMPLETE message; release the call reference and return to the Null state.

If timer T306 expires, the network shall continue clearing by disconnecting the B-channel, sending a RELEASE message with the cause number originally contained in the DISCONNECT message, starting timer T308 and entering the Release Request state.

In addition to the original clearing cause, the RELEASE message may contain a second cause information element with cause No. 102, *recovery on timer expiry*; this cause may optionally contain a diagnostic field identifying the timer that expired.

On receipt of the RELEASE message, the user shall act according to 5.3.3.

5.3.4.2 Clearing when tones/announcements not provided

When in-band tones/announcements are *not* provided, the DISCONNECT message does *not* contain progress indicator No. 8, *in-band information or appropriate pattern is now available*. The network shall initiate clearing by sending the DISCONNECT message; start timer T305; disconnects the B-channel and enters the Disconnect Indication state.

On the receipt of the DISCONNECT message without progress indicator No. 8, the user shall disconnect the B-channel; send a RELEASE message; start timer T308 and enter the Release Request state.

On receipt of the RELEASE message, the network shall stop timer T305; release the B-channel; send a RELEASE COMPLETE message; release the call reference and return to the Null state.

If timer T305 expires, the network shall send a RELEASE message to the user with the cause number originally contained in the DISCONNECT message; start timer T308 and enter the Release Request state. In addition to the original clearing cause, the RELEASE message may contain a second Cause information element with cause No. 102, *recovery on timer expiry*.

5.3.4.3 Completion of clearing

Following the receipt of a RELEASE COMPLETE message from the network, the user shall stop timer T308; release both the B-channel and the call reference and return to the Null state.

If a RELEASE COMPLETE is not received by the user before the first expiry of timer T308, the RELEASE message shall be retransmitted and timer T308 shall be restarted. If no RELEASE COMPLETE message is received from the network before timer T308 expires a second time, the user may place the B-channel in a maintenance condition, shall release the call reference and return to the Null state.

NOTE – The restart procedures contained in 5.5 may be used on B-channels in the maintenance condition.

5.3.5 Clear collision

Clear collision occurs when the user and the network simultaneously transfer DISCONNECT messages specifying the same call reference value. When the network receives a DISCONNECT message whilst in the Disconnect Indication state, the network shall stop timer T305 or T306 (whichever is running); disconnect the B-channel (if not disconnected); send a RELEASE message; start timer T308 and enter the Release Request state. Similarly, when the user receives a DISCONNECT message whilst in the Disconnect Request state, the user shall stop timer T305; send a RELEASE message; start timer T308 and enter the Release Request state.

Clear collision can also occur when both sides simultaneously transfer RELEASE messages related to the same call reference value. The entity receiving such a RELEASE message whilst within the Release Request state shall stop timer T308; release the call reference and B-channel and enter the Null state (without sending or receiving a RELEASE COMPLETE message).

5.4 In-band tones and announcements

When in-band tones/announcements not associated with a call state change are to be provided by the network before reaching the Active state, a PROGRESS message is returned simultaneously with the application of the in-band tone/announcement. The PROGRESS message contains the progress indicator No. 8, *in-band information or appropriate pattern is now available*.

When tones/announcements have to be provided together with a call state change, then the appropriate message [e.g. for ALERTING, DISCONNECT, etc., (see clause 3)] with progress indicator No. 8, *in-band information or appropriate pattern is now available*, is sent simultaneously with the application of the in-band tone/announcement.

NOTE 1 – When the network provides ITU-T standardized telecommunications services, the service requirement for provision of in-band tones/announcements is as indicated in the I.200-series Recommendations.

NOTE 2 – When the PROGRESS message is used, the user may initiate call clearing as a result of the applied in-band tone/announcement, according to the procedures specified in 5.3.3.

5.5 Restart procedure

The restart procedure is used to return calls to the Null state or the interface to an idle condition. The procedure is usually invoked when the other side of the interface does not respond to other call control messages or a failure has occurred (e.g. following a data link failure, when a backup D-channel can be used, or following the expiry of timer T308 due to the absence of response to a clearing message). It may also be initiated as a result of local failure, maintenance action or mis-operation.

NOTE 1 – Layer 3 procedures and resources associated with those data links with SAPI = "0000 000" should be initialized by the restart procedures.

NOTE 2 – The call reference flag of the global call reference applies to restart procedures. In the case when both sides of the interface initiate simultaneous restart requests, they shall be handled independently. In the case when the same channel(s) or interface(s) are specified, they shall not be considered free for reuse until all the relevant restart procedures are completed.

When:

a) both the user and the network are aware of the configuration of the interface; and

b) the interface is a basic access (Recommendation I.431 [27]) where a point-to-point configuration exists; or

c) the interface is a primary rate access (Recommendation I.430 [46]),

then the user and the network shall implement the procedures of 5.5. In all other cases, the procedures of 5.5 are optional.

5.5.1 Sending RESTART message

A RESTART message is sent by the network or user equipment in order to return channels or interfaces to the Null state. The Restart indicator information element shall be present in the RESTART message to indicate whether an *Indicated channel*, *Single interface* or *All interfaces* are to be restarted. If the Restart indicator information element is coded as "Indicated channel", or "Single interface" and the interface is one other than the one containing the D-channel, then the Channel identification information element shall be present to indicate which channel or interface is to be returned to the idle condition. If the Restart indicator information element is coded as "Single interface" and the interface is the one containing the D-channel, then the Channel identification information element may be omitted. If the Restart indicator information element is coded as "All interfaces", then the Channel identification information element shall not be included.

Upon transmitting the RESTART message the sender enters the Restart Request state, starts timer T316 and waits for a RESTART ACKNOWLEDGE message. Also, no further RESTART messages shall be sent until a RESTART ACKNOWLEDGE is received or timer T316 expires. Receipt of a RESTART ACKNOWLEDGE message stops timer T316, frees the channels and call reference values for reuse and enters the Null state.

If a RESTART ACKNOWLEDGE message is not received prior to the expiry of timer T316, one or more subsequent RESTART messages may be sent until a RESTART ACKNOWLEDGE message is returned. Meanwhile, no calls shall be placed or accepted over the channel or interface by the originator of the RESTART message. A network shall limit the number of consecutive unsuccessful restart attempts to a default limit of two. When this limit is reached, the network shall make no further restart attempts. An indication will be provided to the appropriate maintenance entity. The channel or interface is considered to be in an out-of-service condition until maintenance action has been taken.

NOTE – If a RESTART ACKNOWLEDGE message is received indicating only a subset of the specified channels, an indication shall be given to the maintenance entity. It is the responsibility of the maintenance entity to determine what actions shall be taken on the channel(s) which have not been returned to the idle condition.

The RESTART and RESTART ACKNOWLEDGE message shall contain the global call reference value (all zeros) to which the Restart Request state is associated. These messages are transferred via the appropriate point-to-point data link in the multiple frame mode using the DL-DATA request primitive.

5.5.2 Receipt of RESTART message

Upon receiving a RESTART message, the recipient shall enter the Restart state associated to the global call reference and start timer T317; it shall then initiate the appropriate internal actions to return the specified channels to the idle condition and call references to the Null state. Upon completion of internal clearing, timer T317 shall be stopped and a RESTART ACKNOWLEDGE message transmitted to the originator and the Null state entered.

NOTE 1 – If only a subset of the specified channels have been returned to the idle condition when timer T317 expires, a RESTART ACKNOWLEDGE message should be transmitted to the originator, containing a Channel identification information element indicating the channel(s) that have been returned to the idle condition.

If timer T317 expires prior to completion of internal clearing, an indication shall be sent to the maintenance entity (i.e. a primitive should be transmitted to the system management entity).

Even if all call references are in the Null state, and all channels are in the idle condition, the receiving entity shall transmit a RESTART ACKNOWLEDGE message to the originator upon receiving a RESTART message.

If the Restart indicator information element is coded as "all interfaces", then all calls on all interfaces associated with the D-channel shall be cleared. If the Restart indicator information element is coded as "all interfaces" and a Channel identification information element is included, the Channel identification information element is treated as described in 5.8.7.3.

If the Restart indicator information element is coded as "indicated channel" and the Channel indication information element is not included, then the procedures in 5.8.6.1 shall be followed.

If the Restart indicator information element is coded as "single interface" and that interface includes the D-channel, then only those calls associated with the D-channel on that interface shall be cleared.

The receiving DSS1 protocol control entity for the global call reference shall indicate a restart request to only those DSS1 protocol control entities for specific call references which:

a) are supported by the same Data Link Connection Endpoint Identifier (DLCI) (see Recommendation Q.920) as the DSS1 protocol control entity for the global call reference which received the RESTART message; and

b) correspond to the specified channel(s) or interface(s), or (if the D-channel was implicitly specified) are not associated with any channel, including calls in the call establishment phase for which a channel has not yet been allocated.

The following entities shall be released:

a) B- and H-channels established by Q.931 messages including channels used for packet access (case B) [consequently all virtual calls carried in the released channel(s) will be handled as described in 6.4.1];

b) user signalling bearer service connections;

c) other resources associated with a call reference, where specified in other DSS1 Recommendations.

NOTE 2 – Application to the Register procedures in Recommendation Q.932 requires further study.

The following entities shall not be released:

a) semi-permanent connections that are established by man-machine commands;

b) calls associated with DSS1 protocol control entities supported by any DLCI other than the one supporting the DSS1 protocol entity for the global call reference which received the RESTART message;

c) X.25 virtual calls and permanent virtual circuits using SAPI = 16;

d) TID and USID values established using terminal initialization procedures (see Annex A/Q.932).

If semi-permanent connections established by man-machine command are implicitly specified (by specifying "single interface or "all interfaces"), no action shall be taken on these channels, but a RESTART ACKNOWLEDGE message shall be returned containing the appropriate indications (i.e. "single interface" or "all interfaces").

If semi-permanent connections established by man-machine command are explicitly specified (by including a Channel identification information element in the RESTART message), no action shall be taken on these channels and a STATUS message should be returned with cause No. 82, *identified channel does not exist*, optionally indicating in the diagnostic field the channel(s) that could not be handled.

5.6 Call rearrangements

The elements of procedure in this subclause provide for physical layer and/or data link layer rearrangements after a call has entered the Active state as defined in 2.2.1.8. The procedure is restricted to use on the same interface structure, and resumption on the same B-channel. The use of call rearrangement procedure is restricted to basic access, i.e. it will not be available for primary rate access. For call rearrangements controlled by an NT2, see 5.6.7.

The activation of this procedure at a user-network interface may correspond to a number of possible events such as the following:

a) physical disconnection of user equipment and reconnection;

b) physical replacement of one user equipment by another;

c) the human user moves from one equipment to another;

d) suspension of call and its subsequent reactivation at the same user equipment.

These procedures have only local significance, i.e. the invocation of call rearrangement affects only states at the originating end, and it does not affect any terminating states.

The procedures in this subclause are described in terms of functional messages and information elements.

If the procedures for call suspension in this subclause are not followed prior to the physical disconnection of the terminal from the interface, then the integrity of the call cannot be guaranteed by the network.

5.6.1 Call suspension

The procedure is initiated by the user, who shall send a SUSPEND message containing the current call reference; start timer T319 and enter the Suspend Request state. The user may optionally include in this message a bit sequence (e.g. IA5 characters) to be known by the application or human user, and by the network, as the call identity for subsequent reconnection. Where no call identity information is included by the user (e.g. the Call identity information element is absent or empty), the network shall store this fact so that resumption is possible only by a procedure conveying no call identity information.

NOTE – If the Call identity information element is present with a null length, the message shall be handled as if it was absent.

The default maximum length of the call identity value within the Call identity information element is eight octets. If the network receives a call identity value longer than the maximum length supported, the network shall truncate the call identity value to the maximum length; take the action specified in 5.8.7 and continue processing.

5.6.2 Call suspended

Following the receipt of a SUSPEND message, the network enters the Suspend Request state. After a positive validation of the received call identity, the network shall send a SUSPEND ACKNOWLEDGE message and start timer T307. (The value of T307 is specified in 9.1.)

At this time, the network shall consider the call reference to be released and enter the Null state for that call reference. The call identity associated with the suspended call has to be stored by the network and cannot be accepted for another suspension until it is released.

The B-channel involved in the connection will be reserved by the network until reconnection of the call (or until a clearing cause occurs, e.g. expiry of timer T307). A NOTIFY message with notification indicator No. 0, user suspended is sent to the other user.

When the user receives the SUSPEND ACKNOWLEDGE message, the user shall stop timer T319, release the B-channel and call reference and enter the Null state.

Following the receipt of the SUSPEND ACKNOWLEDGE message, the user may disconnect the underlying data link connection. In any case, if the user physically disconnects from the interface without having disconnected the data link connection, standard data link layer procedures are started by the network side of the data link layer supervision, resulting in the release of the data link layer connection.

5.6.3 Call suspend error

If the network does not support the call rearrangement procedures, it shall reject a SUSPEND message according to the error handling procedures of 5.8.4. If the network supports the call rearrangement procedures on a subscription basis, but the user does not subscribe to the service, the network shall reject a SUSPEND message with cause No. 50, *requested facility not subscribed*; the Cause information element shall not contain a diagnostic field under these circumstances.

On receipt of a SUSPEND message, the network will respond by sending a SUSPEND REJECT message with cause No. 84, *call identity in use*, if the information contained in the SUSPEND message is not sufficient to avoid ambiguities on subsequent call re-establishment. This will apply, in particular, when at a given user-network interface, a SUSPEND message is received with a call identity sequence already in use, or when the SUSPEND message does not contain any call identity sequence and the null-value call identity is already allocated for that interface. On receipt of the SUSPEND REJECT message, the user shall stop timer T319 and return to the active state. If timer T319 expires, the user shall notify the user application and return to the Active state.

In these cases, the state of the call is not altered within the network (i.e. it remains in the Active state).

5.6.4 Call re-establishment

At the connection end where suspension was initiated, the user may request re-establishment of a call after physical reconnection of a terminal by sending a RESUME message containing the call identity exactly as that used at the time of call suspension, starting timer T318 and entering the Resume Request state. If the SUSPEND message did not include a Call identity information element, then the corresponding RESUME message shall also not include a Call identity information element. The call reference included in the RESUME message is chosen by the user according to the normal allocation of outgoing call reference (see 4.3).

On receipt of a RESUME message, the network enters the Resume Request state. After a positive validation of the call identity that relates to the suspended call containing a valid identity that relates to a currently suspended call, the network shall send a RESUME ACKNOWLEDGE message to the user, release the call identity, stop timer T307 and enter the Active state. The RESUME ACKNOWLEDGE message shall specify the B-channel reserved to the call by the network by means of the channel identification element, coded *B-channel is indicated, no alternative is acceptable*.

The network shall also send a NOTIFY message with the notification indicated *user resumed* to the other user.

No memory of the previously received call identity sequence is kept by the network after sending the RESUME ACKNOWLEDGE message. This call identity is now available for another suspension.

On receipt of the RESUME ACKNOWLEDGE message, the user shall stop timer T318 and enter the Active state.

No compatibility is performed during the call arrangement phase.

5.6.5 Call resume errors

If the network does not support the call rearrangement procedures, it shall reject a RESUME message according to the error handling procedures of 5.8.3.2 a). For this purpose, the RESUME message would be deemed to be an unrecognized message.

If a received RESUME message cannot be actioned by the network (e.g. as a result of an unknown call identity a RESUME REJECT message shall be returned to the requesting user indicating one of the following causes:

a) No. 83 – *A suspended call exists, but this call identity does not*;

b) No. 85 – *No call suspended*; or

c) No. 86 – *Call having the requested call identity has been cleared*.

The call identity remains unknown. The call reference contained in the RESUME message is released by both the user and network side. Upon receipt of the RESUME REJECT message, the user shall stop timer T318 and enter the Null state.

If timer T307 expires the network shall initiate clearing of the network connection with cause No. 102, *recovery on timer expiry*, discard the call identity and release the reserved B-channel.

On release, the call identity can then be used for subsequent call suspension. If before the expiry of timer T307 the call is cleared by the remote user, the B-channel reservation is released but the call identity may be preserved by some networks along with a clearing cause (e.g. cause No. 16, *normal call clearing*).

If timer T318 expires, the user shall initiate call clearing in accordance with 5.3.2 f).

5.6.6 Double suspension

Simultaneous suspension of the call at both ends is possible. The procedures do not prevent this from occurring. If double suspensions are not desired the users must protect against this by other means, e.g. higher layer negotiation protocols.

5.6.7 Call rearrangement notification controlled by an NT2

When the call rearrangement is controlled by the NT2, the procedures shall be applied by the NT2 at reference point S. The NT2 shall inform the remote user by sending a NOTIFY message described in 5.6.2 and 5.6.4 across reference point T.

5.7 Call collisions

Call collisions as such cannot occur at the network. Any simultaneous incoming or outgoing calls are dealt with separately and assigned different call references.

Channel selection conflicts may occur if an incoming call and outgoing call select the same channel. This is resolved by the network through channel selection mechanisms described in 5.1.2 and 5.2.2.

In the case of such conflicts, the network shall give priority to the incoming call over the call request received from the user. It shall clear the outgoing call whenever the B-channel cannot be allocated by the network or accepted by the user originating the call.

NOTE – Some terminal adaptors supporting existing non-voice terminals (e.g. Recommendation X.21) may need to resolve double channel selection by clearing the incoming call and re-attempting the outgoing call setup in order to satisfy the requirements of the interface at reference point R.

5.8 Handling of error conditions

All procedures transferring signalling information by using the protocol discriminator of Q.931 user-network call control messages are applicable only to those messages which pass the checks described in 5.8.1 through 5.8.7. The error handling procedures of 5.8.1 through 5.8.7 apply to messages using an ordinary call reference or the global call reference, except where otherwise noted.

Detailed error handling procedures are implementation dependent and may vary from network-to-network. However, capabilities facilitating the orderly treatment of error conditions are provided for in this subclause and shall be provided in each implementation.

Subclauses 5.8.1 through 5.8.7 are listed in order of precedence.

5.8.1 Protocol discrimination error

When a message is received with a protocol discriminator coded other than *Q.931 user-network call control message*, that message shall be ignored. Ignore means to do nothing, as if the message had never been received.

5.8.2 Message too short

When a message is received that is too short to contain a complete message type information element, that message shall be ignored.

5.8.3 Call reference error

5.8.3.1 Invalid Call reference format

If the Call reference information element octet 1, bits 5 through 8 do not equal 0000, then the message shall be ignored.

If the Call reference information element octet 1, bits 1 through 4 indicate a length greater than the maximum length supported by the receiving equipment (see 4.3), then the message shall be ignored.

When a message is received with the dummy call reference, it shall be ignored unless it is required for a supplementary service (see Recommendation Q.932 [4]).

5.8.3.2 Call reference procedural errors

Only item f) applies to messages using the global call reference.

a) Whenever any message except SETUP, RELEASE, RELEASE COMPLETE, STATUS, STATUS ENQUIRY or (for networks supporting the call rearrangement procedures of 5.6) RESUME is received specifying a call reference which it does not recognize as relating to an active call or a call in progress, clearing is initiated by sending a RELEASE message with cause No. 81, *invalid call reference value*, and following the procedures in 5.3, specifying the call reference in the received message.

   Alternatively, the receiving entity may send a RELEASE COMPLETE message with cause No. 81, *invalid call reference value,* and remain in the Null state.

b) When a RELEASE message is received that specified a call reference which is not recognized as relating to an active call or a call in progress, a RELEASE COMPLETE message with cause No. 81, *invalid call reference value*, is returned, specifying the call reference in the received message.

c) When a RELEASE COMPLETE message is received specifying a call reference which it does not recognize as relating to an active call or a call in progress, no action should be taken.

d) When a SETUP or RESUME message is received specifying a call reference with a call reference flag incorrectly set to "1", that message shall be ignored.

e) When a SETUP message is received specifying a call reference which is recognized as relating to an active call, or a call in progress, this SETUP message shall be ignored.

f) When any message except RESTART, RESTART ACKNOWLEDGE, or STATUS is received using the global call reference, no action should be taken on this message and a STATUS message using the global call reference with a call state indicating the current state associated with the global call reference and cause No. 81, *invalid call reference value*, shall be returned.

g) When a STATUS message is received specifying a call reference which is not recognized as relating to an active call or to a call in progress, the procedures of 5.8.11 shall apply.

h) When a STATUS ENQUIRY message is received specifying a call reference which is not recognized as relating to an active call or to a call in progress, the procedures of 5.8.10 shall apply.

NOTE – Some implementations conforming to Recommendation Q.931 (1988) may choose to initiate clearing by sending a RELEASE message with cause No. 81, *invalid call reference value*, and continue to follow the procedures in 5.3, specifying the call reference in the received message, or respond with a RELEASE COMPLETE message with cause No. 81, *invalid call reference value*, and remain in the Null state.

i) When a RESTART message is received specifying a global call reference with a call reference flag set to "1", that message shall be ignored.

5.8.4 Message type or message sequence errors

Whenever an unexpected message, except RELEASE or RELEASE COMPLETE, or unrecognized message is received in any state other than the Null state, a STATUS message shall be returned with cause No. 98, *message not compatible with call state or message type non-existent or not implemented*, and the corresponding diagnostic. If a network or user can distinguish between unimplemented (or non-existent) message types and implemented message types which are incompatible with the call state, then a STATUS message may be sent with one of the following causes:

a) No. 97 – *Message type non-existent or not implemented*; or

b) No. 101 – *Message not compatible with call state*.

Alternatively, a STATUS ENQUIRY message may be sent requesting the call state of the entity (see 5.8.10). No change in state shall be made in either case at this time. This alternative is not applicable to messages using the global call reference.

However, two exceptions to this procedure exist. The first exception is when the network or the user receives an unexpected RELEASE message (e.g. if the DISCONNECT message was corrupted by undetected transmission errors). In this case, no STATUS or STATUS ENQUIRY message is sent. Whenever the network receives an unexpected RELEASE message, the network shall disconnect and release the B-channel; clear the network connection and the call to the remote user with the cause in the RELEASE message sent by the user or, if not included, cause No. 31, *normal, unspecified*, return a RELEASE COMPLETE message to the user; release the call reference, stop all timers and enter the Null state. Whenever the user receives an unexpected RELEASE message, the user shall disconnect and release the B-channel, return a RELEASE COMPLETE message to the network, release the call reference; stop all timers and enter the Null state.

The second exception is when the network or the user receives an unexpected RELEASE COMPLETE message. Whenever the network receives an unexpected RELEASE COMPLETE

message, the network shall disconnect and release the B-channel, clear the network connection and the call to the remote user with the cause indicated by the user or, if not included, cause No. 111, *protocol error, unspecified*; release the call reference; stop all timers and enter the Null state. Whenever the user receives an unexpected RELEASE COMPLETE message, the user shall disconnect and release the B-channel; release the call reference; stop all timers and enter the Null state.

5.8.5 General information element errors

The general information element error procedures may also apply to information elements in codesets other than 0. In that case, the diagnostics in the cause information element may indicate information elements other than those in codeset 0 by applying the locking or non-locking shift procedures as described in 4.5.

5.8.5.1 Information element out of sequence

A variable length information element which has a code value lower than the code value of the variable length information element preceding it shall be considered as an out of sequence information element.

If the network or user receives a message containing an out of sequence information element, it may ignore this information element and continue to process the message. If this information is mandatory, and the network or user chooses to ignore this out of sequence information element, then the error handling procedure for missing mandatory information elements as described in 5.8.6.1 shall be followed. If the ignored information element is non-mandatory, the receiver continues to process the message.

NOTE – Some implementations may choose to process all the information elements received in a message regardless of the order in which they are placed.

5.8.5.2 Duplicated information elements

If an information element is repeated in a message in which repetition of the information element is not permitted, only the contents of information element appearing first shall be handled and all subsequent repetitions of the information element shall be ignored. When repetition of information elements is permitted, only the contents of permitted information elements shall be handled. If the limit on repetition of information elements is exceeded, the contents of information elements appearing first up to the limit of repetitions shall be handled and all subsequent repetitions of the information element shall be ignored.

5.8.6 Mandatory information element errors

5.8.6.1 Mandatory information element missing

When a message other than SETUP, DISCONNECT, RELEASE or RELEASE COMPLETE is received which has one or more mandatory information elements missing, no action should be taken on the message and no state change should occur. A STATUS message is then returned with cause No. 96, *mandatory information element is missing*.

When a SETUP or RELEASE message is received which has one or more mandatory information elements missing, a RELEASE COMPLETE message with cause No. 96, *mandatory information element is missing*, shall be returned.

When a DISCONNECT message is received with the cause information element missing, the actions taken shall be the same as if a DISCONNECT message with cause No. 31, *normal, unspecified*, was received (see 5.3), with the exception that the RELEASE message sent on the local interface contains cause No. 96, *mandatory information element is missing*.

When a RELEASE COMPLETE message is received with a cause information element missing, it will be assumed that a RELEASE COMPLETE message was received with cause No. 31, *normal, unspecified*.

Information elements with a length indication of zero shall be treated as a missing information element.

5.8.6.2 Mandatory information element content error

If the Bearer capability information element is coded as circuit-mode, and the network cannot interpret octets 5b, 5c, 5d, 6 and 7, the network may accept these octets without declaring a protocol error and pass these octets on without change.

When a message other than SETUP, DISCONNECT, RELEASE, or RELEASE COMPLETE is received which has one or more mandatory information elements with invalid content, no action should be taken on the message and no state change should occur. A STATUS message is then returned with cause No. 100, *invalid information element contents*.

When a SETUP or RELEASE message is received which has one or more mandatory information elements with invalid content, a RELEASE COMPLETE message with cause No. 100, *invalid information element contents*, shall be returned.

When a DISCONNECT message is received with invalid content of the Cause information element, the actions shall be the same as if a DISCONNECT message with cause No. 31, *normal, unspecified*, was received (see 5.3), with the exception that the RELEASE message sent on the local interface contains cause No. 100, *invalid information element contents*.

When a RELEASE COMPLETE message is received with invalid content of the Cause information element, it will be assumed that a RELEASE COMPLETE message was received with cause No. 31, *normal, unspecified*.

Information elements with a length exceeding the maximum length (given in [3]) shall be treated as an information element with content error.

NOTE – As an option of the user equipment (e.g. NT2), cause values, location codes, and diagnostics which are not understood by the NT2 may be passed on to another entity (e.g. user or NT2) instead of treating the cause value as if it were cause No. 31, *normal, unspecified*, and sending cause No. 100, *invalid information element contents*, with the RELEASE message. This option is intended to aid user equipment to be compatible with future additions of cause values, location codes and diagnostics to the Recommendation.

5.8.7 Non-mandatory information element errors

The following subclauses identify actions on information elements not recognized as mandatory.

5.8.7.1 Unrecognized information element

When a message is received which has one or more unrecognized information elements, the receiving entity shall check whether any are encoded to indicate "comprehension required" (refer to Table 4-3 for information element identifiers reserved with this meaning). If any unrecognized information element is encoded to indicate "comprehension required", then the procedures in 5.8.6.1 are followed, i.e. as if a "missing mandatory information element" error condition had occurred. If all unrecognized information elements are **not** encoded to indicate "comprehension required", then the receiving entity shall proceed as follows:

Action shall be taken on the message and those information elements which are recognized and have valid content. When the received message is other than DISCONNECT, RELEASE or RELEASE COMPLETE, a STATUS message may be returned containing one cause information element. The STATUS message indicates the call state of the receiver after taking action on the message. The

Cause information element shall contain cause No. 99, *information element/parameter non-existent or not implemented*, and the diagnostic field, if present, shall contain the information element identifier for each information element which was unrecognized.

Subsequent actions are determined by the sender of the unrecognized information elements. If a clearing message contains one or more unrecognized information elements, the error is reported to the local user in the following manner:

a) When a DISCONNECT message is received which has one or more unrecognized information elements, a RELEASE message with cause No. 99, *information element/parameter non-existent or not implemented*, shall be returned. The Cause information element diagnostic field, if present, shall contain the information element identifier for each information element which was unrecognized.

b) When a RELEASE message is received which has one or more unrecognized information elements, a RELEASE COMPLETE message with cause No. 99 *information element/parameter non-existent or not implemented*, shall be returned. The Cause information element diagnostic field, if present, shall contain the information element identifier for each information element which was unrecognized.

c) When a RELEASE COMPLETE message is received which has one or more unrecognized information elements, no action shall be taken on the unrecognized information.

NOTE – The diagnostic(s) of cause No. 99 facilitates the decision in selecting an appropriate recovery procedure at the reception of a STATUS message. Therefore, it is recommended to provide cause No. 99 with diagnostic(s) if a layer 3 entity expects the peer to take an appropriate action at the receipt of a STATUS message, although inclusion of diagnostic(s) is optional.

5.8.7.2 Non-mandatory information element content error

When a message is received which has one or more non-mandatory information elements with invalid content, action shall be taken on the message and those information elements which are recognized and have valid content. A STATUS message may be returned containing one Cause information element. The STATUS message indicates the call state of the receiver after taking action on the message. The Cause information element shall contain cause No. 100, *invalid information element contents*, and the diagnostic field, if present, shall contain the information element identifier for each information element which has invalid contents.

Information elements with a length exceeding the maximum length (given in [3]) will be treated as an information element with content error. But for access information elements (e.g. User-user information element, Called party subaddress information element), cause No. 43, *access information discarded*, is used instead of cause No. 100, *invalid information element contents*. However, in some networks, access information elements may be truncated and processed.

The Call identity information element shall have a special treatment and shall be truncated and processed in the case that it exceeds the maximum length implemented.

As an option of user equipment (e.g. NT2), cause values, location codes, and diagnostics which are not understood by the NT2 may be accepted, or in the case of an NT2, passed on to another entity (e.g. user or NT2) instead of ignoring the cause information element contents and optionally sending a STATUS message with cause No. 100, *invalid information element contents*. This option is intended to aid user equipment to be compatible with future additions of cause values, location codes and diagnostics to the Recommendation.

If the network cannot interpret the Low layer compatibility or the High layer compatibility information elements, it may accept these information elements without declaring a protocol error.

5.8.7.3 Unexpected recognized information element

When a message is received with a recognized information element that is not marked as comprehension required and is not defined to be contained in that message, the receiving entity shall (except as noted below) treat the information element as an unrecognized information element and follow the procedures defined in 5.8.7.1. When a message is received with a recognized information element that is marked as comprehension required and is not defined to be contained in that message, the receiving entity shall follow the procedures of 5.8.6.1.

NOTE – Some implementations may choose to process unexpected recognized information elements when the procedure for processing the information element is independent of the message in which it is received.

5.8.8 Data link reset

Whenever a Q.931 entity is informed of a spontaneous data link layer reset by means of the DL-ESTABLISH indication primitive, the following procedures apply:

a) For calls in the Overlap Sending and Overlap Receiving states, the entity shall initiate clearing by sending a DISCONNECT message with cause No. 41, *temporary failure*, and following the procedures of 5.3.

b) For calls in the disestablishment phase (states N11, N12, N19, N22, U11, U12 and U19), no action shall be taken.

c) Calls in the establishment phase (states N1, N3, N4, N6, N7, N8, N9, U1, U3, U4, U6, U7, U8 and U9) and in the Active, Suspend Request, and Resume Request states shall be maintained according to the procedures contained in other parts of clause 5.

5.8.9 Data link failure

Whenever the network layer entity is notified by its data link layer entity via the DL-RELEASE indication primitive that there is a data link layer malfunction, the following procedure shall apply:

a) Any calls not in the Active state shall be cleared internally.

b) For any call in the Active state, a timer T309 shall be started (if implemented).

   If timer T309 is already running, it shall not be restarted.

   The Q.931 entity shall request layer 2 re-establishment by sending a DL-ESTABLISH request primitive.

   When informed of layer 2 re-establishment by means of the DL-ESTABLISH confirmation primitive, the following procedure shall apply:

   the Q.931 entity shall stop timer T309, and either:

   – the Q.931 entity shall send a STATUS message with cause No. 31, *normal, unspecified*, to report the current state to the peer entity; or

   – the Q.931 entity shall perform the status enquiry procedure according to 5.8.10 to verify the call state of the peer entity.

   If timer T309 expires prior to data link re-establishment, the network shall clear the network connection and call to the remote user with cause No. 27, *destination out of order*; disconnect and release the B-channel; release the call reference and enter the Null state.

   If timer T309 expires prior to data link re-establishment, the user shall clear the attached connection (if any) with cause No. 27, *destination out of order*; disconnect and release the B-channel; release the call reference and enter the Null state.

When a backup D-channel is available, the procedures in Annex F may be used.

The implementation of timer T309 in the user side is optional and in the network side is mandatory.

When a Q.931 entity internally clears the call as a result of data link failure, as an option, it may request the re-establishment of the data link in order to attempt to send a DISCONNECT message across the interface.

5.8.10 Status enquiry procedure

Whenever an entity wishes to check the correctness of a call state at a peer entity, a STATUS ENQUIRY message may be sent requesting the call state. This may, in particular, apply to procedural error conditions described in 5.8.8 and 5.8.9.

Upon sending the STATUS ENQUIRY message, timer T322 shall be started in anticipation of receiving a STATUS message. While timer T322 is running, only one outstanding request for call state information shall exist. Therefore, if timer T322 is already running, it shall not be restarted. If a clearing message is received before timer T322 expires, timer T322 shall be stopped, and call clearing shall continue.

Upon receipt of a STATUS ENQUIRY message, the receiver shall respond with a STATUS message, reporting the current call state (the current state of an active call or a call in progress, or the Null state if the call reference does not relate to an active call or to a call in progress) and cause No. 30, *response to STATUS ENQUIRY*; No. 97, *message type non-existent or not implemented*, or No. 98*, message not compatible with call state or message type non-existent or not implemented* (see 5.8.4). Receipt of the STATUS ENQUIRY message does not result in a state change.

The sending or receipt of the STATUS message in such a situation will not directly affect the call state of either the sender or receiver. The side having received the STATUS message shall inspect the Cause information element. If the STATUS message contains cause No. 97, *message type non-existent or not implemented*, or No. 98*, message not compatible with call state or message type non-existent or not implemented*, timer T322 shall continue to time for an explicit response to the STATUS ENQUIRY message. If a STATUS message is received that contains cause No. 30, *response to STATUS ENQUIRY*, timer T322 shall be stopped and the appropriate action taken, based on the information in that STATUS message, relative to the current state of the receiver. If timer T322 expires and a STATUS message with cause No. 97, *message type non-existent or not implemented* or No. 98, *message not compatible with call state or message type non-existent or not implemented* was received, the appropriate action shall be taken, based on the information in that STATUS message, relative to the current call state of the receiver.

These further *appropriate actions* are implementation dependent. However, the actions prescribed in the following subclause shall apply.

If timer T322 expires, and no STATUS message was received, the STATUS ENQUIRY message may be retransmitted one or more times until a response is received. The number of times the STATUS ENQUIRY message is retransmitted as an implementation dependent value. The call shall be cleared to the local interface with cause No. 41, *temporary failure*, if the STATUS ENQUIRY is retransmitted the maximum number of times. If appropriate, the network shall also clear the network connection, using cause No. 41, *temporary failure*.

5.8.11 Receiving a STATUS message

On receipt of a STATUS message reporting an incompatible state, the receiving entity shall:

a) clear the call by sending the appropriate clearing message with cause No. 101, *message not compatible with call state*; or

b) take other actions which attempt to cover from a mismatch and which are an implementation option.

Except for the following rules, the determination of which states are incompatible is left as an implementation decision:

a) If a STATUS message indicating any call state except the Null state is received in the Null state, then the receiving entity shall either:

   1) send a RELEASE message with cause No. 101, *message not compatible with call state*, and then follow the procedures of 5.3; or

   2) send a RELEASE COMPLETE message with cause No. 101, *message not compatible with call state*, and remain in the Null state.

b) If a STATUS message indicating any call state except the Null state is received in the release request state, no action shall be taken.

c) If a STATUS message, indicating the Null state, is received in any state except the Null state, the receiver shall release all resources and move into the Null state. If appropriate, the network shall also clear the network connection, using cause No. 41, *temporary failure.*

When in the Null state, the receiver of a STATUS message indicating the Null state shall take no action other than to discard the message and shall remain in the Null state.

A STATUS message may be received indicating a compatible call state but containing one of the following causes:

i) No. 96 – *Mandatory information element is missing*;

ii) No. 97 – *Message type non-existent or not implemented*;

iii) No. 98 – *Message not compatible with call state or message type non-existent or not implemented*;

iv) No. 99 – *Information element/parameter non-existent or not implemented*; or

v) No. 100 – *Invalid information element contents*.

In this case, the actions to be taken are an implementation option. If other procedures are not defined, the receiver shall clear the call with the appropriate procedure defined in 5.3, using the cause specified in the received STATUS message.

On receipt of a STATUS message specifying the global call reference and reporting an incompatible state in the restart request or restart state, the receiving Q.931 entity shall inform layer management and take no further action on this message.

When in the Null state, then on receipt of a STATUS message with the global call reference no action shall be taken.

NOTE – Further actions, as a result of higher layer activity (e.g. system or layer management) are implementation dependent (including the retransmission of RESTART).

Except for the above case, the error handling procedures when receiving a STATUS message specifying the global call reference are an implementation option.

5.9 User notification procedure

This procedure allows the network to notify a user of any appropriate call-related event during the active state of a call. It also allows a user to notify the remote user of any appropriate call-related event during the active state of a call by sending a NOTIFY message containing a notify indicator to the network; upon receipt of this message, the network must send a NOTIFY message containing the same notify indicator to the other user involved in the call. No state change occurs at any of the interface sides following the sending or the receipt of this message.

5.10 Basic telecommunication service identification and selection

5.10.1 Additional procedures at the coincident S and T reference point

5.10.1.1 Normal operation

Procedures for bearer capability selection are described in subclauses 5.11.1 and 5.11.2. Procedures for high layer compatibility selection are described in subclauses 5.12.1 and 5.12.2.

Each basic telecommunications service has the required bearer capability information element encodings, and if applicable the required High layer compatibility information element encodings, defined for that service (e.g. see Recommendation Q.939).

The destination user shall identify the requested teleservice by taking the presented Bearer capability and High layer compatibility information elements in all combinations. Where a permutation is not identified as a defined teleservice, that combination shall be ignored. Where a combination is identified as a defined teleservice, that combination may be considered for the purposes of service provision. If there are no valid combinations, the presented Bearer capability information elements shall be considered in order to identify a bearer service.

The destination user shall identify the requested bearer services from the values of the presented Bearer capability information elements.

NOTE – These requirements do not preclude the user performing compatibility checking on all compatibility information according to Annex B.

The originating network shall optionally perform subscription checks for all valid combinations in the order defined for the particular service. If the user has not subscribed to the prime service, the network shall check for the next following basic service and so on. If the user has not subscribed to any of the basic services, the call shall be released with cause No. 57, *Bearer capability not authorized*. If a fallback occurs as a result of these checks, the procedures of 5.11.1 and 5.12.1 shall apply.

The destination network shall optionally perform subscription checks for all valid combinations in the order defined for the particular service. If the user has not subscribed to the prime service, the network shall check for the next following basic service and so on. The subscription check may then result in one of the following four possibilities:

a) **The user has subscribed to the prime service**

The call shall be offered to the called user without any modification following the procedures in 5.11.2 or 5.11.3 "bearer capability selection", and 5.12.2 or 5.12.3 "high layer compatibility selection".

b) **The user has not subscribed to the prime service, but to one of the valid combinations, different to the lowest basic service**

The call shall be offered to the called user, with the highest subscribed basic service including the fallback possibility. The procedures in 5.11.2 or 5.11.3 "bearer capability selection", and 5.12.2 or 5.12.3 "high layer compatibility selection" will then apply. No indication of the fallback will be sent towards the calling user before a bearer has been established, unless the called user indicates fallback prior to the bearer establishment.

c) **The user has subscribed to the lowest basic service among the valid combinations**

The call shall be offered to the called user containing the subscribed basic service and an indication of fallback shall be sent towards the calling user in the next message to be sent.

d) **The user has not subscribed to any service**

The call shall be cleared.

5.10.1.2 Exceptional procedures

Not applicable.

5.10.2 Procedures for interworking with private ISDNs

5.10.2.1 Normal operation

Procedures for bearer capability selection are described in 5.11.3. Procedures for high layer compatibility selection are described in subclause 5.12.3.

Each basic telecommunications service has the required Bearer capability information element encodings, and if applicable the required High layer compatibility information element encodings, defined for that service (e.g. see Recommendation Q.939).

The user (the private ISDN) shall identify the requested teleservices by taking the presented Bearer capability and High layer compatibility information elements in all combinations. Where a combination is not identified as a defined teleservice, that combination shall be ignored. Where a combination is identified as a defined teleservice, that combination may be considered for the purposes of service provision.

The user (the private ISDN) shall identify the requested bearer services from the values of the requested Bearer capability information elements.

5.10.2.2 Exceptional procedures

Not applicable.

5.11 Signalling procedures for bearer capability selection

The procedures in this subclause form an optional part of this Recommendation, but are a mandatory requirement for the provision of certain bearer services or teleservices. Provision of these procedures between the originating user and the originating network, and also between the destination network and the destination user, is thus subject to bilateral agreement, e.g. a subscription arrangement for the provision of that bearer service or teleservice to each user.

The support of such basic services may require the ability to support transmission and reception of either two or three bearer capability information elements.

These procedures shall apply only in the case where the call, or call request, as currently routed, is entirely within the ISDN. It will not apply to situations involving interworking with non-ISDNs.

NOTE – The use of the Low layer compatibility information element in conjunction with these procedures requires further study and the interpretation of any received Low layer compatibility information element is not defined.

5.11.1 Procedures for the originating user to indicate bearer capability selection is allowed

5.11.1.1 Normal operation

For some bearer services or teleservices, the originating user can indicate that:

– fallback to an alternative bearer capability is allowed; or

– fallback to an alternative bearer capability is not allowed.

If the calling user allows fallback to occur to an alternative bearer capability, then the user shall indicate this to the network by including repeated Bearer capability information elements within the SETUP message sent to indicate the presence of a call request. This procedure allows a maximum of three Bearer capability information elements in the SETUP message.

When two or three Bearer capability information elements are present, their order shall indicate the priority of the bearer capabilities. Bearer capability information elements shall be in ascending order of priority, i.e. a subsequent Bearer capability information element shall indicate a bearer capability with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs at the destination user, or fallback does not occur, the originating network shall include in the CONNECT message sent to the calling user the Bearer capability information element of the resultant bearer service or teleservice.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs within the ISDN (e.g. bearer capability selection is not supported or the selected route does not support the preferred bearer capability), the originating network shall include in a PROGRESS message or other appropriate call control message sent to the calling user a Progress indicator information element with the progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*. The originating network shall include the Bearer capability information element of the resultant bearer service or teleservice.

When a PROGRESS message is sent containing a Progress indicator information element with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, neither the user nor the network shall stop timers described in 5.1.6 as a result of this action.

5.11.1.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that:

a) If the calling user receives no Bearer capability information element in the CONNECT message, or prior to the CONNECT message in some other call control message, the user shall assume that the bearer service or teleservice corresponds to the first Bearer capability information element that the user included in the SETUP message.

b) If the calling user receives a Progress indicator information element with a progress description No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band*, or progress description No. 2, *destination address is non-ISDN*, subsequent to a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, then the last received Progress indicator information element shall be taken account of. Where the progress description is No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band* or progress description is No. 2, *destination address is non-ISDN*, the user shall assume a bearer service category of circuit-mode 64 kbit/s 8 kHz structured usable for 3.1 kHz audio information transfer.

c) If the calling user includes a Low layer compatibility information element in a SETUP message containing a repeated Bearer compatibility information element, even though this is an error condition, the network shall continue normal call handling, i.e. transport the Low layer compatibility information element transparently across the network.

d) If the calling user receives a call control message other than the CONNECT message containing a Bearer capability information element but without a Progress indicator information element, with progress description No. 5, *interworking has occurred and has*

resulted in a telecommunication service change, the calling user shall handle the call in the normal manner.

e) If the calling user receives no Bearer capability information element in a call control message other than CONNECT but the Progress indicator information element, with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, present, the calling user shall assume that the bearer service or teleservice corresponds to the first Bearer capability information element that was included in the SETUP message.

5.11.2 Procedures for bearer capability selection at the destination side

5.11.2.1 Normal operation

If the calling user and the network operator allow fallback to occur to an alternative bearer capability, then the destination network shall indicate this to the destination user by including repeated Bearer capability information elements within the SETUP message sent to indicate the presence of a call request.

When two or three Bearer capability information elements are present, their order shall indicate the priority of the bearer capabilities. Bearer capability information elements shall be in ascending order of priority, i.e. a subsequent Bearer capability information element shall indicate a bearer capability with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and the user wishes to accept the call without having fallback occur, the user shall include in the CONNECT message sent to the network the Bearer capability information element of the requested bearer service or teleservice.

If fallback allowed is indicated in the SETUP message as described above, and the user wishes to accept the call with having fallback occur to the lowest priority alternative bearer capability, the user may, but need not, include in the CONNECT message sent to the network the Bearer capability information element of the alternative bearer service or teleservice.

If no Bearer capability information element is indicated by the called user, the network shall assume that the lowest priority bearer capability is selected.

If fallback allowed is indicated in the call request, and no interworking has been encountered (i.e. a progress description No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band*, or progress description No. 2, *destination address is non-ISDN*; has not been sent), the destination network shall indicate the resultant bearer capability and connection type to the originating network at the time the bearer is established, even if no Bearer capability information element is received from the destination user.

5.11.2.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that:

a) If a low layer compatibility information is received from the originating network for a connection request for which bearer capability selection is indicated, even though this is an error condition, the network shall include the low layer compatibility information in the Low layer compatibility information element in the SETUP message sent to the destination user.

b) If a Low layer compatibility information element is included in the received SETUP message containing a repeated Bearer capability information element, the destination user may ignore the received Low layer compatibility information element.

c) If the called user sends a Bearer capability information element in the CONNECT message that contains an information transfer capability which is not that requested or the nominated alternative, the destination network shall clear the call using normal clearing procedures with clearing cause No. 111, *protocol error unspecified*.

5.11.3 Procedures for interworking with private ISDNs

5.11.3.1 Procedures for the originating user to indicate bearer capability selection is allowed

The procedures of 5.11.1 shall apply.

5.11.3.2 Procedures for bearer capability selection at the destination side of a public ISDN

5.11.3.2.1 Normal operation

If a private ISDN is attached to the access at the destination interface, the following procedures are applicable at call request. The private ISDN acts as the called user.

If the calling user allows fallback to occur to an alternative bearer capability, then the network shall indicate this to the called user by means of repeated Bearer capability information elements within the SETUP message sent to indicate the presence of a call request.

When two or three Bearer capability information elements are present, their order shall indicate the priority of the bearer capabilities. Bearer capability information elements shall be in ascending order of priority, i.e. a subsequent Bearer capability information element shall indicate a bearer capability with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs at the destination user (beyond the private ISDN) or fallback does not occur, the user shall include in the CONNECT message sent to the network the Bearer capability information element of the resultant bearer service or teleservice.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs within the private ISDN, the user shall include in a PROGRESS message or other appropriate call control message sent to the network a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*. The user shall include the Bearer capability information element of the resultant bearer service or teleservice.

When a PROGRESS message is sent containing a Progress indicator information element with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, neither the user nor the network shall stop timers described in 5.2.6 as a result of this action.

5.11.3.2.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that:

a) If the network receives no Bearer capability information element in the CONNECT message, or prior to the CONNECT message in some other call control message, the network shall assume that the bearer service or teleservice corresponds to the first Bearer capability information element that the network included in the SETUP message.

b) If the network receives a Progress indicator information element with a progress description No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band*, or progress description No. 2, *destination address is non-ISDN*, subsequent to a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, then the last received

Progress indicator information element shall be taken account of. Where the progress description is No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band* or the progress description is No. 2, *destination address is non-ISDN*, the network shall assume a bearer service category of circuit-mode 64 kbit/s 8 kHz structured usable for 3.1 kHz audio information transfer.

c) If a low layer compatibility information is received from the originating network for a connection request for which bearer capability selection is indicated, even though this is an error condition, the network shall include the low layer compatibility information in the Low layer compatibility information element in the SETUP message sent to the destination user.

d) If the network includes a Low layer compatibility information element in a SETUP message containing a repeated Bearer compatibility information element, even though this is an error condition, the user shall continue normal call handling, i.e. transport the Low layer compatibility information element transparently across the private network.

e) If the called user sends a Bearer capability information element in any call control message that contains an information transfer capability which is not that requested or the nominated alternative, the destination network shall clear the call using normal clearing procedures with clearing cause No. 111, *protocol error, unspecified*.

f) If the network receives a call control message other than a CONNECT message containing a Bearer capability information element, but without a Progress indicator information element with progress description No. 5, *interworking has occurred and has resulted in a bearer service change*, the network shall act as if the Progress indicator information element, with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, was present and handle the call in the normal manner.

g) If the network receives no Bearer capability information element in a call control message other than CONNECT but the Progress indicator information element, with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, present, the network shall assume that the bearer service or teleservice corresponds to the first Bearer capability information element that the network included in the SETUP message.

5.11.4 Channel selection

When all of the alternative bearers offered by the calling side use the same capacity on the ISDN interface channel, selection procedures in 5.1.2 and 5.2.3 shall apply.

When a preferred bearer requires a larger bearer than an allowed alternate fallback, e.g. 6×64 kbit/s to 64 kbit/s, channel selection procedures will follow 5.1.2 and 5.2.3 for the most preferred bearer, i.e. with the largest capacity requirement. If fallback occurs, the fallback bearer will use the lowest available time slot(s) corresponding to the channel selected for the preferred bearer. The remaining time slots will be available for further use after indication of fallback on the interface. For example, if time slots 7 through 12 were selected for a preferred 6×64 kbit/s bearer and later the call falls back to 64 kbit/s or speech, time slot 7 will be used for the call and time slots 8 through 12 will be available for further use after the CONNECT message has been received for the call. If early backward cut through is applicable to any of the bearer capabilities indicated, it will be provided on the lowest numbered time slot within the channel selected.

NOTE – These fallback procedures are not permitted if the user explicitly requests H0, H11 or H12.

Optional procedures in Annex N may be used to provide for flexible channel negotiation in conjunction with bearer fallback procedures.

5.12 Signalling procedures for high layer compatibility selection

The procedures in this subclause form an optional part of this Recommendation, but are a mandatory requirement for the provision of certain teleservices. Provision of these procedures between the originating user and the originating network, and also between the destination network and the destination user is thus subject to bilateral agreement, e.g. a subscription arrangement for the provision of that teleservice to each user.

These procedures shall apply only in the case where the call, or call request, as currently routed, is entirely within the ISDN. It will not apply to situations involving interworking with non-ISDNs.

5.12.1 Procedures for the originating user to indicate high layer compatibility selection is allowed

5.12.1.1 Normal operation

In some networks, the originating user can indicate that:

– fallback to an alternative high layer compatibility is allowed; or

– fallback to an alternative high layer compatibility is not allowed.

If the calling user allows fallback to occur to an alternative high layer compatibility, then the user shall indicate this to the network by means of repeated High layer compatibility information elements within the SETUP message sent to indicate the presence of a call request. This procedure allows a maximum of two High layer compatibility information elements in the SETUP message.

The order of the information elements shall be in ascending order of priority, i.e. a subsequent High layer compatibility information element shall indicate a high layer compatibility with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs at the destination user, or fallback does not occur, the originating network shall include in the CONNECT message sent to the calling user the High layer compatibility information element of the resultant high layer compatibility.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs within the ISDN, the originating network shall include in a PROGRESS message or other appropriate call control message sent to the calling user a Progress indicator information element with the progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*. The originating network shall include the High layer compatibility information element of the resultant high layer compatibility.

When a PROGRESS message is sent containing a Progress indicator information element with progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, neither the user nor the network shall stop timers described in 5.1.6 as a result of this action.

5.12.1.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that:

a) If the calling user receives no High layer compatibility information element in the CONNECT message, or prior to the CONNECT message in some other call control message, the user shall assume that the high layer compatibility is unknown.

   NOTE – It may be possible to subsequently identify the high layer compatibility from any in-band protocol within the B-channel.

b) If the calling user receives a Progress indicator information element with a progress description No. 1, *call is not end-to-end ISDN*; *further call progress information may be*

available in-band or progress description No. 2, *destination address is non-ISDN*, subsequent to a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, then the last received Progress indicator information element shall be taken account of. Where the progress description is No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band*, or progress description is No. 2, *destination address is non-ISDN*, the user shall assume a bearer service category of circuit-mode 64 kbit/s 8 kHz structured usable for 3.1 kHz audio information transfer.

5.12.2 Procedures for high layer compatibility selection at the destination side

5.12.2.1 Normal operation

If the calling user and the network operator allow high layer compatibility selection, then the destination network shall indicate this to the destination user by including multiple High layer compatibility information elements within the SETUP message sent to indicate the presence of a call request.

The order of the information elements shall be in ascending order of priority, i.e. a subsequent High layer compatibility information element shall indicate a high layer compatibility with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and the user wishes to accept the call without having fallback occur, the user shall include in the CONNECT message sent to the network the High layer compatibility information element of the requested high layer compatibility.

If fallback allowed is indicated in the SETUP message as described above, and the user wishes to accept the call with having fallback occur to the lowest priority alternative high layer compatibility, the user may, but need not, include in the CONNECT message sent to the network the High layer compatibility information element of the alternative high layer compatibility.

If no High layer compatibility information element is indicated by the called user, the network shall assume that the lowest priority high layer compatibility is selected.

If fallback allowed is indicated in the call request, and no interworking has been encountered (i.e. a progress description No. 1, *call is not end-to-end ISDN, further call progress information may be available in-band*, or progress description No. 2, *destination address is non-ISDN* has not been sent), then the destination network shall indicate the resultant high layer compatibility to the originating network at the time the bearer is established, even if no High layer compatibility information element is received from the destination user.

5.12.2.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that if the called user sends a High layer compatibility information element in the CONNECT message that is not as requested or the nominated alternative, the destination network shall pass this transparently towards the calling user.

5.12.3 Procedures for interworking with private ISDNs

5.12.3.1 Procedures for the originating user to indicate high layer compatibility selection is allowed

The procedures of 5.12.1 shall apply.

5.12.3.2 Procedures for high layer compatibility selection at the destination side of a public ISDN

5.12.3.2.1 Normal operation

If a private ISDN is attached to the access at the destination interface, the following procedures are applicable at call request. The private ISDN acts as the called user.

If the calling user allows fallback to occur to an alternative high layer compatibility, then the network shall indicate this to the called user by including multiple High layer compatibility information elements within the SETUP message sent to indicate the presence of a call request.

The order of the information elements shall be in ascending order of priority, i.e. a subsequent High layer compatibility information element shall indicate a high layer compatibility with higher priority.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs at the destination user (beyond the private ISDN) or fallback does not occur, the user shall include in the CONNECT message sent to the network the High layer compatibility information element of the resultant high layer compatibility.

If fallback allowed is indicated in the SETUP message as described above, and fallback occurs within the private ISDN, the user shall include in a PROGRESS message or other appropriate call control message sent to the network a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*. The user shall include the High layer compatibility information element of the resultant high layer compatibility.

When a PROGRESS message is sent containing a Progress indicator information element with progress No. 5, *interworking has occurred and has resulted in a telecommunication service change*, neither the user nor the network shall stop timers described in 5.2.6 as a result of this action.

5.12.3.2.2 Exceptional procedures

The procedures of 5.8 shall apply, with the addition that:

a) If the network receives no High layer compatibility information element in the CONNECT message, or prior to the CONNECT message in some other call control message, the network shall assume that the high layer compatibility is unknown.

   NOTE – It may be possible to subsequently identify the high layer compatibility from any in-band protocol within the B-channel.

b) If the network receives a Progress indicator information element with a progress description No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band,* or progress description No. 2, *destination address is non-ISDN* subsequent to a Progress indicator information element with a progress description No. 5, *interworking has occurred and has resulted in a telecommunication service change*, then the last received Progress indicator information element shall be taken account of. Where the progress description is No. 1, *call is not end-to-end ISDN*; *further call progress information may be available in-band*, or progress description is No. 2, *destination address is non-ISDN*, the network shall assume a bearer service category of circuit-mode 64 kbit/s 8 kHz structured usable for 3.1 kHz audio information transfer.

6 Packet communication procedures

This clause is intended to explain the role of the D-channel signalling procedures in the support of packet communications in an ISDN. A complete description of terminal adaptor functions can be found in Recommendation X.31 [14].

According to Recommendation X.31, the user may access packet facilities by means of one of the following alternatives:

a) *Circuit-switched access to PSPDN services (Case A)*

by establishing a transparent circuit-switched access connection through the ISDN to the access port of a public network (e.g. PSPDN) referred to as "Access Unit (AU)" in the following subclauses. This connection may be initiated by the user or the AU. From the ISDN point of view, the circuit-switched call control procedures of clause 5 apply. Only the B-channel is used in this case.

b) *Packet-switched access to an ISDN virtual circuit service (Case B)*

by establishing a packet-mode access connection to the Packet Handler (PH) of an ISDN. This connection may be initiated by the user or the ISDN. Both B- and D-channels may be used in this case.

A more detailed description of the protocol and text of 6.1 to 6.5 is in Recommendation X.31. Appendix II/Q.931 and Appendix III/X.31 are essentially the same.

The term "user" refers to the user equipment which may consist of an ISDN packet-mode terminal (TE1) or a combination of an existing data terminating equipment (DTE/TE2) attached to a Terminal Adaptor (TA). A DTE may not receive all of the information provided in Q.931 signalling messages at the user-network interface.

The ISDN TA/TE1 presents an S/T interface towards the network and therefore the TA/TE1 implementation should embody the procedures described in Recommendation Q.921 and this Recommendation for B- and D-channel connection establishment and control.

For demand access connections, subclauses 6.1 through 6.4 apply. Example message flows for demand access connections are shown in Appendix II.

Two physical types of semi-permanent connections on B- and D-channels are covered in this clause:

1) physical layer semi-permanently established between the terminal and the PH/AU, i.e. the I.430 and I.431 physical layer remains activated and the physical path through the ISDN is connected semi-permanently; and

2) X.25 data link and physical layers semi-permanently established between the terminal and the PH/AU (in this type, both the user and the network shall keep the X.25 data link layer in the established state).

When a PVC is used, there must exist a type 2 semi-permanent connection.

In semi-permanent connection type 1, the procedures of 6.3 are followed for X.25 call establishment and release.

In semi-permanent connection type 2, only the procedures of 6.3.2 are followed for X.25 call establishment and release.

When semi-permanent connection type 2 is used for PVCs, none of the following procedures apply.

Semi-permanent connections are established via a provisioning process without Q.931 procedures.

6.1 Outgoing access

If the user selects an already established channel for the outgoing X.25 virtual call, then the procedures described in 6.3 apply. If the selected channel is not established to the AU/PH, then the procedures for activating a channel described in the following subclauses are to be used before establishing the virtual call using the procedures of 6.3.

For outgoing X.25 data calls, the user first must decide whether circuit-switched (Case A) or packet-switched services (Case B) are desired from the network. For outgoing circuit calls, the user follows the procedures of 6.1.1. For outgoing packet calls, the user decides whether the B-channel or D-channel is to be used for the packet call. If the user decides to use the B-channel, then the procedures described in 6.1.2.1 are used. If the user decides to use the D-channel, then the procedures described in 6.1.2.2 are used.

NOTE – Some networks may not support every type of access. In the case of B-channel access, the network will clear a request for unsupported services by sending a RELEASE COMPLETE message with cause No. 65, *bearer capability not implemented*. In the case of a request for D-channel access (an SABME with SAPI = 16), on a network port which does not support the service, no response is required of the network.

6.1.1 Circuit-switched access to PSPDN services (Case A)

The B-channel connection between the user and the AU shall be controlled using the D-channel signalling procedures for call establishment described in 5.1. The specific B-channel to be used as a switched connection is selected using the channel selection procedures described in 5.1.2 and summarized in Table 6-1.

Table 6-1/Q.931 – User requested channel and network response, Outgoing access to either an AU or PH

Channel indicated in the SETUP message user-to-network direction			**Allowable network response**
Information channel selection	**Preferred or exclusive**	**D-channel indicator (Note 1)**	**(network-user)**
Bi	Exclusive	No	Bi
	Preferred	No	Bi, Bi′
Any	(Ignore)	No	Bi′
(Absent)			Bi′

Bi The indicated (idle) B-channel

Bi′ Any (other) idle B-channel

NOTE 1 – D-channel indicator shall be encoded "0" to indicate No and "1" to indicate Yes.

NOTE 2 – All other encodings are invalid.

NOTE 3 – All columns under the heading "Channel indicated in the SETUP message" indicate possible user codings of the Channel identification information element contained in the SETUP message sent by the user to the network requesting a connection to an AU or PH (see 4.5.13). The column under "Allowable network response" refers to the allowable responses by the network to the user.

On the basis of the call set-up information (e.g. called party number identifying an AU, transit network selection, etc.) and/or a subscription time agreement, the network provides a connection to the appropriate AU. The Bearer capability information element included in the SETUP message shall be encoded with:

– information transfer capability set to either:
  a) *unrestricted digital information*; or
  b) *restricted digital information*;
– transfer mode set to *circuit mode*;
– information rate set to *64 kbit/s*.

The user may also specify the layer 1 (e.g. rate adaption), layer 2 (i.e. LAPB), and layer 3 (i.e. X.25) information transfer protocols in the Low layer compatibility information element in the SETUP message (see Annex I).

6.1.2 Access to the ISDN virtual circuit service (Case B)

6.1.2.1 B-channel

Demand access B-channel connections are controlled using the D-channel signalling procedures for call establishment described in 5.1 using the messages defined in 3.2 with the following exceptions:

a) the procedures for overlap sending specified in 5.1.3 do not apply;

b) the procedures for call proceeding and overlap sending specified in 5.1.5.2 do not apply;

c) the procedures for notification of interworking at the origination interface specified in 5.1.6 do not apply;

d) the procedures for call confirmation indication specified in 5.1.7 do not apply;

e) the procedures for call connection specified in 5.1.8 apply as follows:
   - upon accepting the access connection, the network shall send a CONNECT message across the user-network interface to the calling user and enter the Active state;
   - this message indicates to the calling user that an access connection to the packet handler has been established;
   - on receipt of the CONNECT message, the calling user shall stop timer T310, if running, may optionally send a CONNECT ACKNOWLEDGE message, and shall enter the Active state;

f) the procedures for call rejection specified in 5.1.9 apply as follows:
   - when unable to accept the access connection, the network shall initiate ISDN access connection clearing at the originating user-network interface as described in 5.3;

g) the procedures for transit network selection specified in 5.1.10 do not apply.

The specific B-channel to be used as a demand connection is selected using the channel selection procedures described in 5.1.2 and summarized in Table 6-1.

For a demand connection to an ISDN PH, the Bearer capability information element included in the SETUP message shall be coded with:

- information transfer capability set to *unrestricted digital information*;
- transfer mode set to *packet mode*;
- information transfer rate set to 00000;
- user information layer 2 protocol set to *Recommendation X.25, link layer*;
- user information layer 3 protocol set to *Recommendation X.25, packet layer*.

NOTE – Octets 5a, 5b, 5c and 5d shall not be included.

The demand access connection can then be used to support packet communications according to X.25 link layer and X.25 packet layer procedures as specified in 6.3.

Some ISDNs may require the Calling party number and the Calling party subaddress information elements to be included in the SETUP message to select a specific user profile.

6.1.2.2 D-channel

The D-channel provides a connection which enables the ISDN user terminal to access a PH function within the ISDN by establishing a link layer connection (SAPI = 16) to that function which can then

be used to support packet communications according to X.25 layer 3 procedures as defined in 6.3. The X.25 packet layer uses the acknowledged information transfer service (i.e. I-frames) provided by LAPD (see Recommendation Q.920 [45]). Consequently, Q.931 procedures are not required to provide D-channel access.

A number of packet mode user equipment can operate simultaneously over the D-channel, each using a separate ISDN layer 2 data link identified by an appropriate address (see Recommendation Q.921) in frames transferred between the user and the PH.

6.2 Incoming access

6.2.1 Access from PSPDN services (Case A)

The ISDN signals the establishment of the circuit-mode connection using the procedures described in 5.2. The X.25 virtual calls are signalled between the user and the AU using the procedures described in 6.3.

6.2.1.1 General

The general procedures performed by the AU are those defined in Recommendation X.32.

6.2.1.2 Channel selection

If the ISDN physical circuit desired by the AU does not exist between the terminal and the AU, the procedures for physical channel establishment described in the following subclauses apply.

The format of the SETUP message sent by the network to the user is in accordance with 3.1.

The Bearer capability information element included in the SETUP message shall be encoded with:

– information transfer capability set to either:
  a) *unrestricted digital information*; or
  b) *restricted digital information*;
– transfer mode set to *circuit mode*;
– information rate set to *64 kbit/s*.

The Channel identification information element shall be encoded according to Table 6-2.

The B-channel connection to the called user shall be established by the network using the signalling procedures described in 5.2. The call is offered by sending the SETUP message on a point-to-point data link or on the broadcast data link.

The user responds to the SETUP as defined in clause 5.

Table 6-2/Q.931 – Network requested channel and user response, Incoming access from an AU

Channel indicated in the SETUP message network-to-user direction			Allowable network response (user-network)
Information channel selection	**Preferred or exclusive**	**D-channel indicator (Note 1)**	
Bi	Exclusive	No	Bi
Bi	Preferred	No	Bi, Bi′ (Note 2)

Bi Indicated (idle) B-channel

Bi′ Any other idle B-channel (not permitted for broadcast call offering)

NOTE 1 – D-channel indicator shall be encoded "0" to indicate No and "1" to indicate Yes.

NOTE 2 – This encoding is not used for broadcast call offering.

NOTE 3 – All other encodings are invalid.

6.2.2 Access from the ISDN virtual circuit service (Case B)

To offer an incoming X.25 call, the network must perform the following steps in sequence:

1) *Channel selection* – The physical channel/logical link to be used for the incoming call must be identified. The network may use customer profile information, network resources, etc., to choose the channel or the procedures of Step 2 below.

2) *Physical channel/logical link establishment* – If the physical B-channel or the logical link of the D-channel has not been determined by Step 1, the network may use the procedures in 6.2.2.3. The network may then proceed with Step 3.

3) *X.25 virtual call establishment* – The network establishes the virtual call using the procedures described in 6.3.

In the configuration for the ISDN virtual circuit bearer service, the choice of channel type to be used for the delivery of a new *incoming call* packet shall be made by the network as described below.

a) A new *incoming call* packet may be indicated to the ISDN customer by a call offering procedure between the network and all user packet-mode terminals (see 3.2.3.2/X.31 and 3.2.3.3/X.31 [14]).

b) An incoming virtual call directed to a terminal with an established connection to the PH may be offered directly to the terminal over the established access connection without the use of Q.931 call offering procedures (see 3.2.3.1/X.31 and 3.2.3.2/X.31 [14]).

6.2.2.1 B-channel

When X.25 calls are to be offered on the B-channels without channel negotiation, the procedures described in 5.2 using the messages of 3.2 apply with the following exceptions:

a) The procedures for overlap receiving specified in 5.2.4 do not apply.

b) The procedures for receipt of CALL PROCEEDING and ALERTING specified in 5.2.5.2 apply with the following exception:

   – The receipt of an ALERTING message shall not cause the network to send a corresponding ALERTING message to the calling user.

c) The procedures for call failure specified in 5.2.5.4 apply with the following remark:

   – The network clears the incoming X.25 virtual call towards the calling X.25 DTE using the appropriate cause from Table 6-5.

d) The procedures for notification of interworking at the terminating interface specified in 5.2.6 apply with the following exceptions:
   - The case of the call entering an ISDN environment during call establishment is not applicable.
   - In the case of a call leaving the ISDN environment within the called user's premises, no notification is sent to the calling party.
   - The case of in-band information/patterns is not applicable.

e) The procedures for active indication specified in 5.2.8 apply with the following exception:
   - The network shall not initiate procedures to send a CONNECT message towards the calling user.

f) The procedures for user notification specified in 5.9 do not apply.

Where an established B-channel connection is to be used, the *incoming call* packet will be delivered in accordance with 6.3.

Where a new B-channel connection is to be established, the identity of the selected user will be associated with the Connection Endpoint Suffix (CES) from which the first CONNECT message has been received.

6.2.2.2 D-channel

The D-channel provides a connection which enables the ISDN PH to access an ISDN user terminal or vice versa. This access is accomplished by establishing an ISDN link layer connection (SAPI = 16) to the terminal or network which can then be used to support packet communications according to X.25 [5] layer 3 procedures as defined in 6.3.

The layer 2 procedures shall be in accordance with Recommendation Q.921 [3]. The D-channel provides a semi-permanent connection for packet access since all D-channel layer 2 frames containing a packet-mode SAPI (16) are routed automatically between the user and the PH function.

When an incoming call is offered to packet-mode user equipment at the user interface, the channel selection procedures described in 6.2.2.3 shall be used.

A number of packet mode terminals can operate simultaneously over the D-channel, each using a separate layer 2 link identified by an appropriate TEI (see Recommendation Q.921) in frames transferred between the terminal and the network.

6.2.2.3 Call offering

6.2.2.3.1 Channel selection through call offering

The call offering procedure is performed using the layer 3 messages and procedures of clause 5. The call offering procedure is integrated into the circuit-switched call control procedures, signalled on the D-channel, with the channel selection being accomplished by means of the channel selection procedure if offered as a network option.

As described in clause 5, the network selects the first user which responds to the call offering with a CONNECT message. When the selected user has requested that the X.25 call be set up over a new B-channel, the network will indicate that the channel is acceptable by returning a CONNECT ACKNOWLEDGE message to the user. If multiple terminals have responded positively to the SETUP message, the network shall clear each of the non-selected terminals with a RELEASE message containing cause No. 26, *non-selected user clearing*.

When the selected user has requested that the X.25 call be set up over an established B-channel or the D-channel, the network shall respond to the CONNECT message with a RELEASE message

containing cause No. 7, *call awarded and being delivered in an established channel*. The network shall also return a RELEASE message containing cause No. 26, *non-selected user clearing*, to any other positively responding terminals. The network will then deliver the X.25 virtual call over the selected channel.

NOTE 1 – There is no time significance between the delivery of the RELEASE message and the incoming call packet, i.e. either may occur first.

NOTE 2 – The network shall send the RELEASE message(s) and the user(s) shall respond with RELEASE COMPLETE.

If the channel indicated by the first positively responding user is not available, the network will use Q.931 call clearing procedures to clear the call with cause No. 6, *channel unacceptable*. If the channel indicated in the SETUP message is not acceptable to the user, the user will clear the call with a RELEASE COMPLETE message containing cause No. 34, *no circuit/channel available*, or cause No. 44, *requested circuit/channel not available*.

On the basis of a network option or subscription agreement, the network may choose the access channel or access channel type (e.g. B or D) for a particular incoming packet call.

When the Channel indication information element indicates *Channel indication = No channel, Exclusive*, and *D-channel indication – Yes*, then the Bearer capability information element should be coded as follows:

– information transfer capability set to *unrestricted digital information*;
– transfer mode set to *packet mode*;
– information rate set to *packet mode (00000)*;
– layer 2 protocol set to *Recommendation Q.921*;
– layer 3 protocol set to *Recommendation X.25, packet layer*.

In all other cases, the Bearer capability information element should be encoded as follows:

– information transfer capability set to either:
  a) *unrestricted digital information*; or
  b) *restricted digital information*;
– transfer mode set to *packet mode*;
– information rate set to *packet mode (00000)*;
– layer 2 protocol set to *Recommendation* X.25, *link layer*;
– layer 3 protocol set to *Recommendation* X.25, *packet layer*.

There exists an understanding that if the terminal responds with D-channel indication set (see Table 6-3), the Layer 2 protocol to be used is Recommendation Q.921 (LAPD).

Table 6-3/Q.931 – Network requested channel and user response, Incoming access for packet-mode

Channel indicated in the SETUP message network-to-user direction			Allowable network response (network-user)
Information channel selection	**Preferred or exclusive**	**D-channel indicator (Note 1)**	
Bi	Exclusive	No	Bi
		Yes	Bi, D
Bi	Preferred	No	Bi, Bi′, Bj
		Yes	Bi, Bi′, Bj, D
No channel	Preferred	No	Bj
		Yes	Bj, D
	Exclusive	Yes	D

Bi Indicated (idle) B-channel

Bi′ Any other idle B-channel (not permitted in response to broadcast call offering)

Bj An established B-channel under the user's control (a semi-permanent B-channel which is allocated to the user may be indicated if the user subscribes to the unconditional notification class)

D The D-channel

NOTE 1– D-channel indicator shall be encoded "0" to indicate No and "1" to indicate Yes.

NOTE 2– All other encodings are invalid.

The channel selection procedure for incoming calls is independent of the type of channel selected at the calling end. In this respect, any combination of channel type used at each end is possible, provided the user rates and available bandwidth are compatible.

The channel selection principle to be used in the procedure is shown in Table 6-3.

NOTE 3 – When the incoming SETUP message is sent on a broadcast data link with a Channel identification information element which indicates an idle B-channel and *preferred*, the called user is not permitted to respond with a different idle B-channel in the response. The option to respond with a different idle channel is restricted to point-to-point call offerings.

NOTE 4 – Networks providing packet-mode call offering shall provide Q.931 signalling procedures for packet-mode calls on SAPI = 0. For an interim period, some networks, by subscription agreement, may offer SAPI = 16 broadcast call offering procedures for providing Q.931 signalling. This option shall use all Q.931 procedures for packet-mode calls with the following restriction: All calls will be offered as *D-channel exclusive* and will not provide channel selection procedures. Terminals implementing SAPI = 16 procedures should also implement SAPI = 0 procedures for portability.

6.2.2.3.2 Information element mapping

Some networks may choose to provide a service of mapping some or all of the information from the *incoming call* packet into the SETUP message (see 3.2.3/X.31). Table 6-4 shows the mapping of the X.25 incoming call elements to Q.931 information elements. The *incoming call* packet will still contain these fields when it is delivered. See 3.2.3/X.31 for mapping requirements.

Table 6-4/Q.931 – Mapping of X.25 information elements to corresponding Q.931 SETUP message information elements in packet-mode incoming call[a)]

	Information elements in X.25 *incoming call* packet	**Corresponding information element in Q.931 SETUP message**
	Calling DTE address	Calling party number (Note 6)
	Called DTE address	Called party number
	User data (UD)	User-user information (Note 1)
	A-bit (Note 2)	For further study
	D-bit	Packet layer binary parameters
	Modulus	Packet layer binary parameters
X.25 user facility	Flow control parameter negotiation	Packet size, Packet layer window size
	Throughput class negotiation	Information rate (Note 4)
	Fast select	Packet layer binary parameters
	Reverse charging	Reverse charging indication
	Closed user group selection	Closed user group
	Closed user group with outgoing access selection	Closed user group
	Bilateral closed user group	For further study
	Transit delay selection and indication	Transit delay selection and indication
	Call redirection and deflection notification	Redirecting number
DTE facility	Calling address extension	Calling party subaddress
	Called address extension	Called party subaddress (Note 5)
	End-to-end transit delay	End-to-end transit delay
	Minimum throughput class	Information rate (Note 3)
	Expedited data negotiation	Packet layer binary parameters
	Priority	For further study
	Protection	For further study

Table 6-4/Q.931 – Mapping of X.25 information elements to corresponding Q.931 SETUP message information elements in packet-mode incoming call[a)] *(concluded)*

[a)] Mapping is optional or required as indicated in 8.2.3/X.31.

NOTE 1 – The maximum length of the user data within the User-user information element is network dependent and is either 32 or 128 octets.

NOTE 2 – The need and procedures for A-bit mapping is for further study.

NOTE 3 – This information is not always present even when the "Information rate" is provided in the Q.931 SETUP message.

NOTE 4 – When the "Throughput class negotiation" is not set in the X.25 *incoming call* packet, this information shall be provided as the default throughput values applying to the virtual call.

NOTE 5 – The network will map bits 8 and 7 of the first octet of the called address extension facility parameter field in X.25 *incoming call* packet to *type of subaddress* field in octet 3 of the Called party subaddress information element in the Q.931 SETUP message, assuming that the X.25 *incoming call* packet is coded based on the 1988 version of X.25. Therefore, the called user should notice that the received *type of subaddress* may not be correct when the X.25 *incoming call* packet is coded based on the 1984 version of X.25.

NOTE 6 – This mapping is mandatory and octet 3a shall be set with Presentation indicator set to *presentation allowed* and Screening indicator set to *network provided*.

6.2.2.3.3 Channel selection without call offering

Where the network and user have agreed beforehand, the network may route an incoming call to the called user over an established B-channel connection or D-channel link without the need for any signalling for channel selection.

6.3 X.25 virtual call establishment and release

In all cases, once the physical channel has been selected and, if necessary, connected to the PH or AU, the virtual call is established according to the procedures below. Some networks may require some of the terminal identification procedures of Recommendation X.32 as well.

6.3.1 Link layer establishment and release

Link layer (LAPB on the B-channel or LAPD on the D-channel) establishment shall be initiated by:

– the calling terminal in the case of outgoing calls;

– the AU in the case of incoming calls in Case A; or

– the PH in the case of incoming calls in Case B.

Link layer release may be initiated by:

– the terminal;

– the AU in Case A; or

– the PH in Case B.

6.3.2 Packet layer virtual call set-up and release

The packet layer procedures of Recommendation X.25 will be used for layer 3 call setup and release. The packet layer procedures will additionally be able to control and monitor the established or released state of the link layer.

In Case B, the PH may maintain a timer T320 (defined in this Recommendation). T320, if implemented, is started:

a) upon clearance of the last virtual call; or

b) upon transmission of a CONNECT message by the network in case of an outgoing B-channel access connection; or

c) upon transmission of a CONNECT ACKNOWLEDGE message by the network in case of an incoming B-channel access connection; or

d) upon establishment of the link layer for D-channel access connections.

T320 is cancelled upon:

1) establishment of the first (next) virtual call; or

2) receipt of a Q.931 clearing message from the user; or

3) disconnection of the SAPI = 16 link on the D-channel.

Upon expiry of timer T320, the PH will release the link layer and, in the case of B-channel access, initiate clearing of the B-channel.

X.25 logical channels are associated with their underlying logical link. Specifically, in the case of the use of the B-channel for packet communication, there is an association between the logical channels and the LAPB logical link below them. Thus, the same logical channel number may be used simultaneously on each different B-channel.

6.4 Call clearing

6.4.1 B-channel access

The clearing of the switched connection shall be effected by using the D-channel signalling procedures for call clearing as specified in 5.3. For access to PSPDN services, no exceptions apply. For the ISDN virtual circuit service, the messages of 3.2 are used, and the following exceptions apply:

– The terms defined in 5.3.1 (Terminology) apply by replacing "circuit-switched ISDN connection" with "demand packet mode access connection".

– The exception condition f) specified in 5.3.2 does not apply.

– The procedures for clearing with tones and announcements provided in 5.3.4.1 do not apply.

The B-channel may be cleared at any time by the user though, in general, it will be cleared following the clearing of the last virtual call over that B-channel. In the ISDN virtual circuit service, if the user clears the B-channel access connection using a Q.931 clearing message while X.25 virtual calls still exist on the B-channel, the network shall clear the X.25 virtual call(s) with cause No. 17, *remote procedure error*, and diagnostic No. 64, *call set-up, call clearing or registration problem*.

In Case B, if a Q.931 restart indication is received by the PH during the X.25 data transfer phase, the X.25 virtual calls shall be treated as follows:

– For switched virtual circuits which are established on a demand connection to the packet handler, an X.25 *clear indication* packet shall be sent with cause No. 9, *out of order*, and diagnostic No. 0, *no additional information*.

– For any virtual calls which are established on a semi-permanent connection to the packet handler, no action shall be taken.

At the expiration of timer T320, the network may disconnect the X.25 link layer and the access connection. B-channel clearing is as described in 5.3 with the exceptions above, with cause No. 102, *recovery on timer expiry*.

6.4.2 D-channel access

D-channel access connections are cleared using the disconnect procedures as defined in 6.3.

6.4.3 Additional error handling information

When an ISDN access connection failure occurs, or the X.25 virtual call is cleared prematurely, the rules of 5.8 shall apply. In addition, the following rules for determining the appropriate cause to be used shall apply in order of decreasing priority:

1) If a Q.931 clearing message or RESTART message is received by the PH during the X.25 data transfer phase, 6.4.1 applies.

2) In general, if an ISDN access connection is rejected by the destination user using Q.931 messages, the X.25 virtual call shall be cleared using a *clear indication* packet and cause No. 0, *DTE originated*, with diagnostic No. 0, *no additional information*. Some networks may map some Q.931 causes to the corresponding X.25 causes according to Table 6-5.

3) If a condition exists that prevents the Q.931 SETUP message from being delivered at the user-network interface, the X.25 virtual call shall be cleared using a *clear indication* packet and a cause shall be selected appropriate to the condition. Table 6-5 shall serve as a guide to selecting an appropriate cause, i.e. the X.25 mapping of the Q.931 cause describing the interface condition shall be used.

4) If the Q.931 SETUP message is sent across the user-network interface, but no response is received prior to the second expiry of timer T303, rule No. 3 applies.

5) If the Q.931 SETUP message is sent across the user-network interface, and a response other than a call rejection is received from a user which results in the clearing of the ISDN access connection at the user-network interface, the X.25 virtual call shall be cleared using a *clear indication* packet containing cause No. 17, *remote procedure error,* with diagnostic No. 64, *call set-up, call clearing or registration problem*.

6) If an X.25 *clear request* packet is received from the originating user prior to the delivery of the X.25 *incoming call* packet to the called user (premature clearing), the PH shall send a *clear confirmation* packet to the calling user and the access connection shall be treated as follows:

   – If the Q.931 SETUP message was associated with the Unconditional notification class of service (see 3.2.3/X.31), the access connection, when and if established, shall be cleared. The Q.931 clearing message shall contain the appropriate cause as described in Table 6-6.

   – If the Q.931 SETUP message was associated with the Conditional notification class of service (see 3.2.3/X.31) and there exists at least one terminal which responds positively to the Q.931 SETUP message, then two options are allowed:

      a) the access connection is cleared as described for the Unconditional class of service; or

      b) the access connection is established and timer T320 is started. Upon expiry of timer T320, the access connection is cleared with cause No. 102, *recovery on timer expiry*, and diagnostic indicating timer T320.

6.4.4 Cause mappings

6.4.4.1 Access to/from PSPDN services (Case A)

The AU may choose to follow the procedures in 6.4.4.2 when mapping between causes delivered by the ISDN or the PSPDN.

6.4.4.2 Access to/from the ISDN virtual circuit service (Case B)

There are several cases where it is necessary to map causes between this Recommendation and Recommendation X.25. ISDN networks shall use Tables 6-5 and 6-6 to map the causes between Q.931 and X.25 messages. The figures in Appendix II describe some example situations.

Table 6-5/Q.931 – Mapping of Q.931 cause fields to X.25 cause field

Item	Q.931 cause	Code	Q.931 diagnostic	X.25 cause	Code	X.25 diagnostic	Code
1	Unallocated (unassigned) number	1	Condition: unknown, transient, permanent	Not obtainable	13	Invalid called address	67
2	No route to destination	3	Condition: unknown, transient, permanent	Not obtainable	13	Invalid called address	67
3	Channel unacceptable	6	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
4	Normal call clearing	16	Condition: unknown, transient, permanent	DTE originated	0	No additional information	0
5	User busy	17	(None)	Number busy	1	No logical channel available	71
6	No user responding	18	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
7	No answer from user (user alerted)	19	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
8	Call rejected	21	Condition: unknown, transient, permanent + user applied diagnostics	DTE originated	0	No additional information	0
9	Number changed	22	New destination address	Not obtainable	13	Invalid called address	67
10	Destination out of order	27	(None)	Out of order	9	No additional information	0

Table 6-5/Q.931 – Mapping of Q.931 cause fields to X.25 cause field *(continued)*

Item	Q.931 cause	Code	Q.931 diagnostic	X.25 cause	Code	X.25 diagnostic	Code
11	Invalid number format (address incomplete)	28	(None)	Local procedure error	19	Invalid called address	67
12	Normal, unspecified	31	(None)	DTE originated	0	No additional information	0
13	No circuit/channel available	34	(None)	Number busy	1	No logical channel available	71
14	Network out of order	38	(None)	Out of order	9	No additional information	0
15	Temporary failure	41	(None)	Out of order	9	No additional information	0
16	Switching equipment congestion	42	Network identity	Network congestion	5	No additional information	0
17	Requested circuit/channel not available	44	(None)	Number busy	1	No logical channel available	71
18	Resource unavailable, unspecified	47	(None)	Network congestion	5	No additional information	0
19	Quality of service not available	49	Condition: unknown, transient, permanent	Network congestion	5	No additional information	0
20	Bearer capability not authorized	57	Attribute number	Incompatible destination	33	No additional information	0
21	Bearer capability not presently available	58	Attribute number	Remote procedure error	17	Call set-up, call clearing or registration problem	64
22	Service or option not available, unspecified	63	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
23	Bearer capability not implemented	65	Attribute numbers	Incompatible destination	33	No additional information	0
24	Channel type not implemented	66	Channel type	Remote procedure error	17	Call set-up, call clearing or registration problem	64
25	Service or option not implemented, unspecified	79	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
26	Valid call reference value	81	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64

Table 6-5/Q.931 – Mapping of Q.931 cause fields to X.25 cause field *(continued)*

Item	Q.931 cause	Code	Q.931 diagnostic	X.25 cause	Code	X.25 diagnostic	Code
27	Identified channel does not exist	82	Channel identity	Remote procedure error	17	Call set-up, call clearing or registration problem	64
28	Incompatible destination	88	Incompatible parameter	Incompatible destination	33	No additional information	0
29	Invalid message, unspecified	95	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
30	Mandatory information element is missing	96	Information element identifier(s)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
31	Message type non-existent or not implemented	97	Message type	Remote procedure error	17	Call set-up, call clearing or registration problem	64
32	Message not compatible with call state or message type non-existent or not implemented	98	Message type	Remote procedure error	17	Call set-up, call clearing or registration problem	64
33	Information element/parameter non-existent or not implemented	99	Information element identifier(s)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
34	Invalid information element contents	100	Information element identifier(s)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
35	Message not compatible with call state	101	Message type	Remote procedure error	17	Call set-up, call clearing or registration problem	64
36	Recovery on timer expiry	102	Timer number	Remote procedure error	17	Call set-up, call clearing or registration problem	64
37	Protocol error, unspecified	111	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64
38	Interworking, unspecified	127	(None)	Remote procedure error	17	Call set-up, call clearing or registration problem	64

Table 6-5/Q.931 – Mapping of Q.931 cause fields to X.25 cause field *(concluded)*

NOTE 1 – When clearing occurs during the X.25 data transfer phase, the procedure described in 6.4.1 should be used.

NOTE 2 – When a Q.931 RESTART message is received during the X.25 data transfer phase, switched virtual circuits shall be cleared with a *clear indication* packet containing cause No. 9, *out of order*, with diagnostic No. 0, *no additional information*. Permanent virtual circuits shall have an X.25 *reset* packet sent with the same cause and diagnostic.

6.5 Access collision

When the network offers a packet-mode call at the interface simultaneously with the user requesting a packet-mode call, the network shall give priority to the completion of the incoming call. If the user determines that accepting the incoming call would meet the needs of its own outgoing call request, the user may clear the call request and accept the incoming call.

7 User signalling bearer service call control procedures

7.1 General characteristics

This feature allows the users to communicate by means of user-to-user signalling without setting up a circuit-switched connection. A temporary signalling connection is established and cleared in a manner similar to the control of a circuit-switched connection.

Table 6-6/Q.931 – Mapping of X.25 cause to Q.931 cause for premature clearing of the incoming call

	X.25 cause in clear indication packet				Q.931 error condition		
Item	**X.25/X.96 cause**	**Code**	**Diagnostic**	**Code**	**Q.931 cause**	**Code**	**Diagnostic**
1	DTE originated	0	No additional information	0	Normal call clearing	16	(None)
		1XX	DTE specified	XX			
2	Network congestion	5	No additional information	0	Switching equipment congestion	42	(None)
3	Out of order	9	No additional information	0	Destination out of order	27	(None)
4	Remote procedure error	17	(Any allowed)		Protocol error, unspecified	111	(None)

NOTE – Instead of providing the above mapping of X.25 to Q.931, the PH, as a network option, may code the Q.931 Cause information element to indicate *ITU-T Coding Standard* in octet 3, X.25 in octet 3a, and code octets 4 and 5 according to Recommendation X.25, copying the cause from the X.25 *clear indication* packet rather than mapping it to a Q.931 cause.

7.2 Call establishment

Procedures for call establishment are as described in 5.1 and 5.2 with the following modifications.

On call request, the calling user sends a SETUP message identifying, within the Bearer capability and Channel identification information elements, a temporary signalling connection to be established on SAPI = 0. The SETUP message is encoded to indicate:

i) *Bearer capability information element*
   - Unrestricted digital information in the information transfer capability field;
   - Packet mode in the transfer mode field;
   - User information layer 2 protocol is Recommendation Q.921 and user information layer 3 protocol is this Recommendation in the layer and protocol identification field.

ii) *Channel identification information element*
   - Exclusive in the preferred/exclusive field;
   - D-channel in the D-channel indicator field;
   - no channel in the channel selection field.

If the network determines that the requested temporary signalling connection service is not authorized or is not available, the network shall initiate call clearing in accordance with 5.3.2 a) or 5.3.2 c) with one of the following causes:

a) No. 57 – *Bearer capability not authorized*;

b) No. 58 – *Bearer capability not presently available*;

c) No. 63 – *Service or option not available, unspecified*; or

d) No. 65 – *Bearer capability not implemented*.

The called user accepts the temporary signalling connection request by sending a CONNECT message towards the calling user. After the called user has received a CONNECT ACKNOWLEDGE message, it may begin sending USER INFORMATION messages. Once the calling user receives a CONNECT message, it can begin sending USER INFORMATION messages.

7.3 Transfer of USER INFORMATION messages

Once a temporary signalling connection is established, both users can transfer information between themselves by transferring USER INFORMATION messages across the user-network interface. The network provides for the transfer of such messages from the called to the calling side and vice versa.

The USER INFORMATION message includes the Call reference, the Protocol discriminator, and the User-to-user information elements as defined in 3.3.13. The More data information element may also be sent by the source user to indicate to the remote user that another USER INFORMATION message will follow, containing information belonging to the same block. The use of the More data information element is not supervised by the network.

7.4 Congestion control of USER INFORMATION messages

The network or user will flow-control, when needed, the transfer of USER INFORMATION messages from a user or network by means of a CONGESTION CONTROL message containing a congestion level information element. Two indications of congestion level are specified: "receive not ready" and "receive ready". On receipt of the former, the user or network should suspend sending USER INFORMATION messages; on receipt of the latter, sending may recommence. After having sent a "receive not ready" indication, the network or user shall discard USER INFORMATION messages which are subsequently received. The network or user will send a CONGESTION

CONTROL message with a "receive not ready" indication whenever a USER INFORMATION message is locally discarded, if it is possible. The CONGESTION CONTROL message shall also include a cause No. 43, *access information discarded*.

The network shall notify the user that flow control restriction has been removed by sending a CONGESTION CONTROL message with the congestion level specified as "receive ready" to indicate that further messages may be sent. This message may be sent, as an implementation option:

a) immediately upon removal of the flow control restriction;

b) in response to the first USER INFORMATION message received following the removal of the flow control restriction; or

c) in both cases.

The receipt of the "receive ready" indication shall be interpreted as an indication that no more than *n* USER INFORMATION messages may be sent before another receive ready indication is received.

In each direction a Burst Capability of sending *n* messages is immediately available where *n* initially equals the value of the Burst parameter *x*. The value of *n* shall be decremented by one for every message sent by the user and incremented by y at regular intervals of T (T = 10 seconds) subject to the limitation that n may not exceed *x*, i.e. $n + y \leq x$.

The Burst parameter *x* is a variable which shall be set to a value of $x = 16$.

The replenishment parameter *y* shall be capable of taking a value $y = 8$.

NOTE – While some networks may support higher values of *x* and *y*, the value of *x* and *y* across international interfaces shall be set as above. It is up to the network using higher values to take the appropriate actions, unless bilateral agreements exist.

If USER INFORMATION messages are received at a rate which exceeds the flow control limit set by the network, the network shall discard the messages that cannot be handled and respond to the first discarded message with a control indication. The network shall also respond to the first USER INFORMATION message received following the removal of flow control restriction by returning an indication that further messages may be sent.

The Congestion control procedure should be regarded as local. Congestion control procedure for remote applications is for further study.

7.5 Call clearing

The clearing of an established temporary signalling connection can be initiated by the user or network by sending a RELEASE message towards the far end user. The clearing procedure followed and the timers involved are the same as those for clearing a circuit-switched connection as described in 5.3.3 and 5.3.4.

7.6 Handling of error conditions

In the event of a data link reset or failure, all temporary signalling connections on the D-channel shall be released as in 7.5. For data link resets, the clearing messages shall indicate cause No. 41, *temporary failure*, to both local and remote users. For data link failures, the clearing message to the remote user shall indicate cause No. 27, *destination out of order*, and the local temporary signalling connection shall be cleared internally.

7.7 Restart procedures

Handling of restart for temporary signalling connections shall be as described in 5.5.2. If a RESTART message is received with the Restart indicator information element coded as "all interfaces", or coded as "single interface" and the indicated interface includes the D-channel, then all temporary signalling connections on the D-channel shall be released. During restart the clearing message to the remote users shall include cause No. 41, *temporary failure*.

8 Circuit-mode multirate (64 kbit/s base rate) procedures

This clause provides the D-channel signalling procedures in support of circuit-mode multirate (64 kbit/s base rate) bearer capability.

These procedures are mandatory when a supported bearer capability or teleservice requires a multirate (64 kbit/s base rate) bearer capability, else they are not required.

The procedures of clause 5 shall apply except as identified in the following subclauses.

8.1 Call establishment at the originating interface

8.1.1 Compatibility information

The Bearer capability information element shall be encoded as in 4.5.5 (Bearer capability) with the following exceptions:

1) Octet 3 shall be coded *unrestricted digital information*.

2) Octet 4 shall be coded *circuit-mode* and the information transfer rate (bits 5 to 1) shall be encoded:

Bits	
5 4 3 2 1	Circuit-mode
1 1 0 0 0	Multirate (64 kbit/s base rare)

3) Octet 4.1 (Rate multiplier) shall be included. Bit 8 is for extension and set to 1. Bits 7-1 contain the binary coding of the multiplier that applies to the multirate codepoint contained in the information transfer rate subfield. Bit 1 is least significant. The multiplier value range is 2-30, all other values are reserved. Octet 4.1 shall be included if and only if the transfer rate is coded for multirate.

NOTE – When the information transfer rate is 384 kbit/s, 1536 kbit/s, or 1920 kbit/s the information transfer rate in the Bearer capability information element may also be coded as 384 kbit/s (10011), 1536 kbit/s (10101), or 1920 kbit/s (10111) respectively instead of using the multirate (64 kbit/s base rate) codepoint and the associated rate multiplier field.

8.1.2 Channel selection

The channels selected for the multirate call shall be on one interface and shall be indicated in the SETUP message. The procedures in 5.1.2 and 5.2.3.1 shall be followed to complete the channel selection.

The Channel identification information element is coded as per 4.5.13.

The number of channels identified shall provide the information transfer rate identified in the Bearer capability information element. If the information transfer rate implied by the channel(s) or interface indicated in the Channel identification information element does not match the information transfer rate in the Bearer capability information element, the procedures in 5.8.6.2 shall apply.

Channel selection conflict occurs when the channels selected for an incoming and outgoing call do not constitute two disjoint sets of time slots. When channel selection conflict occurs, the procedures in 5.7 shall apply.

Some networks may offer on access:

1) contiguous channel assignment (channels must be adjacent within a single interface); and/or

   NOTE 1 – On a 2048 kbit/s interface (containing a D-channel), channels 15 and 17 shall be considered contiguous.

2) non-contiguous channel assignment (channels may be either adjacent or non-adjacent within a single interface).

Some networks may require that 384 kbit/s and/or 1536 kbit/s (in a 2048 kbit/s interface) occupy specified contiguous time slots (see Annex A/I.431).

If the entire interface of a primary rate interface is used (i.e. 24 B-channels on a 1544 kbit/s interface or 30 B-channels on a 2048 kbit/s interface), octets 3.2 and 3.3 of the Channel identification information element shall not be included.

If the entire interface of a basic access interface is used (i.e. 2 B-channels), octets 3.2 and 3.3 of the Channel identification information element shall not be included and the "information channel selection" shall be coded "11", *any channel*.

In cases a) and b) of 5.1.2, if all the indicated B-channels are available, the network shall select them for the call.

In case b) of 5.1.2, if the network cannot grant any preferred B-channel, it shall select any other available B-channels associated with the D-channel and on the same access, to replace the unavailable preferred B-channel, or select all the B-channels on another interface controlled by the D-channel.

NOTE 2 – Whether only the B-channels that cannot be provided should be changed, or if all the B-channels can then be changed, is for further study.

In case c) of 5.1.2, the network shall select any available suitable B-channels.

In case a) of 5.1.2, if any specified B-channel is not available, and in cases b) and c) of 5.1.2 if insufficient B-channels are available, the network shall send a RELEASE COMPLETE message with cause No. 44, *requested circuit/channel not available* or No. 34, *no circuit/channel available*, respectively, as described in 5.3.

The following recommendations are made on the use of cause values:

1) When the calling user or called user is not an authorized subscriber of the multirate circuit-mode bearer capability, cause No. 57, *bearer capability not authorized*, shall be returned to the calling user.

2) When a network (public or private) cannot support the specified transfer rate or bearer capability, cause No. 65, *bearer capability not implemented*, shall be returned to the calling user.

3) When there are insufficient channels on a single interface to support the information transfer rate requested, cause No. 34, *no circuit/channel available*, or cause No. 17, *user busy*, shall be returned to the calling user (see Recommendation Q.850 which is also reproduced in Appendix I).

8.1.3 Interworking

Interworking is possible between:

1) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 64 kbit/s unrestricted circuit-mode service when the information transfer rate is 64 kbit/s.
2) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 384 kbit/s unrestricted circuit-mode service when the information transfer rate is 384 kbit/s.
3) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 1536 kbit/s unrestricted circuit-mode service when the information transfer rate is 1536 kbit/s.
4) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 1920 kbit/s unrestricted circuit-mode service when the information transfer rate is 1920 kbit/s.

When any other information transfer rate is specified, interworking is not possible between the multirate circuit-mode bearer capability and other services.

8.2 Call establishment at the destination interface

8.2.1 Compatibility information

The Bearer capability information element shall be encoded as in 4.5.5 (Bearer capability) with the following exceptions:

1) Octet 3 shall be coded *unrestricted digital information*.
2) Octet 4 shall be coded *circuit-mode* and the information transfer rate (bits 5 to 1) shall be encoded:

Bits	
5 4 3 2 1	Circuit-mode
1 1 0 0 0	Multirate (64 kbit/s base rare)

3) Octet 4.1 (Rate multiplier) shall be included. Bit 8 is for extension and set to 1. Bits 7-1 contain the binary coding of the multiplier that applies to the multirate codepoint contained in the information transfer rate subfield. Bit 1 is least significant. The multiplier value range is 2-30, all other values are reserved. Octet 4.1 shall be included if and only if the transfer rate is coded for multirate.

NOTE – When the information transfer rate is 384 kbit/s, 1536 kbit/s, or 1920 kbit/s the information transfer rate in the Bearer capability information element may also be coded as 384 kbit/s (10011), 1536 kbit/s (10101), or 1920 kbit/s (10111) respectively instead of using the multirate (64 kbit/s base rate) codepoint and the associated rate multiplier field.

8.2.2 Channel selection

The channels selected for the multirate call shall be on one interface and shall be indicated in the SETUP message. The procedures in 5.1.2 and 5.2.3.1 shall be followed to complete the channel selection.

The Channel identification information element is coded as per 4.5.13.

The number of channels identified shall provide the information transfer rate identified in the Bearer capability information element. If the information transfer rate implied by the channel(s) or interface

indicated in the Channel identification information element does not match the information transfer rate in the Bearer capability information element, the procedures in 5.8.6.2 shall apply.

Channel selection conflict occurs when the channels selected for an incoming and outgoing call do not constitute two disjoint sets of time slots. When channel selection conflict occurs, the procedures in 5.7 shall apply.

Some networks may offer on access:

1) contiguous channel assignment (channels must be adjacent within a single interface); and/or

   NOTE – On a 2048 kbit/s interface (containing a D-channel), channels 15 and 17 shall be considered contiguous.

2) non-contiguous channel assignment (channels may be either adjacent or non-adjacent within a single interface).

Some networks may require that 384 kbit/s and/or 1536 kbit/s (in a 2048 kbit/s interface) occupy specified contiguous time slots (see Annex A/I.431).

If the entire interface of a primary rate interface is used (i.e. 24 B-channels on a 1544 kbit/s interface or 30 B-channels on a 2048 kbit/s interface), octets 3.2 and 3.3 of the Channel identification information element shall not be included.

If the entire interface of a basic access interface is used (i.e. 2 B-channels), octets 3.2 and 3.3 of the Channel identification information element shall not be included and the "information channel selection" shall be coded "11", *any channel*.

The following recommendations are made on the use of cause values:

1) When a network (public or private) cannot support the specified transfer rate or bearer capability, cause No. 65, *bearer capability not implemented*, shall be returned to the calling user.

2) When the calling user attempts to set up call to a user who has not subscribed to the multirate service, the network will initiate call clearing and return cause No. 57, *bearer capability not authorized* to the calling user.

3) When the number of channels subscribed to on a single interface is sufficient to support the call as requested, but there are insufficient free channels, cause No. 17, *user busy,* is returned to the calling user. However, if the number of channels subscribed to on a single interface is insufficient to support the call as requested, cause No. 65, *bearer capability not implemented*, is returned to the calling user.

8.2.2.1 Point-to-point configuration

In cases 1) and 2) of 5.2.3.1 if all the indicated traffic channels are available, the user shall select them for the call.

In case 2), if the user cannot grant any referred access channel, it shall select any other available access channels associated with the D-channel and on the same access, to replace the unavailable preferred access channel, or select all the channels on another interface controlled by the D-channel.

NOTE – Whether only the B-channels that cannot be provided should be changed, or if all the channels can then be changed, is for further study.

In case 3) of 5.2.3.1 the user shall select any available suitable access channels.

In case 1) of 5.2.3.1 if any specified access channel is not available, and in cases 2) and 3) if insufficient access channels are available, the user shall send a RELEASE COMPLETE message with cause No. 44, *requested circuit/channel not available*, or No. 34, *no circuit/channel available*, respectively, as described in 5.3.

8.2.2.2 Point-to-multipoint configuration

In case a) of 5.2.3.2, if all the indicated traffic channels are available, the user shall select them for the call.

8.2.3 Interworking

Interworking is possible between:

1) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 64 kbit/s unrestricted circuit-mode service when the information transfer rate is 64 kbit/s.
2) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 384 kbit/s unrestricted circuit-mode service when the information transfer rate is 384 kbit/s.
3) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 1536 kbit/s unrestricted circuit-mode service when the information transfer rate is 1536 kbit/s.
4) A user who has subscribed to the multirate circuit-mode bearer capability and a user who has subscribed to the 1920 kbit/s unrestricted circuit-mode service when the information transfer rate is 1920 kbit/s.

When any other information transfer rate is specified, interworking is not possible between the multirate circuit-mode bearer capability and other services.

8.3 Call clearing

When the call is cleared, by the user or by the network, all channels associated with the call shall be cleared.

8.4 Restart procedures

B-channels can be restarted irrespective of their usage within a multirate bearer capability. If a single B-channel is restarted, the Q.931 entity shall clear the call.

8.5 Call rearrangements

The procedures of 5.6 do not apply.

9 List of system parameters

The description of timers in the following tables should be considered a brief summary. The precise details are found in clauses 5 and 6, which should be considered the definitive descriptions.

9.1 Timers in the network side

The timers specified in Table 9-1 are maintained in the network side of the interface.

9.2 Timers in the user side

The timers specified in Table 9-2 are maintained in the user side of the interface. Timers T305, T308 and T313 are mandatory for all user side implementations.

Table 9-1/Q.931 – Timers in the network side

Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross-reference
T301	Minimum 3 min	Call received	ALERT received	CONNECT received	Clear call	Timer is not restarted	(Note 2)
T302	10-15 s (Note 5)	Overlap sending	SETUP ACK sent Receipt of INFO, restarts T302	With sending complete indication, or network alert, or connect request received	Clear if call information determined to be definitely incomplete; else send CALL PROC	Timer is not restarted	Mandatory
T303	4 s (Note 1)	Call present	SETUP sent	ALERT, CONNECT CALL PROC or SETUP ACK received, REL COMPLETE received if SETUP sent on point-point data link	Retransmit SETUP; restart T303. If REL COMPLETE has been received, clear the call	Clear network connection. Enter call abort state	Mandatory
T304	20 s (provisional value)	Overlap receiving	SETUP ACK received. Sending of INFO restarts T304	Send INFO; receive CALL PROC, ALERT or CONNECT	Clear the call	Timer is not restarted	Mandatory only if 5.2.4 implemented
T305	30 s	Disconnect indication	DISC without progress indicator No. 8 sent	REL or DISC received	Network sends REL	Timer is not restarted	Mandatory
T306	30 s (Note 6)	Disconnect indication	DISC with progress indicator No. 8 sent	REL or DISC received	Stop the tone/announcement. Send REL	Timer is not restarted	Mandatory when in-band tones/announcements are provided; see 5.4, 5.3.4.1, and Recs. I.300-series
T307	3 min	Null	SUSPEND ACK sent	RES ACK sent	Clear the network connection. Release call identity	Timer is not restarted	Mandatory

Table 9-1/Q.931 – Timers in the network side *(continued)*

Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross-reference
T308	4 s (Note 1)	Release request	REL sent	REL COMPLETE or REL received	Retransmit REL and restart T308	Place B-channel in maintenance condition. Release call reference (Note 9)	Mandatory
T309	6-90 s (Note 10)	Any stable state	Data link disconnection. Calls in stable states are not lost	Data link reconnected	Clear network connection. Release B-channel and call reference	Timer is not restarted	Mandatory
T310	10 s (Note 7)	Incoming Call Proceeding	CALL PROC received	ALERT, CONNECT or DISC received. If DISC, retain cause and continue timing	Clear call in accordance with 5.2.5.3	Timer is not restarted	Mandatory
T312	T303 + 2 s	Call Present, Call Abort, etc.	SETUP sent or resent on broadcast data link	Timeout	(Note 4)	Timer is not restarted	Mandatory
T314	4 s	Receiving segmented message	Message segment received	Last message segment received	Discard message	Timer is not restarted	Mandatory, see Annex H
T316	2 min	Restart request	RESTART sent	RESTART ACK received	RESTART may be retransmitted several times	RESTART may be retransmitted several times	Mandatory when 5.5 is implemented
T317	(Note 3)	Restart	RESTART received	Internal clearing of call references	Maintenance notification	Timer is not restarted	Mandatory when 5.5 is implemented
T320	30 s (Note 8)	a) For B-channel access: active b) For D-channel access: null	a) For B-channel access: connection b) For D-channel access: DL-ESTABLISH Confirmation or DL-ESTABLISH indication received c) Last logical channel, cleared received	Call request packet received; or incoming call packet delivered; or DISC received; or for D-channel access DL-RELEASE indication received	a) For B-channel access: disconnect link layer and initiate clearing b) For D-channel access: send DL-RELEASE request	Timer is not restarted	Optional. See 6.3

Table 9-1/Q.931 – Timers in the network side *(concluded)*

Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross-reference
T321	30 s	Any call state	D-channel failure	Response to layer 3 message received	Send DL-ESTABLISH request on both D-channels	Timer is not restarted	Mandatory when Annex F is implemented
T322	4 s	Any all state	STATUS ENQ sent	STATUS DISC REL or REL COMPLETE received	STATUS ENQ may be retransmitted several times	STATUS ENQ may be retransmitted several times	Mandatory when 5.8.10 is implemented

NOTE 1 – This default value assumes the use of default values at layer 2, i.e. [N200 + 1] times T200. Whether these values should be modified when layer 2 default values are modified by an automatic negotiation procedure is for further study.

NOTE 2 – The network may already have applied an internal alerting supervision timing function, e.g. incorporated within call control. If such a function is known to be operating on the call, then timer T301 is not used.

NOTE 3 – The value of this timer is implementation dependent but should be less than the value of T316.

NOTE 4 – If in the call abort state, the call reference is released. Otherwise, no action is taken on expiry of timer T312.

NOTE 5 – The value of timer T302 may vary beyond these limits, e.g. as a result of called party number analysis.

NOTE 6 – The value of this timer T306 is network dependent.

NOTE 7 – The value of timer T310 may be different in order to take into account the characteristics of a private network.

NOTE 8 – This value may vary by network-user agreement.

NOTE 9 – The restart procedures contained in 5.5 may be used on B-channels in the maintenance condition.

NOTE 10 – The value of this timer is network dependent.

Table 9-2/Q.931 – Timers in the user side

Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross-reference
T301	Minimum 3 min.	Call Delivered	ALERT received	CONNECT received	Clear call	Timer is not restarted	Mandatory when Annex D is implemented (Note 3)
T302	15 s	Overlap receiving	SETUP ACK sent Restart when INFO received	INFO received with sending complete indication; or internal alerting; or internal connection; or a determination that sufficient information has been received	Clear if call information determined to be incomplete; else send CALL PROC	Timer is not restarted	Mandatory only if 5.2.4 is implemented
T303	4 s (Note 1)	Call Initiated	SETUP sent	ALERT (Annex D), CONNECT (Annex D), SETUP ACK, CALL PROC or REL COMPLETE received	Retransmit SETUP; restart T303. If REL COMPLETE was received, clear the call (Annex D)	Clear internal connection. Send REL COMPLETE. Enter Null state	Mandatory when Annex D is implemented; otherwise optional
T304	30 s	Overlap Sending	INFO sent Restarted when INFO sent again	CALL PROC, ALERT, CONNECT, DISC or prog. ind. 1 or 2 received	DISC sent	Timer is not restarted	Optional
T305	30 s	Disconnect Request	DISC sent	REL or DISC received	REL sent	Timer is not restarted	Mandatory
T308	4 s (Note 1)	Release request	REL sent	REL COMPLETE or REL received	Retransmit REL; and restart T308	B-channel is placed in maintenance condition. Call reference released (Note 5)	Mandatory
T309	6-90 s (Note 6)	Any stable state	Data link disconnection. Calls in stable states are not lost	Data link reconnected	Clear internal connection. Release B-channel and call reference	Timer is not restarted	Optional
T310 (Note 4)	30-120 s	Outgoing Call Proceeding	CALL PROC received	ALERT, CONNECT, DISC, or PROGRESS received	Send DISC	Timer is not restarted	Mandatory when Annex D is implemented
T313	4 s (Note 1)	Connect request	CONNECT sent	CONNECT ACK received	Send DISC	Timer is not restarted	Mandatory

Table 9-2/Q.931 – Timers in the user side *(concluded)*

Timer number	Default time-out value	State of call	Cause for start	Normal stop	At the first expiry	At the second expiry	Cross-reference
T314	4 s	Receiving Segmented Message	Message segment received	Last message segment received	Discard message	Timer is not restarted	Not initially required
T316	2 min	Restart Request	RESTART sent	RESTART ACK received	RESTART may be retransmitted several times	RESTART may be retransmitted several times	Mandatory when 5.5 is implemented
T317	(Note 2)	Restart	RESTART received	Internal clearing of call reference	Maintenance notification	Timer is not restarted	Mandatory when 5.5 is implemented
T318	4 s	Resume Request	RES sent	RES ACK or RES REJ received	Send RELEASE message with cause No. 102	Timer is not restarted	Mandatory when 5.6 is implemented
T319	4 s	Suspend Request	SUSPEND sent	SUSPEND ACK or SUSP REJ received	Enter Active state. Notify user application	Timer is not restarted	Mandatory when 5.6 is implemented
T321	30 s	Any call state	D-channel failure	Response to layer 3 message received	Send DL-ESTABLISH request on both D-channels	Timer is not restarted	Mandatory when Annex F is implemented
T322	4 s	Any call state	STATUS ENQ sent	STATUS, DISC, REL, REL COMPLETE received	STATUS ENQ may be retransmitted several times	STATUS ENQ may be retransmitted several times	Mandatory when 5.8.10 is implemented

NOTE 1 – This default value assumes the use of default values at layer 2, i.e. [N200 + 1] times T200. Whether these values should be modified when layer 2 default values are modified by an automatic negotiation procedure is for further study.

NOTE 2 – The value of this timer is implementation dependent, but should be less than the value of T316.

NOTE 3 – The user may already have applied an internal alerting supervision timing function, e.g. incorporated within call control. If such a function is known to be operating on the call, then timer T301 is not used.

NOTE 4 – T310 is not started if progress indicator 1 or 2 has been delivered in the CALL PROCEEDING message or in a previous PROGRESS message.

NOTE 5 – The restart procedures contained in 5.5 may be used on B-channels in the maintenance conditions.

NOTE 6 – The value of this timer is implementation dependent.

ANNEX A

User side and network side SDL diagrams

This Annex includes overview and detailed SDL diagrams which show Q.931 protocol control for circuits-switched basic calls. In the event of conflict between these diagrams and the text of clause 5, the text should be the prime source. Similarly, in the event of conflict between overview SDL and detailed SDL diagrams, the detailed SDL diagrams should be the prime source.

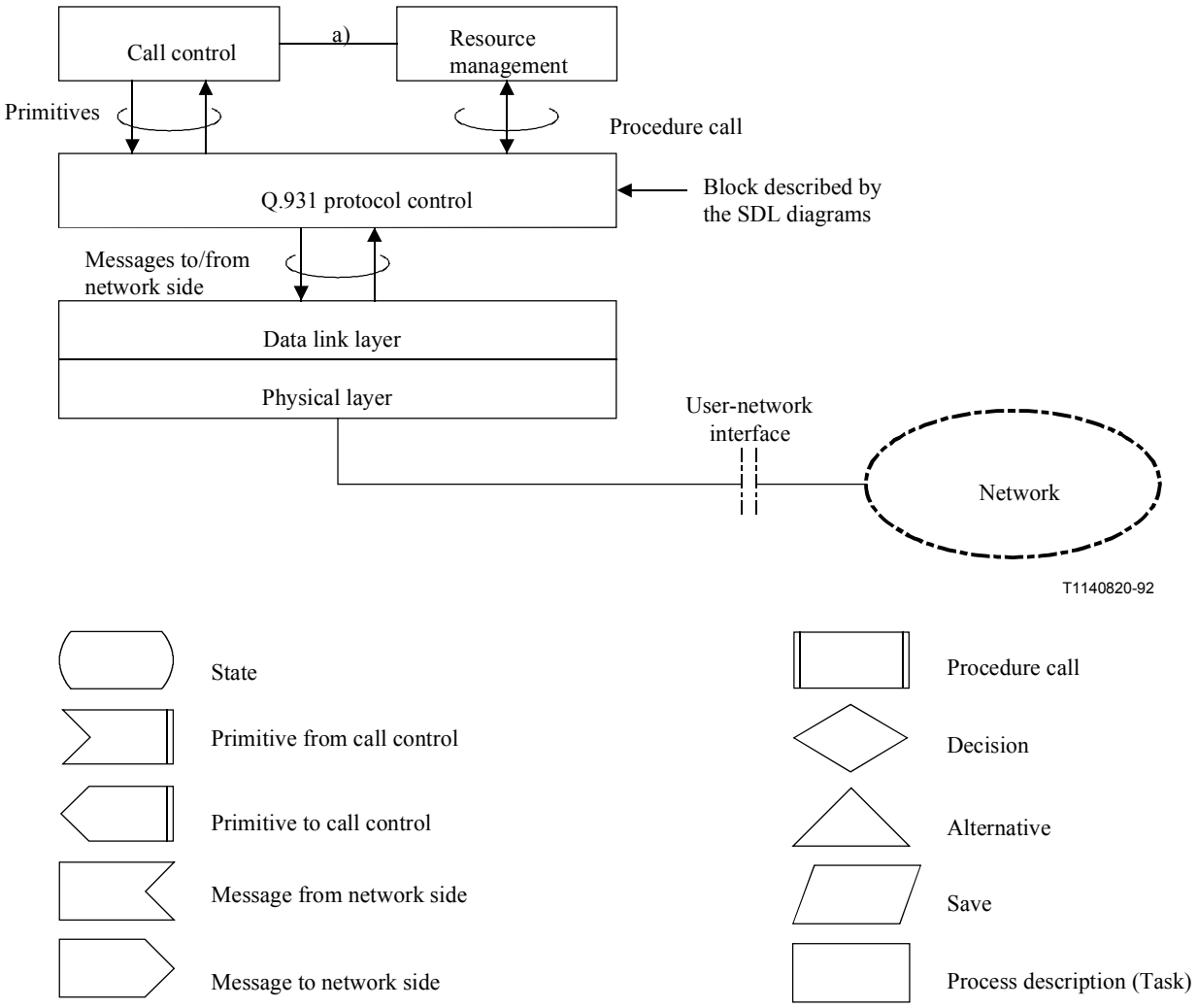
Figure A.1 shows key to Q.931 protocol control SDL diagrams for both user side and network side.

Figures A.2 and A.3 show overview and detailed protocol control SDL diagrams for the user side.

Figures A.5 and A.6 show overview and detailed protocol control SDL diagrams for network side. Only procedures for the point-to-point configuration are described in the network side SDL diagrams.

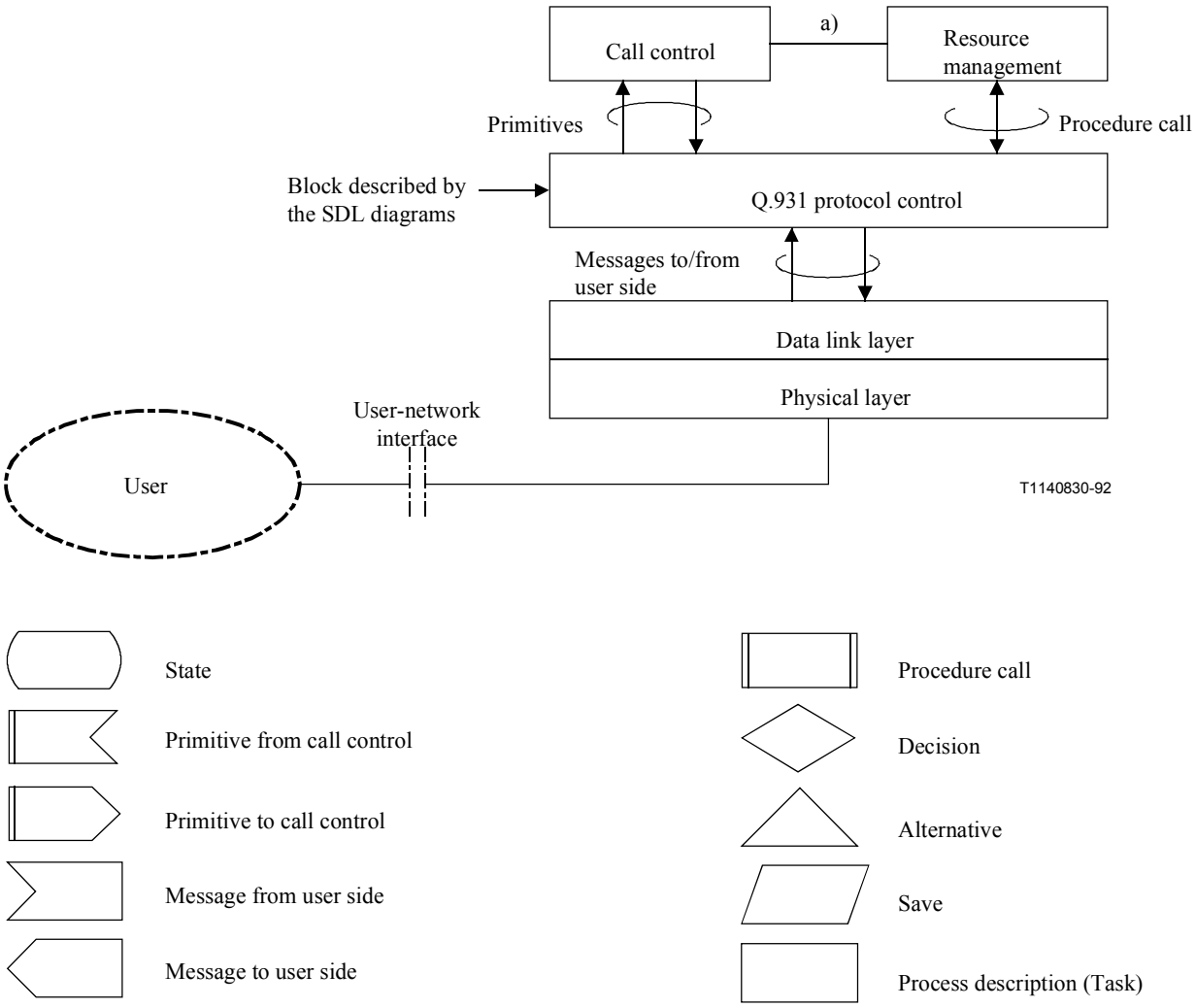
NOTE – Network side SDL diagrams for the point-to-multipoint configuration are left for further study.

Figure A.4 shows detailed SDL diagrams for the global call reference to be applied to both user and network sides. Although Figure A.4 shows SDL diagrams in the user-side only, the same diagrams can be applied to the network side by just changing the direction of input and output symbols.



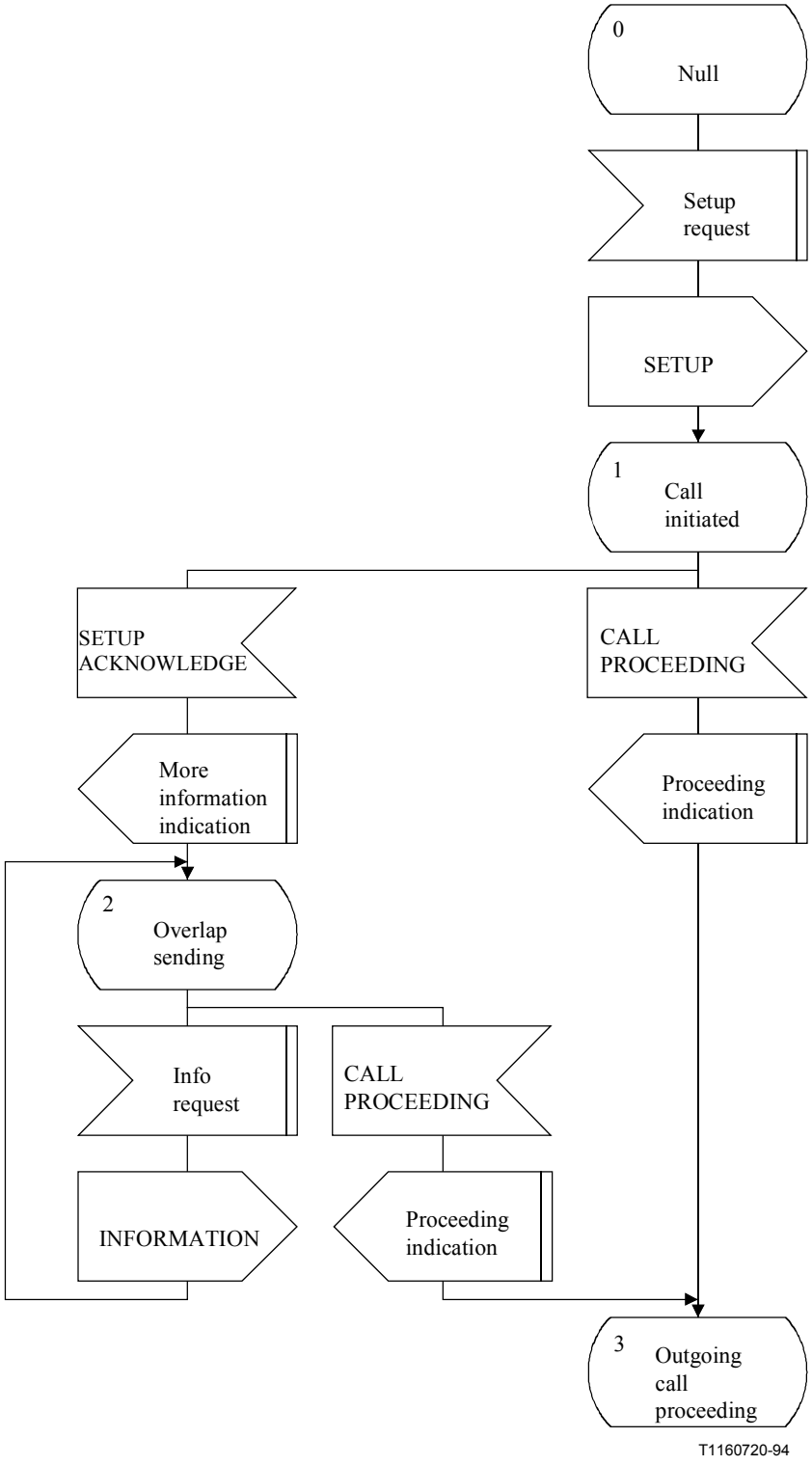
a) Not described in the SDL diagrams.

Figure A.1/Q.931 – Key to Q.931 protocol control SDL diagrams (user side) ***(sheet 1 of 2)***



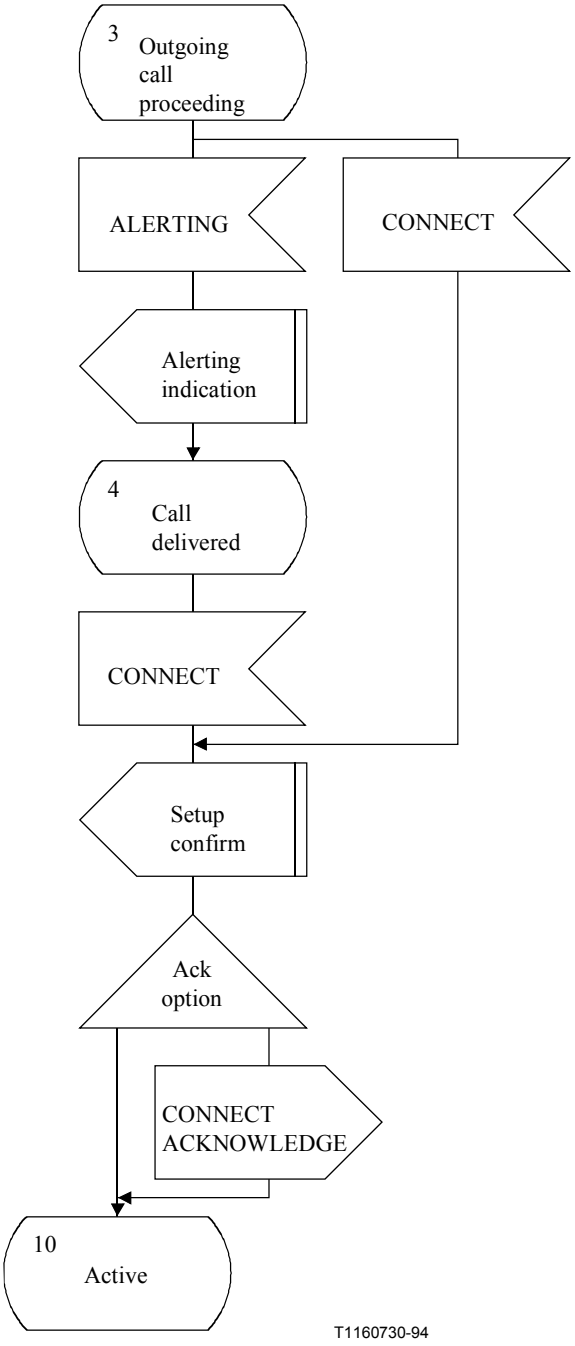
a) Not described in the SDL diagrams.

Figure A.1/Q.931 – Key to protocol control SDL diagrams (network side) ***(sheet 2 of 2)***



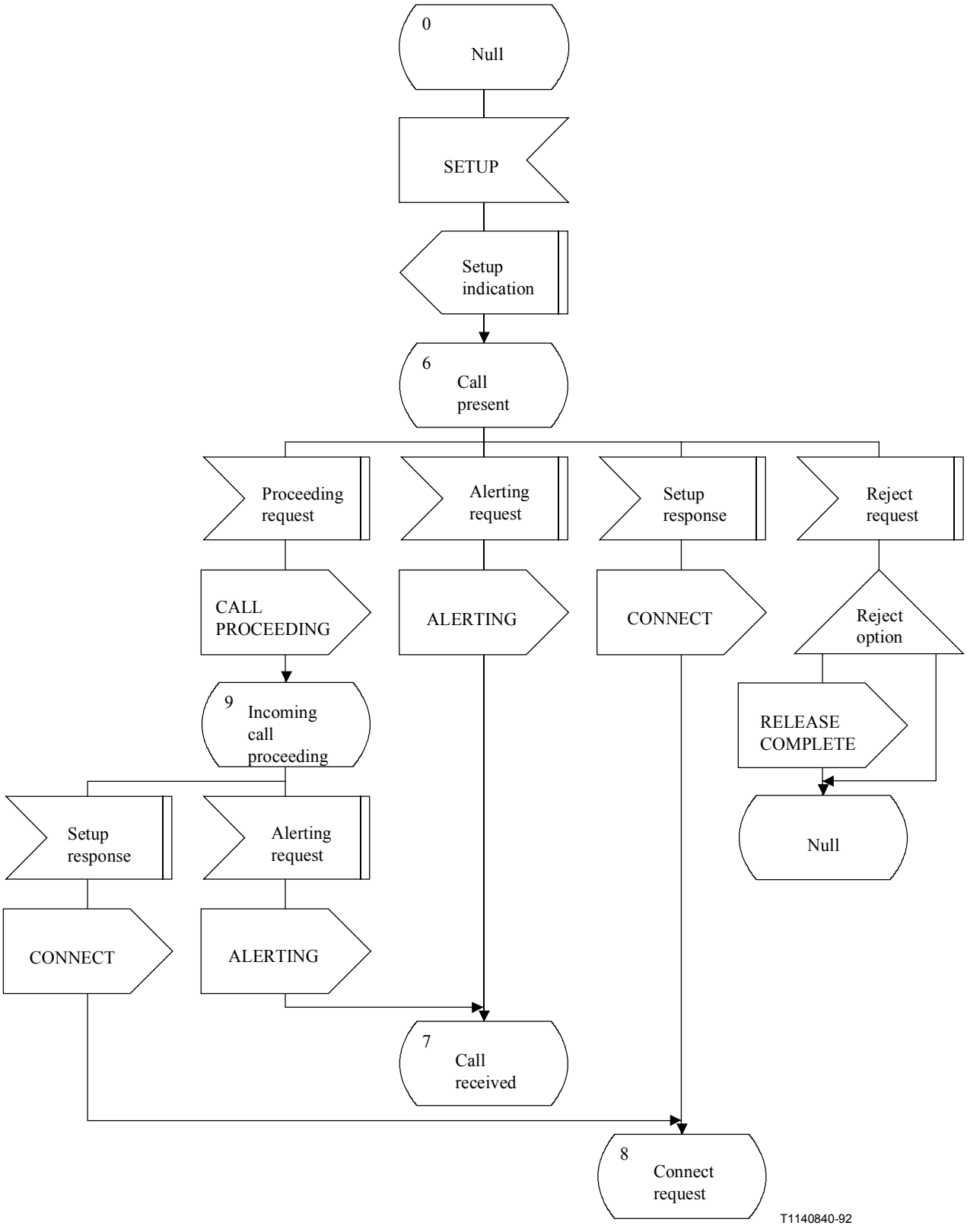
a) Outgoing set-up procedure (1 of 2)

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 1 of 7)***



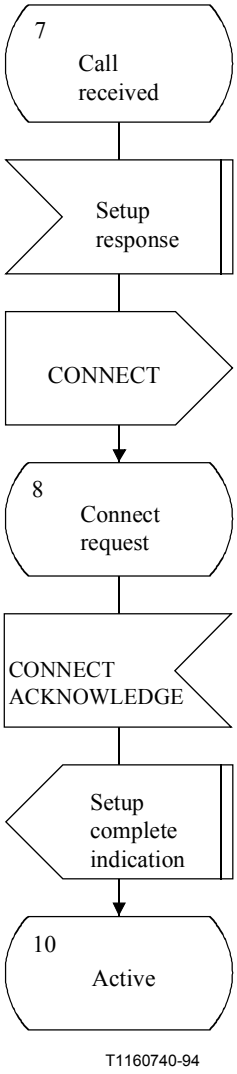
a) Outgoing set-up procedure (2 of 2)

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 2 of 7)***



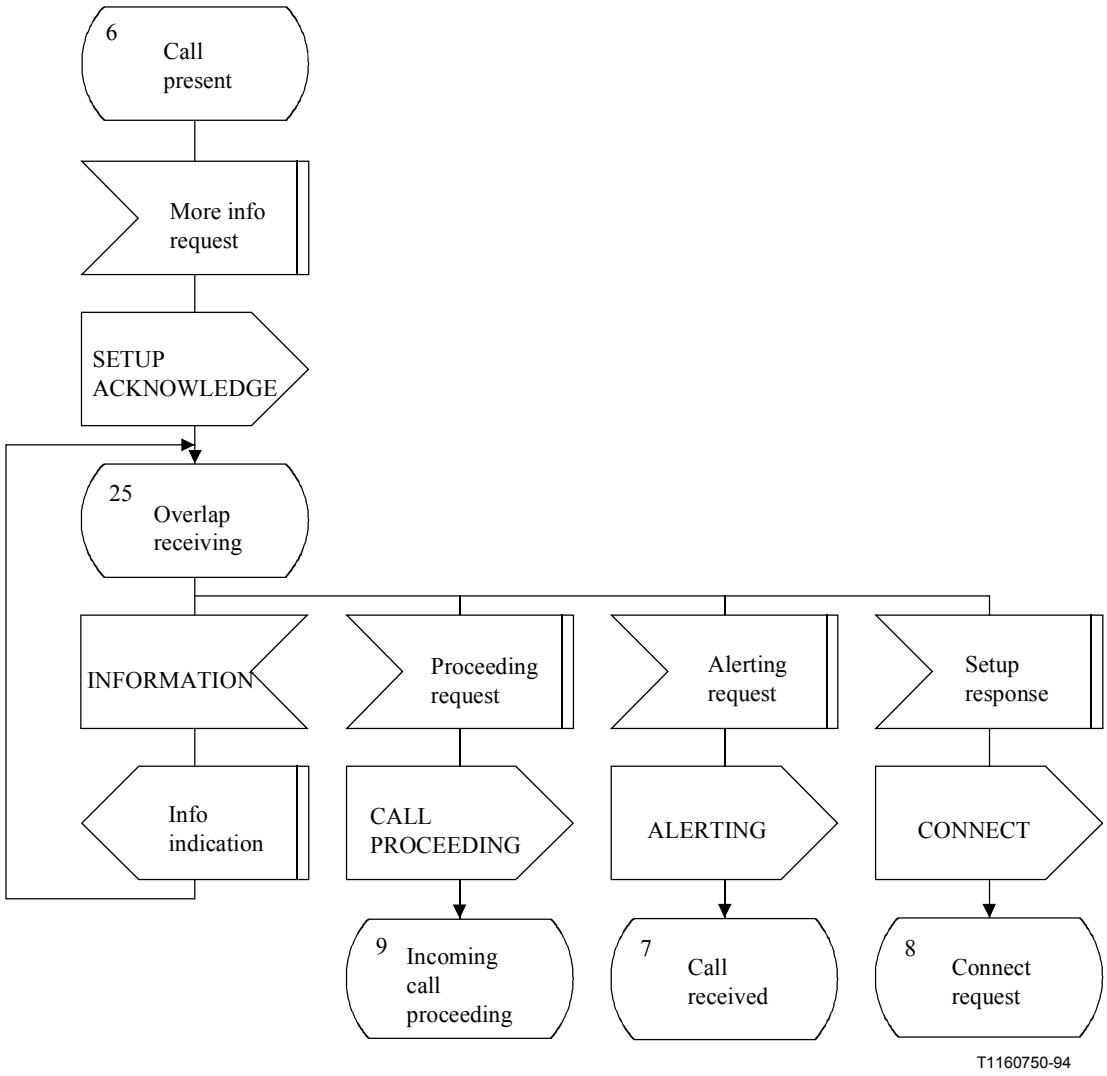
b) Incoming set-up procedure (1 of 2)

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 3 of 7)***



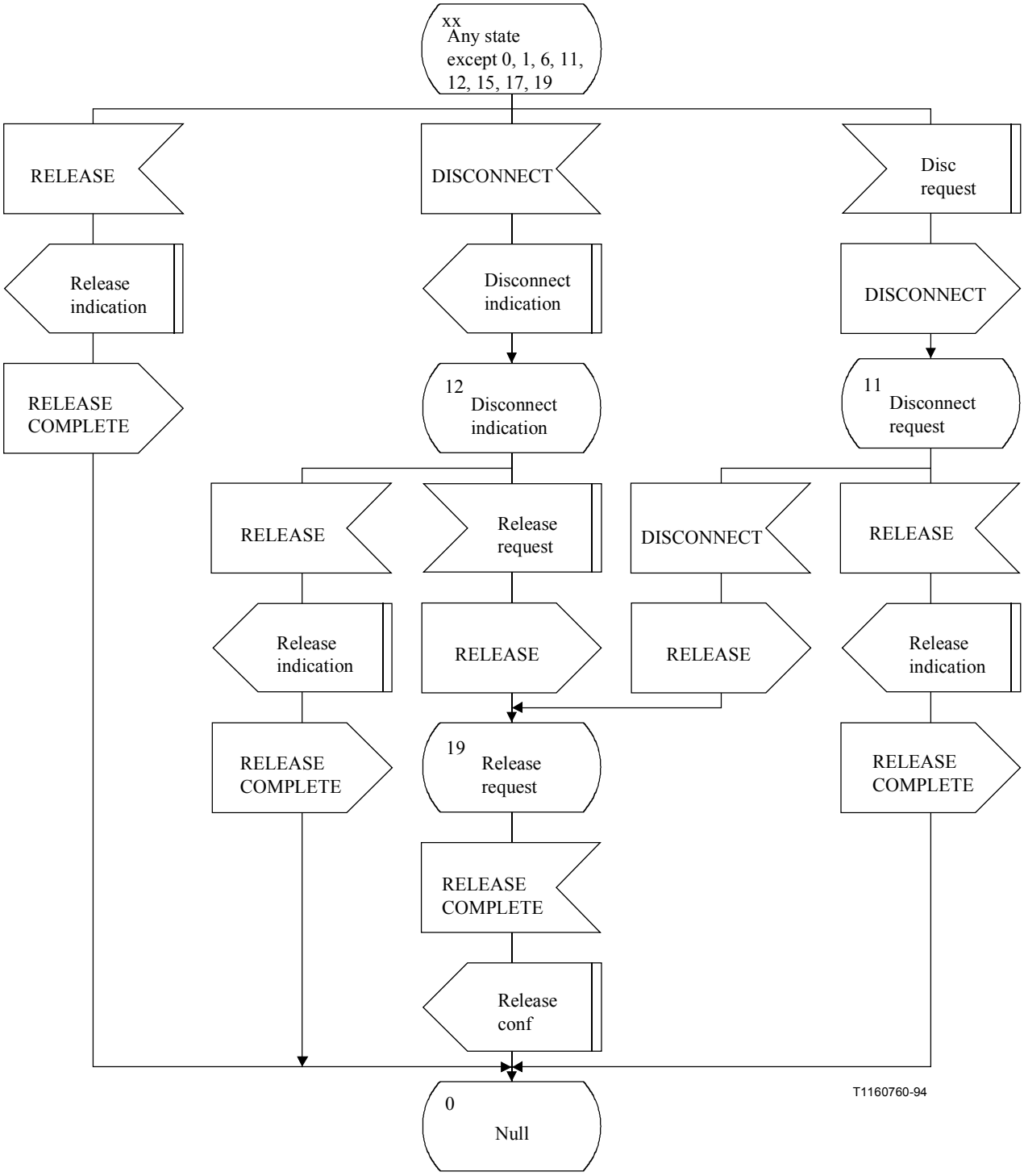
b) Incoming set-up procedure (2 of 2)

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 4 of 7)***



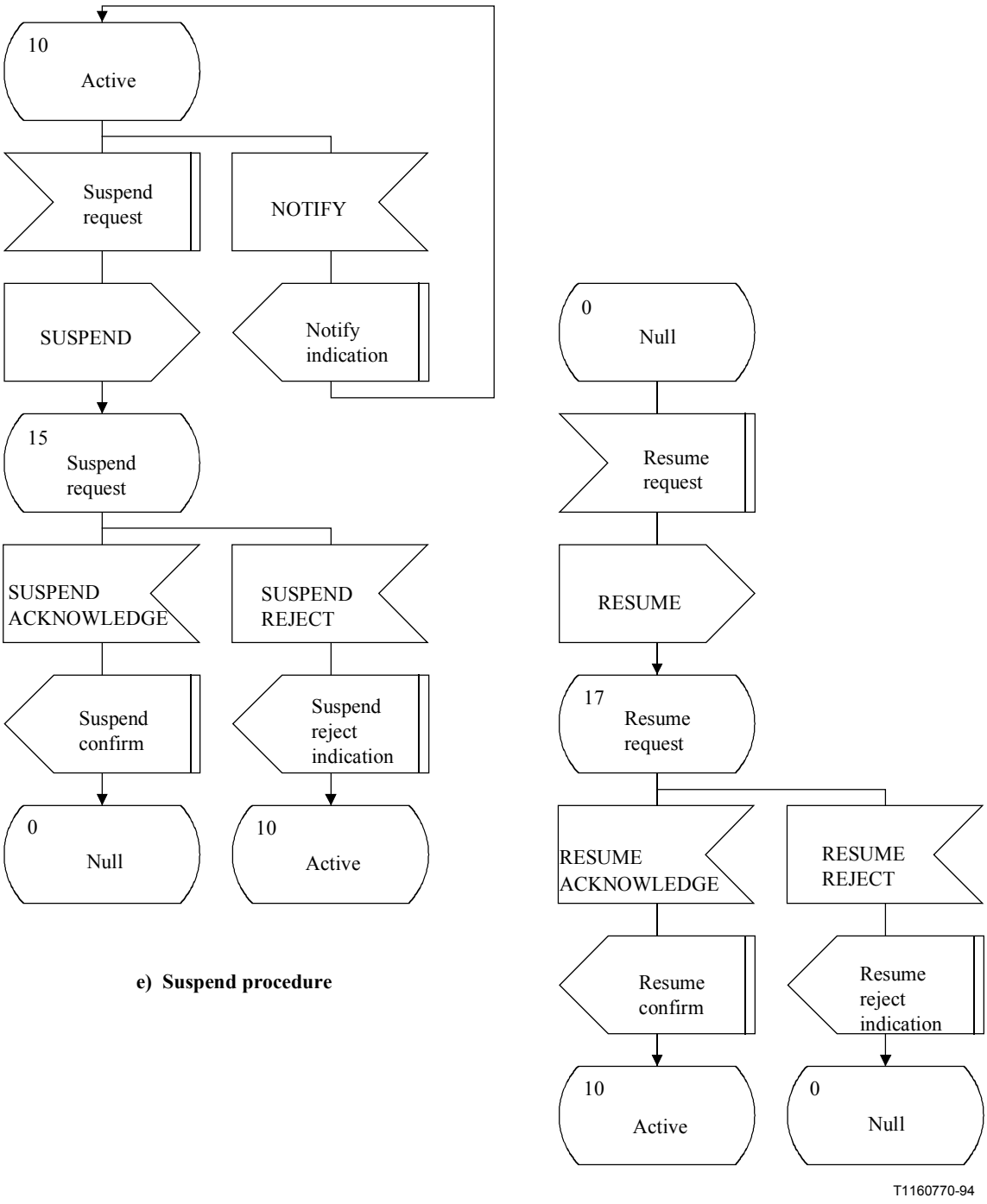
c) Overlap receiving procedure

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 5 of 7)***



d) Clearing procedure

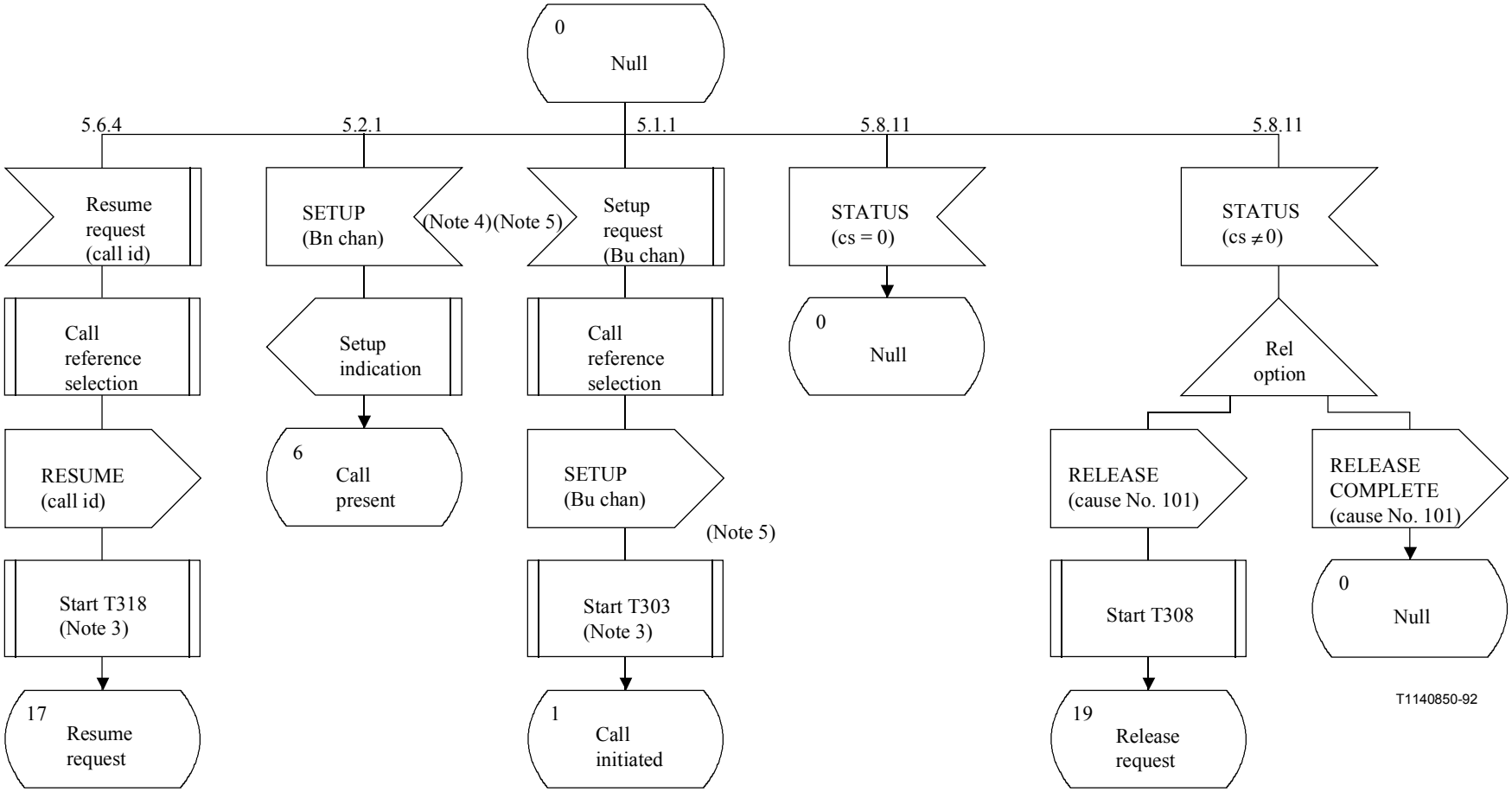
Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 6 of 7)***



e) Suspend procedure

f) Resume procedure

Figure A.2/Q.931 – Overview protocol control (user side) ***(sheet 7 of 7)***



NOTE 1 – In the event of conflict between these diagrams and the text of clause 5, the text should be the prime source.

NOTE 2 – These diagrams show Q.931 protocol control for circuit-switched calls.

NOTE 3 – T303 and T318 are optional (see 9.2).

NOTE 4 – "Bn chan" is a B-channel selected by the network.

NOTE 5 – "Bu chan" is a B-channel selected by the user.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 1 of 25)***

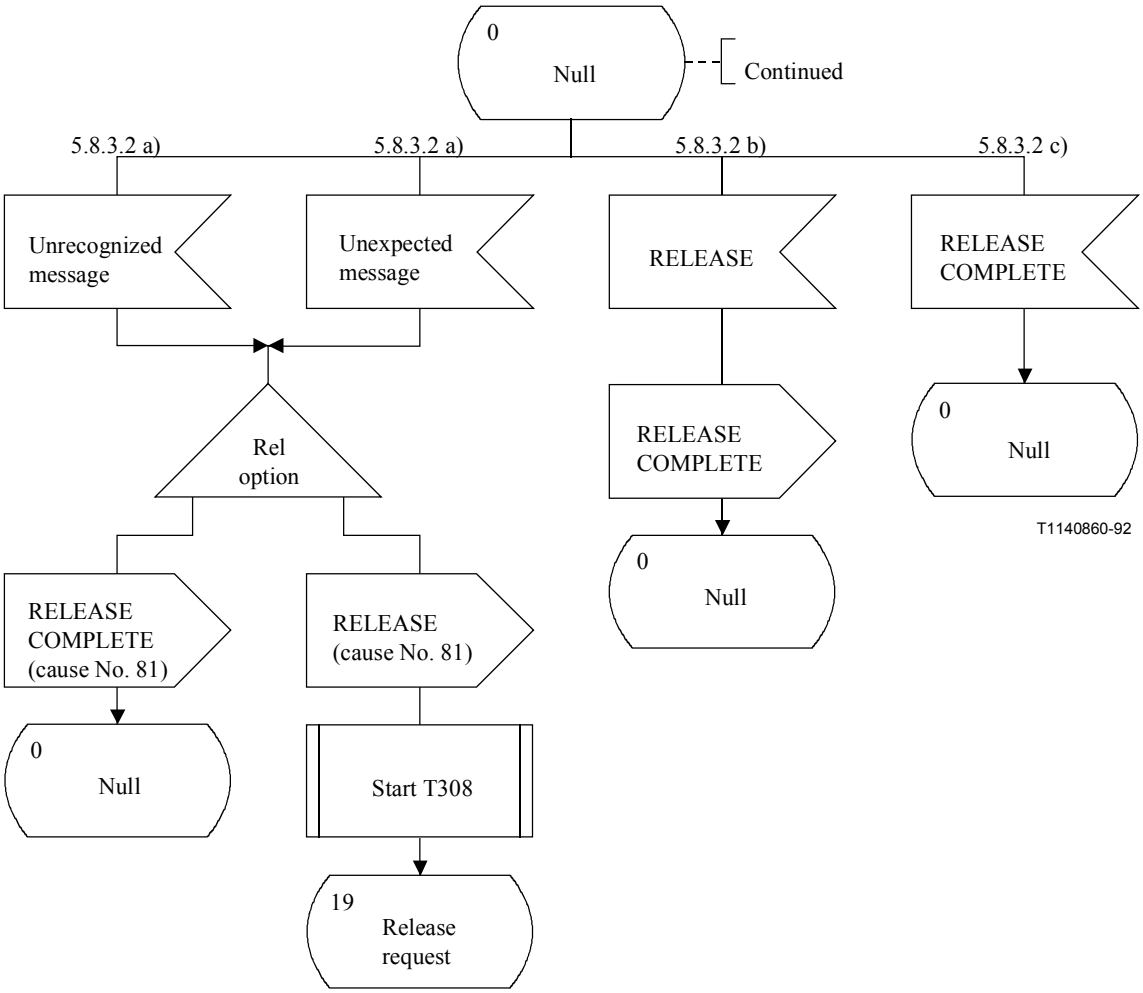
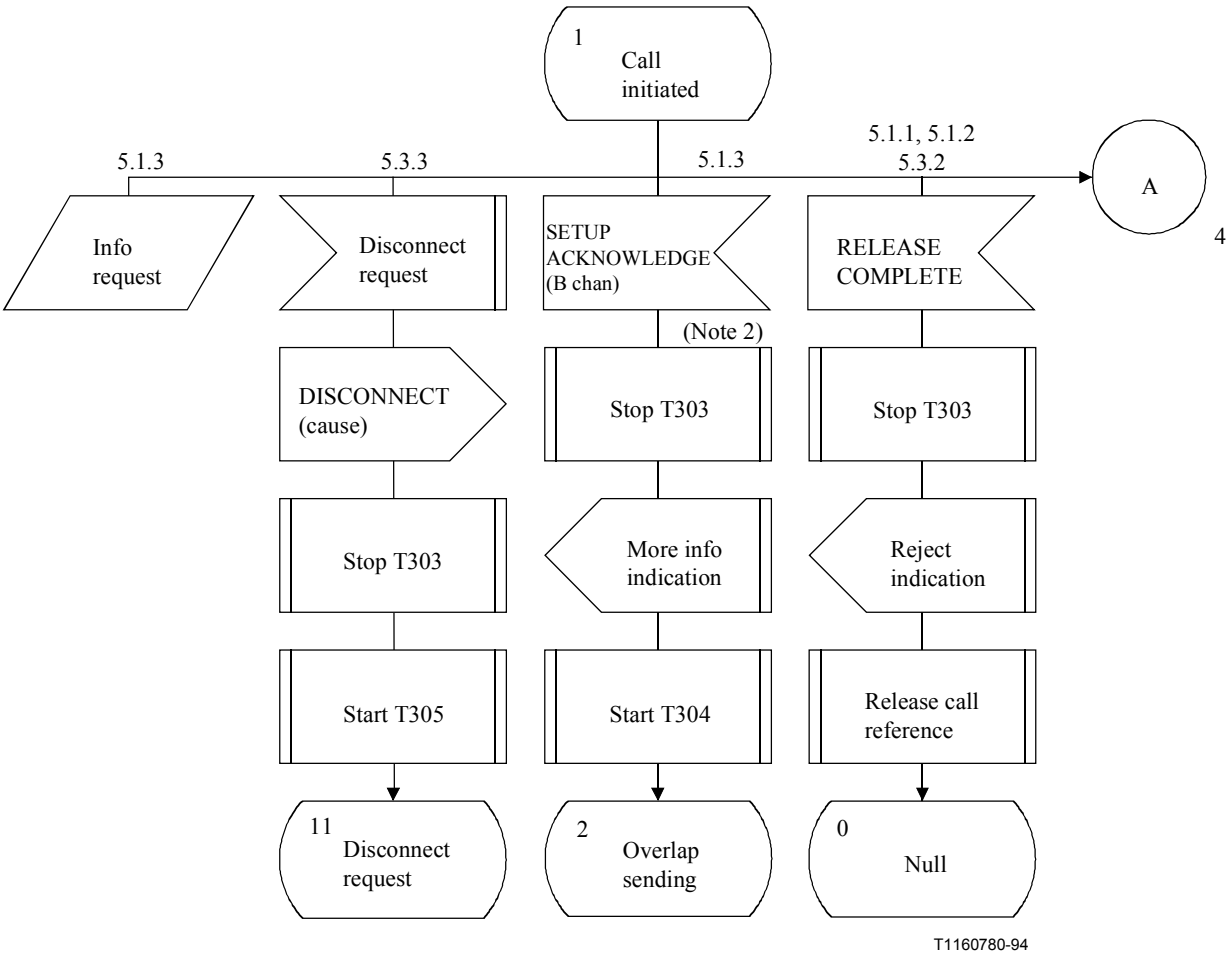


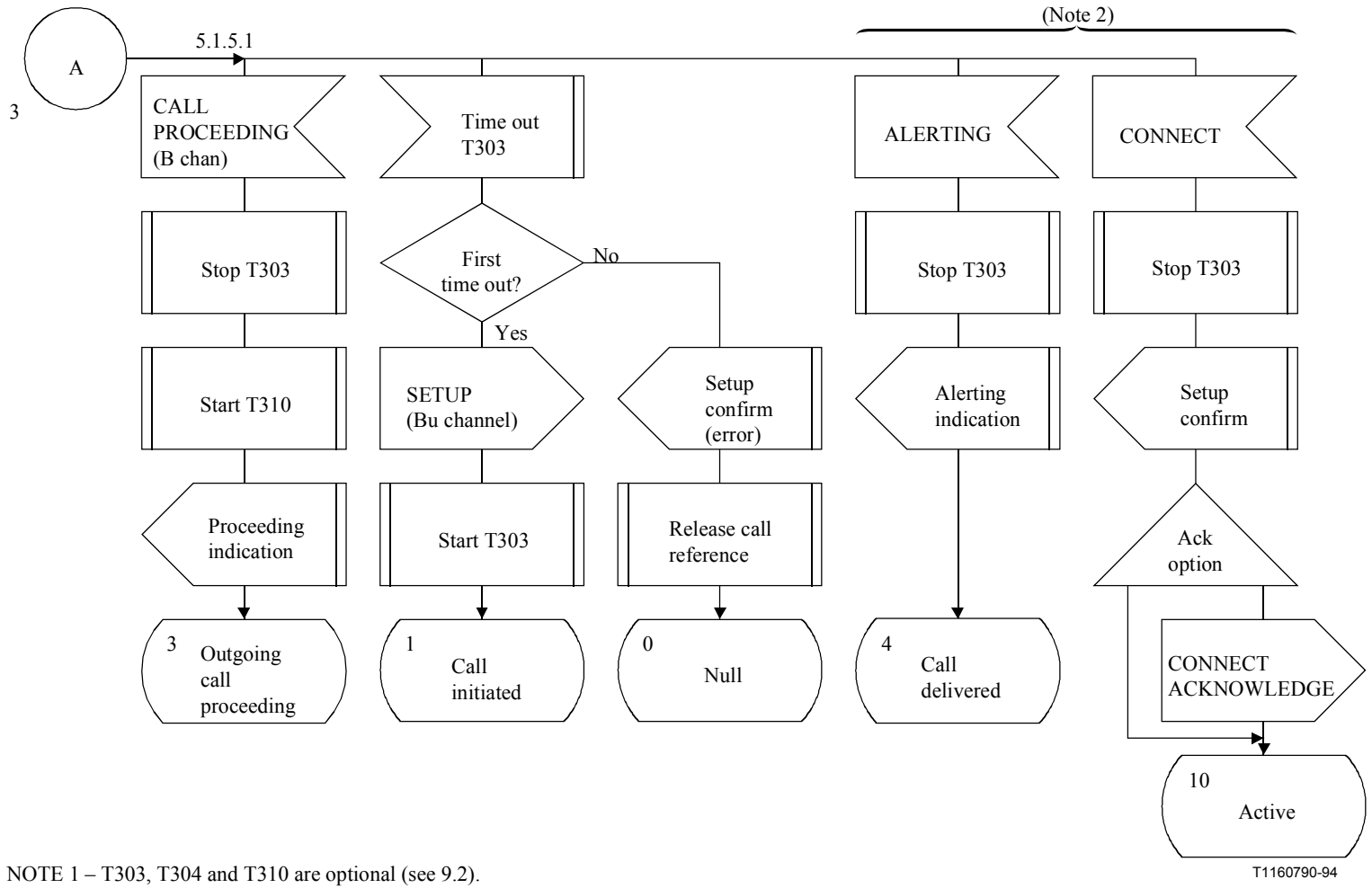
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 2 of 25)***



NOTE 1 – T303, T304 and T310 are optional (see 9.2).

NOTE 2 – "B chan" is a B-channel negotiated by the network and the user.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 3 of 25)***



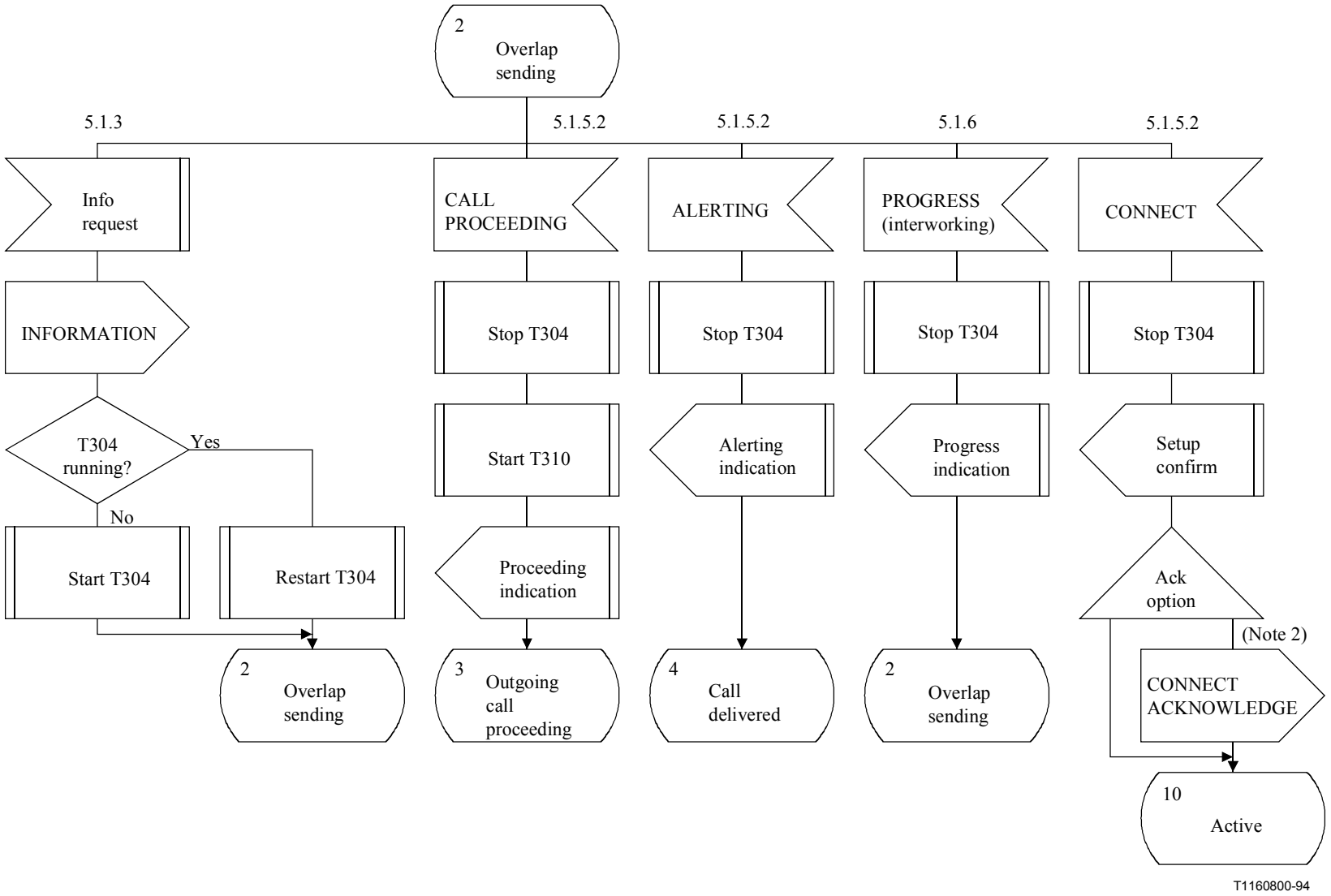
NOTE 1 – T303, T304 and T310 are optional (see 9.2).

NOTE 2 – Only applicable for the procedure defined in Annex D.

NOTE 3 – "B chan" is a B-channel negotiated by the network and the user.

NOTE 4 – T.310 is not started if Progress Indicator 1 or 2 has been delivered in the CALL PROCEEDING MESSAGE.

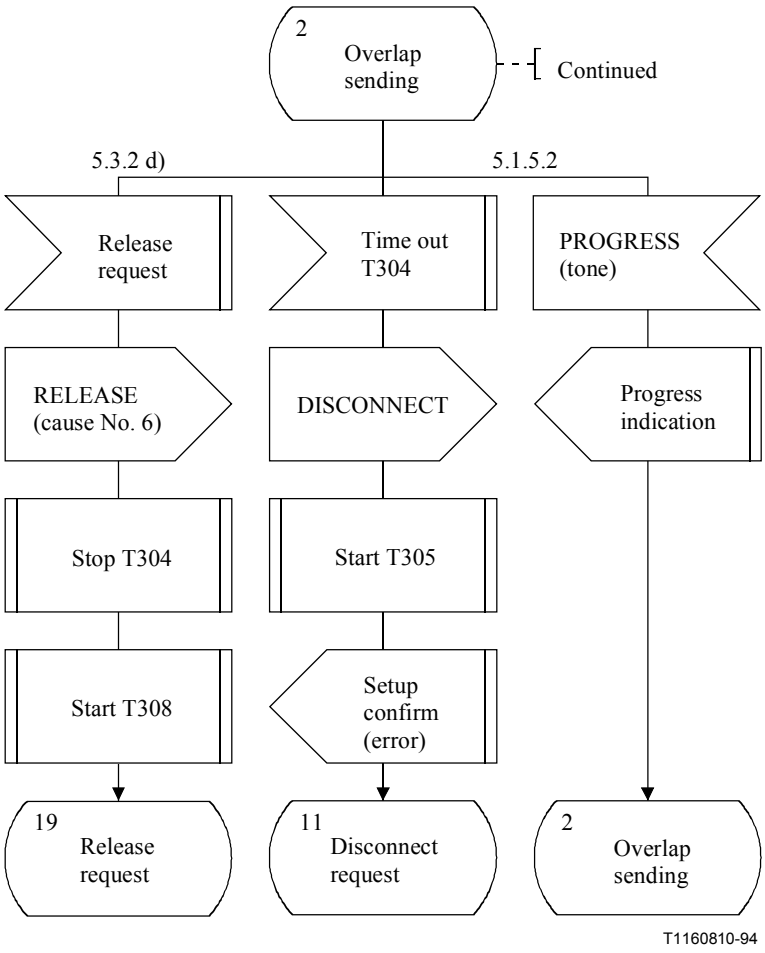
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 4 of 25)***



NOTE 1 – T304 and T310 are optional (see 9.2).

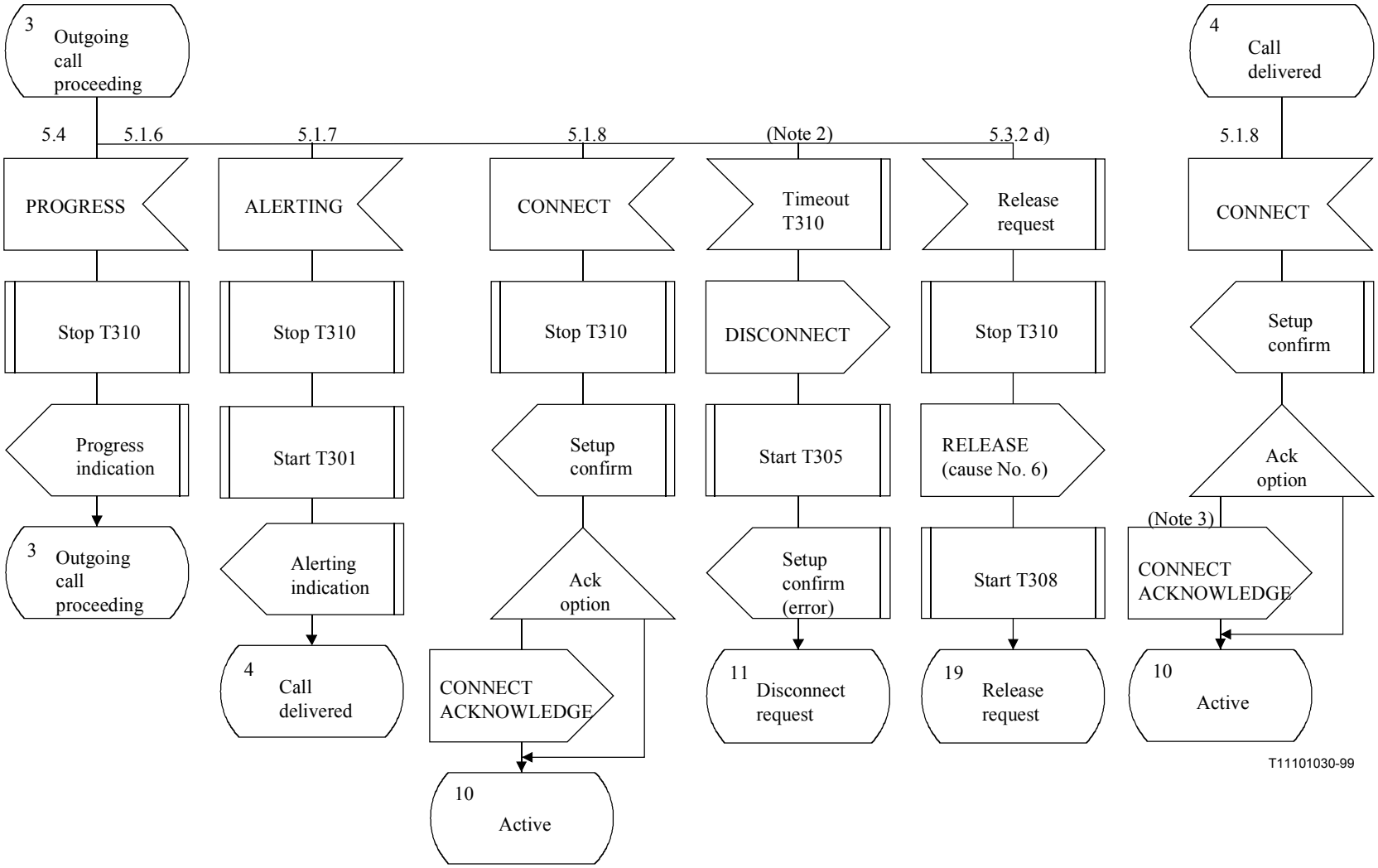
NOTE 2 – This option is used when the procedure in Annex D is implemented.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 5 of 25)***



NOTE – T304 is optional (see 9.2).

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 6 of 25)***

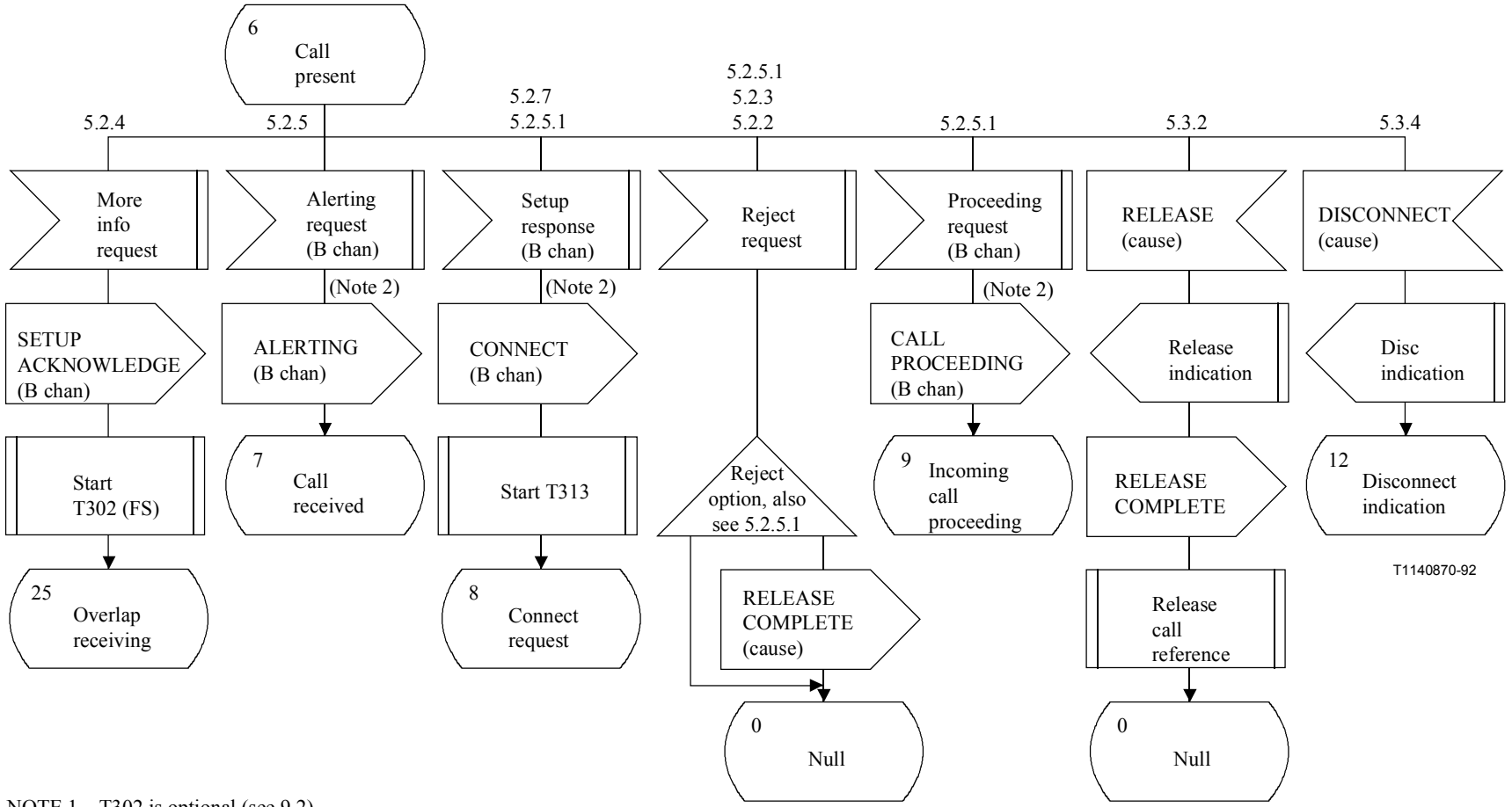


NOTE 1 – T301 and T.310 are optional (see 9.2).

NOTE 2 – Only applicable for the procedure defined in Annex D.

NOTE 3 – This option is used when the procedure in Annex D is implemented.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 7 of 25)***



NOTE 1 – T302 is optional (see 9.2).

NOTE 2 – "B chan" is a B-channel negotiated by the network and the user.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 8 of 25)***

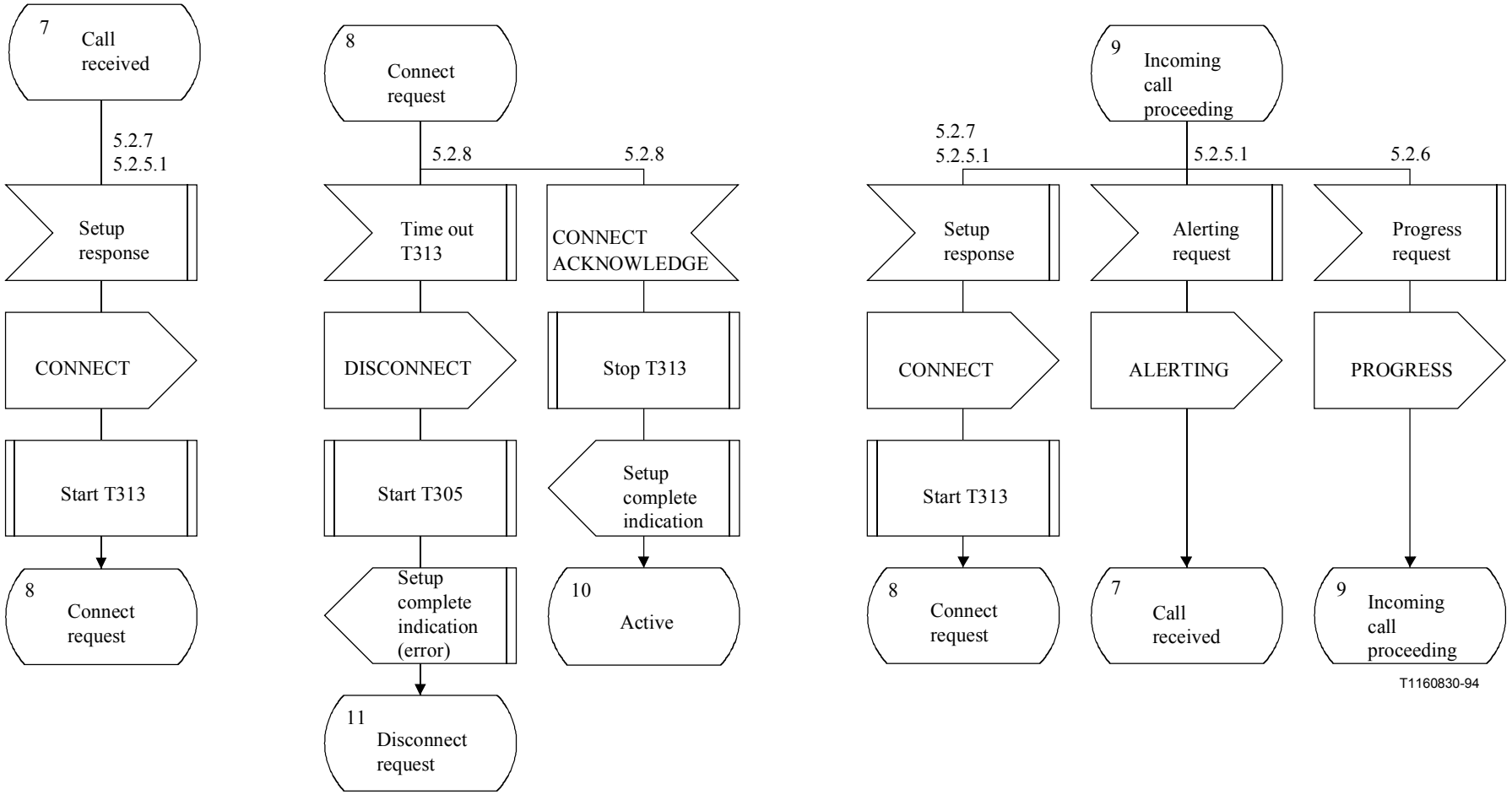


Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 9 of 25)***

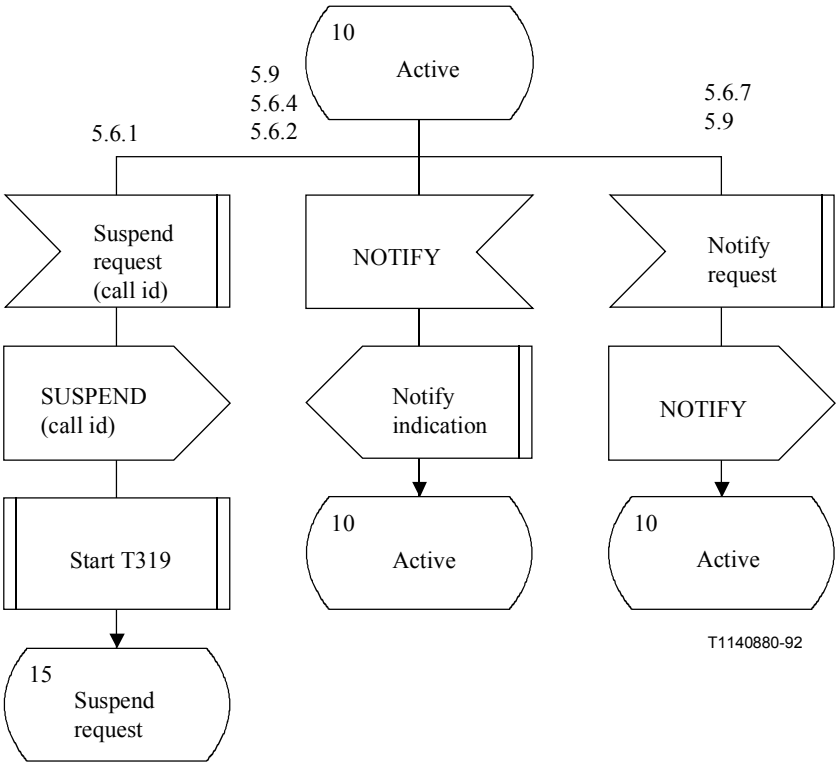
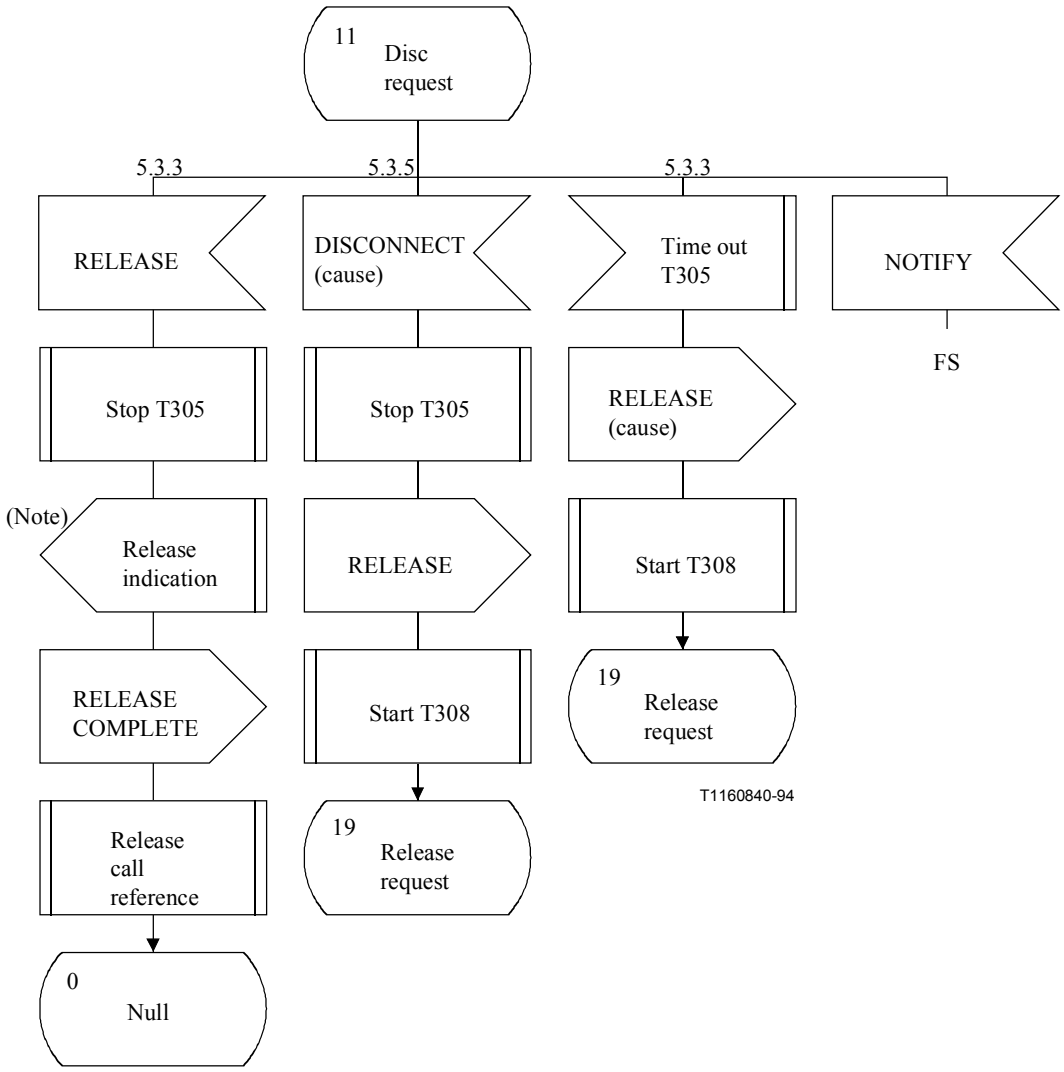
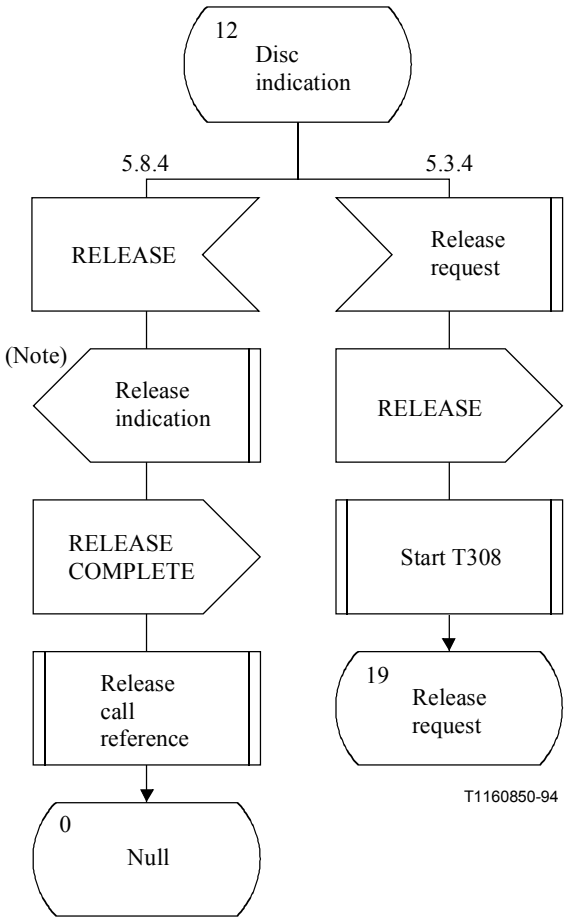


Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 10 of 25)***



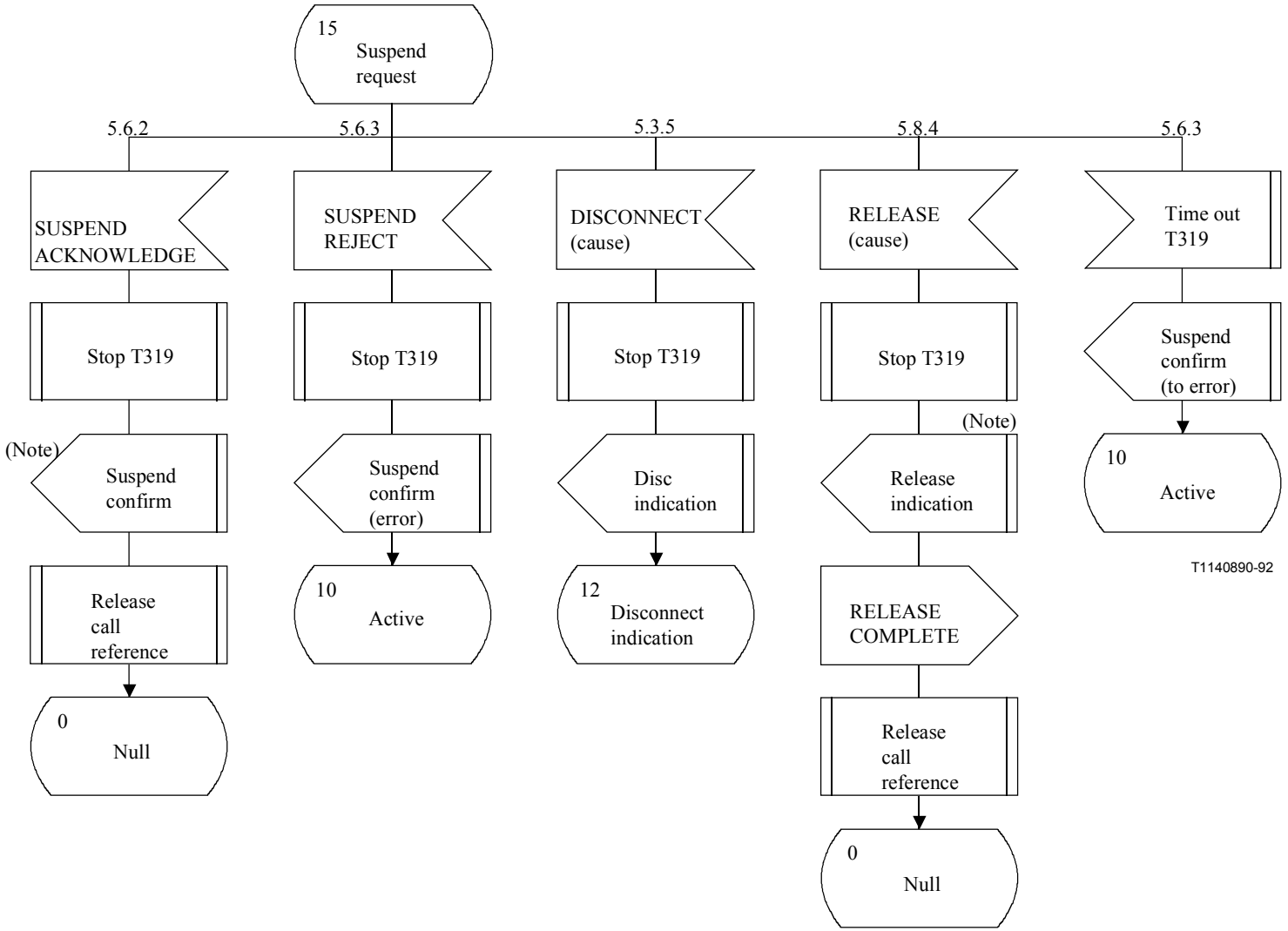
NOTE – After receiving this primitive, call control process should release B-channel.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 11 of 25)***



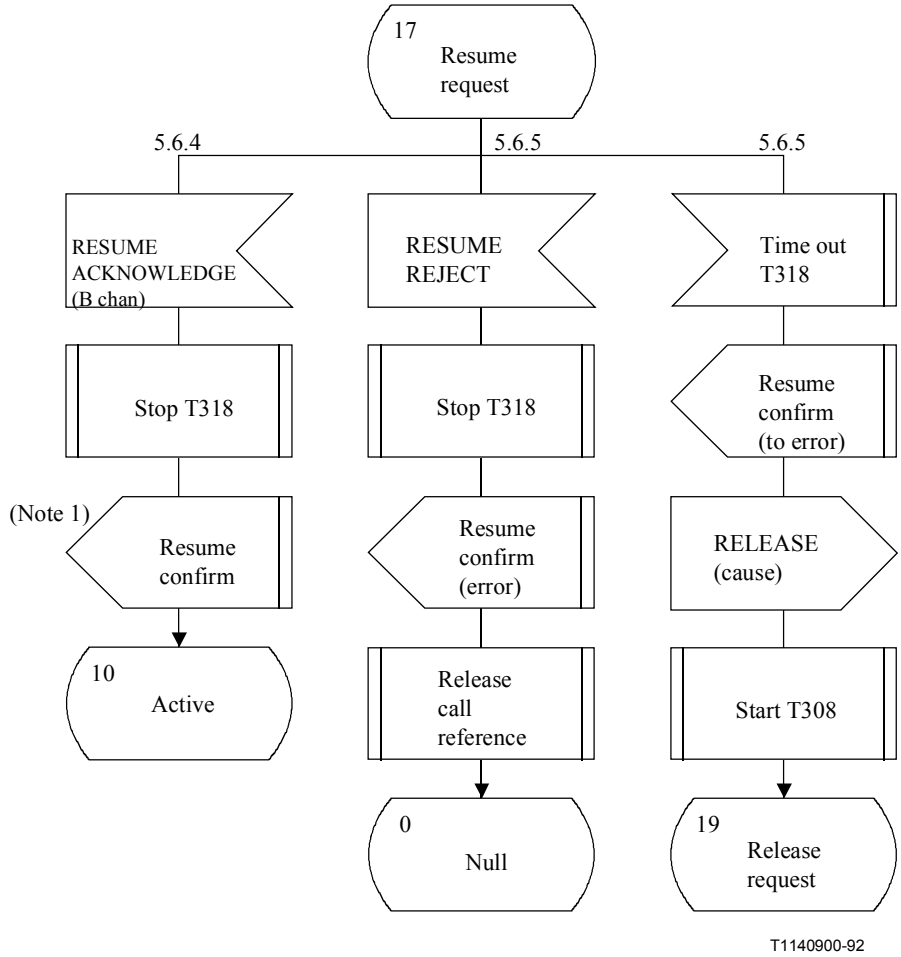
NOTE – After receiving this primitive, call control process should release B-channel.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 12 of 25)***



NOTE – After receiving this primitive, call control process should release B-channel.

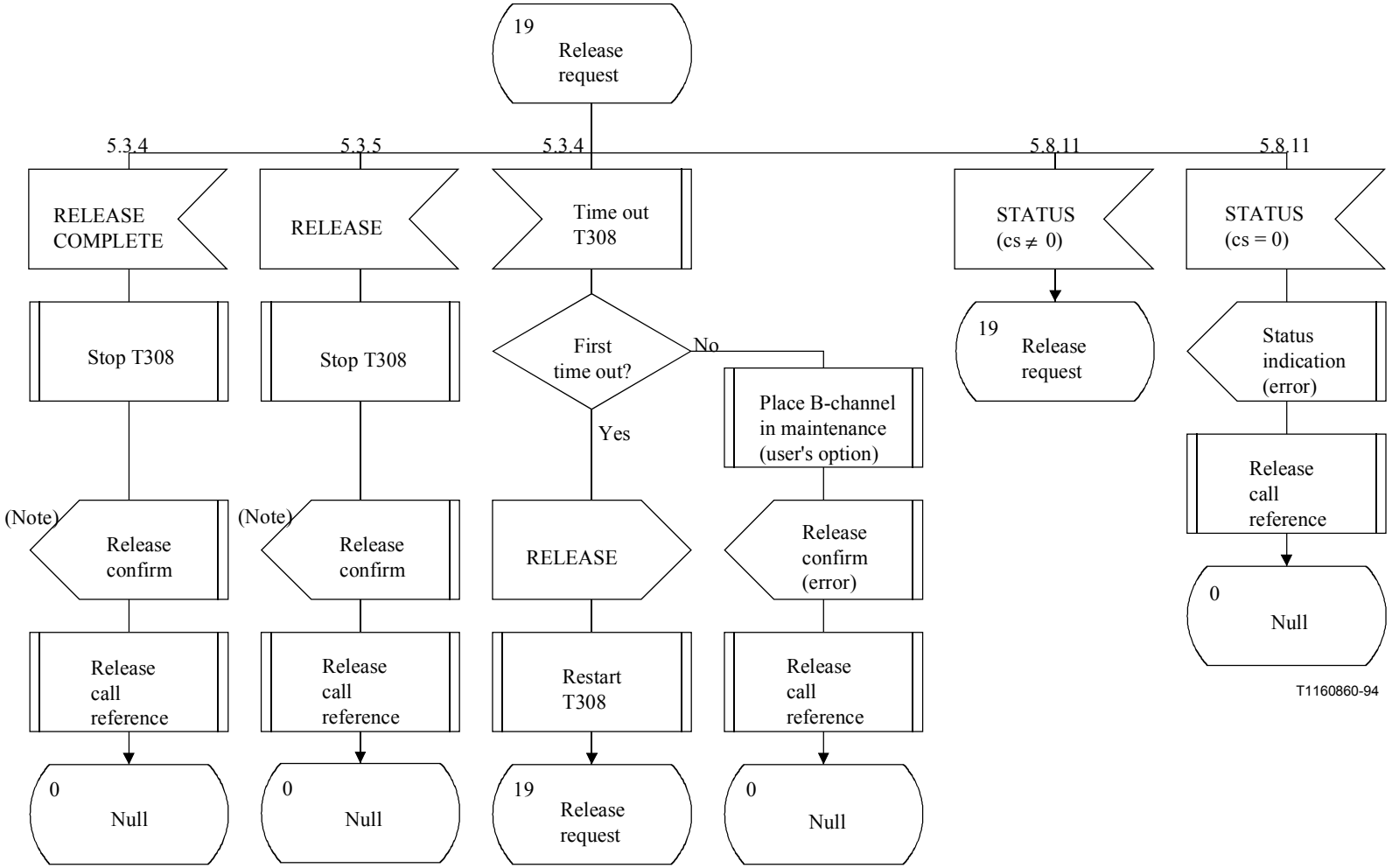
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 13 of 25)***



NOTE 1 – After receiving this primitive, call control process should connect B-channel.

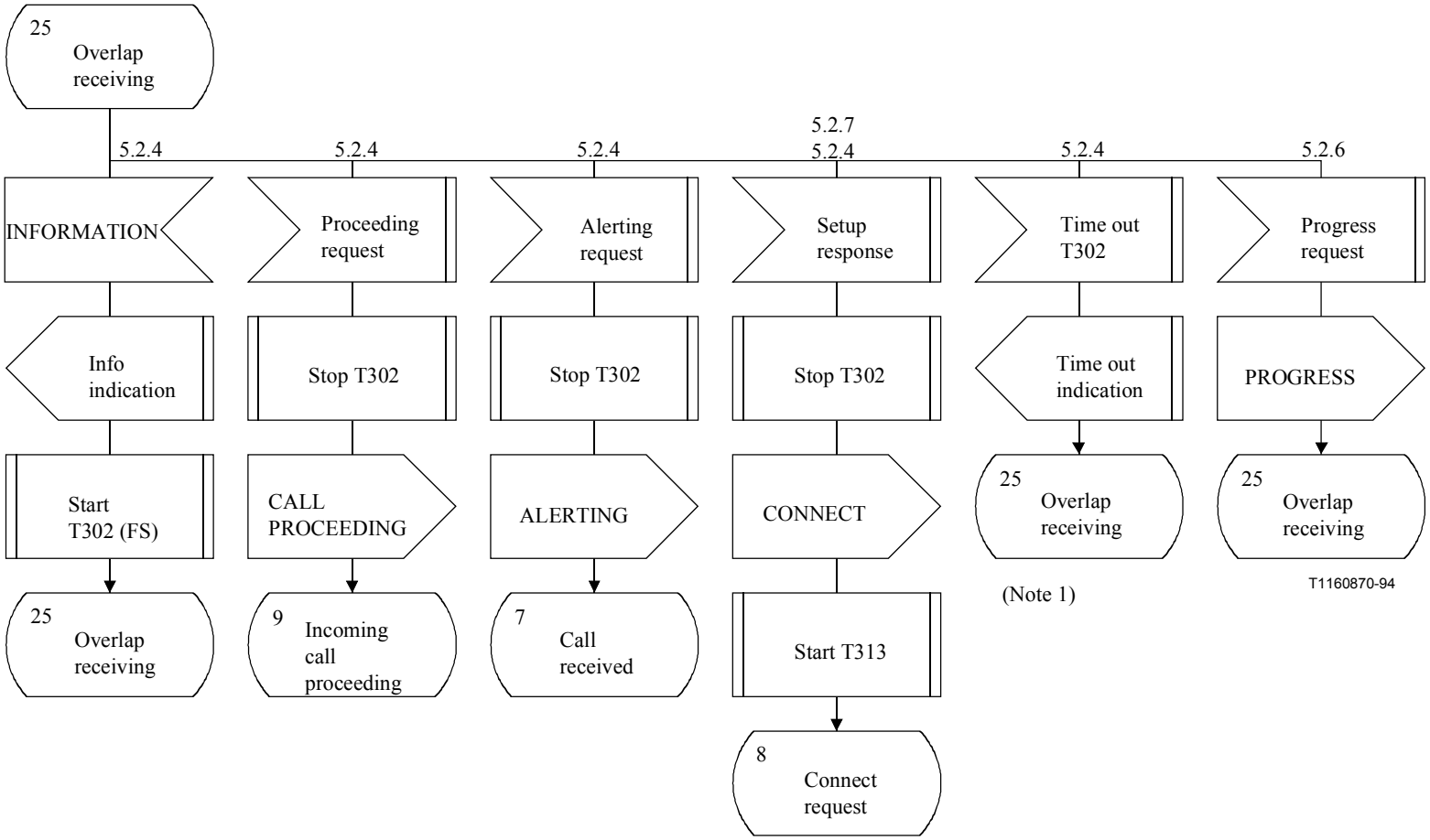
NOTE 2 – Open issue: Handling of disconnect request primitive.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 14 of 25)***



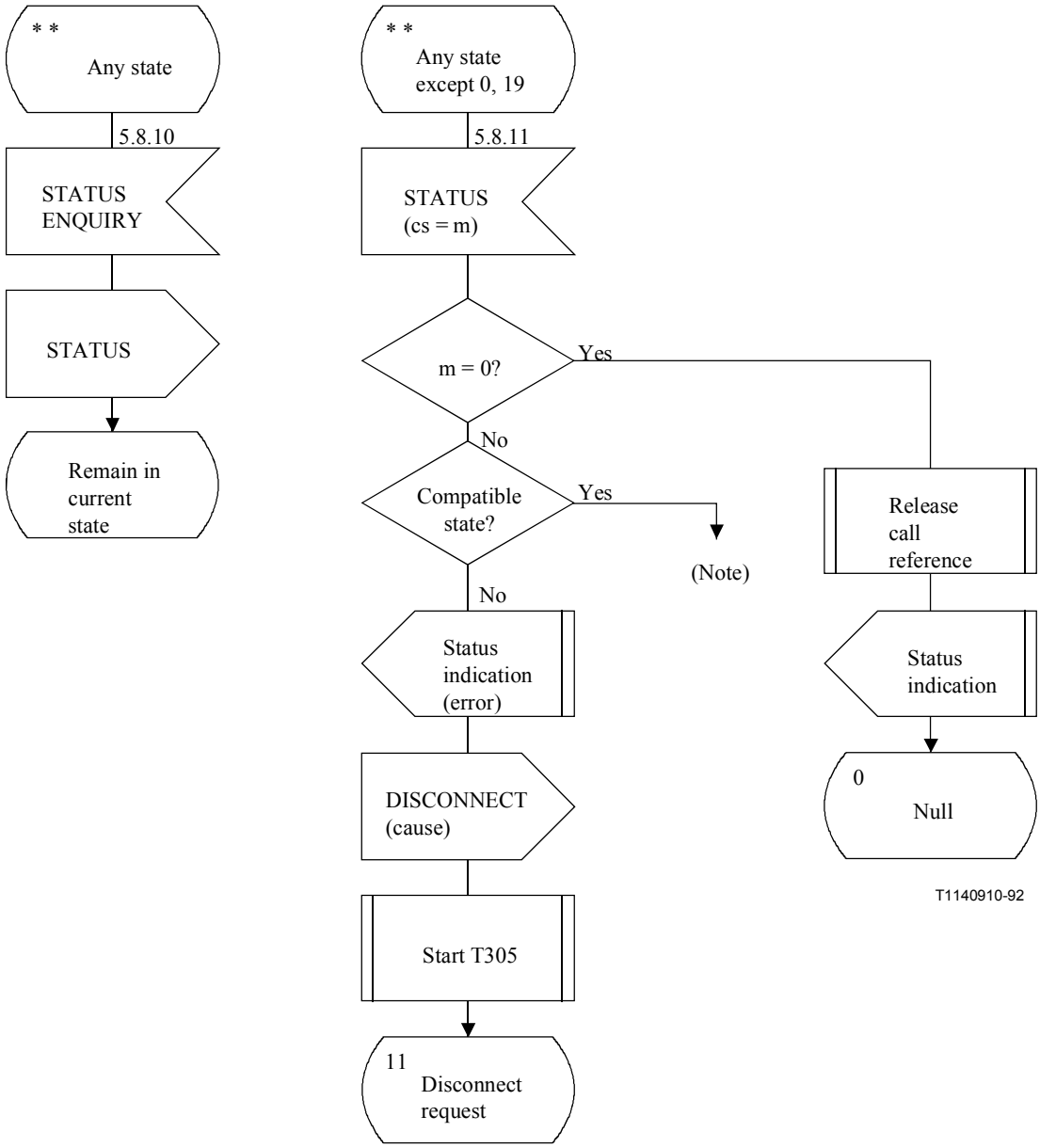
NOTE – After receiving this primitive, call control process should release B-channel.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 15 of 25)***



NOTE 1 – It is assumed that the decision whether complete information has been received or not, at the expiry of T302, will be made by the call control.
NOTE 2 – T302 is optional (see 9.2).

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 16 of 25)***



NOTE – Action on receipt of STATUS indicating a compatible call state is implementation dependent (see 5.8.11).

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 17 of 25)***

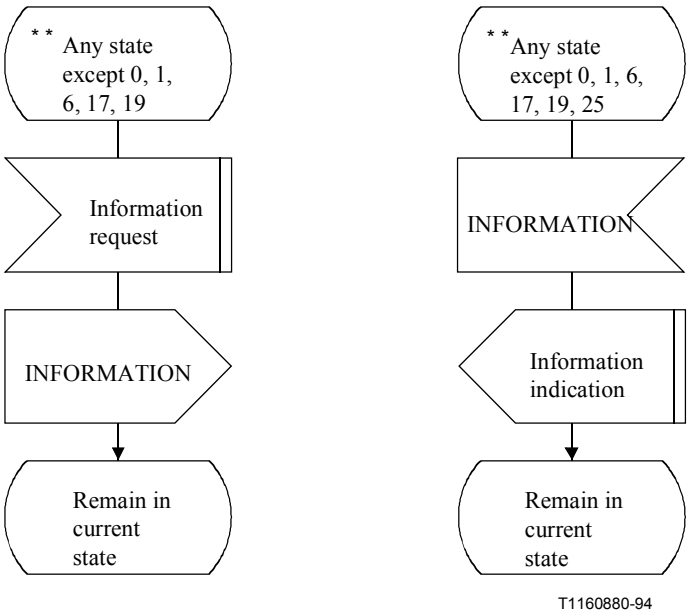
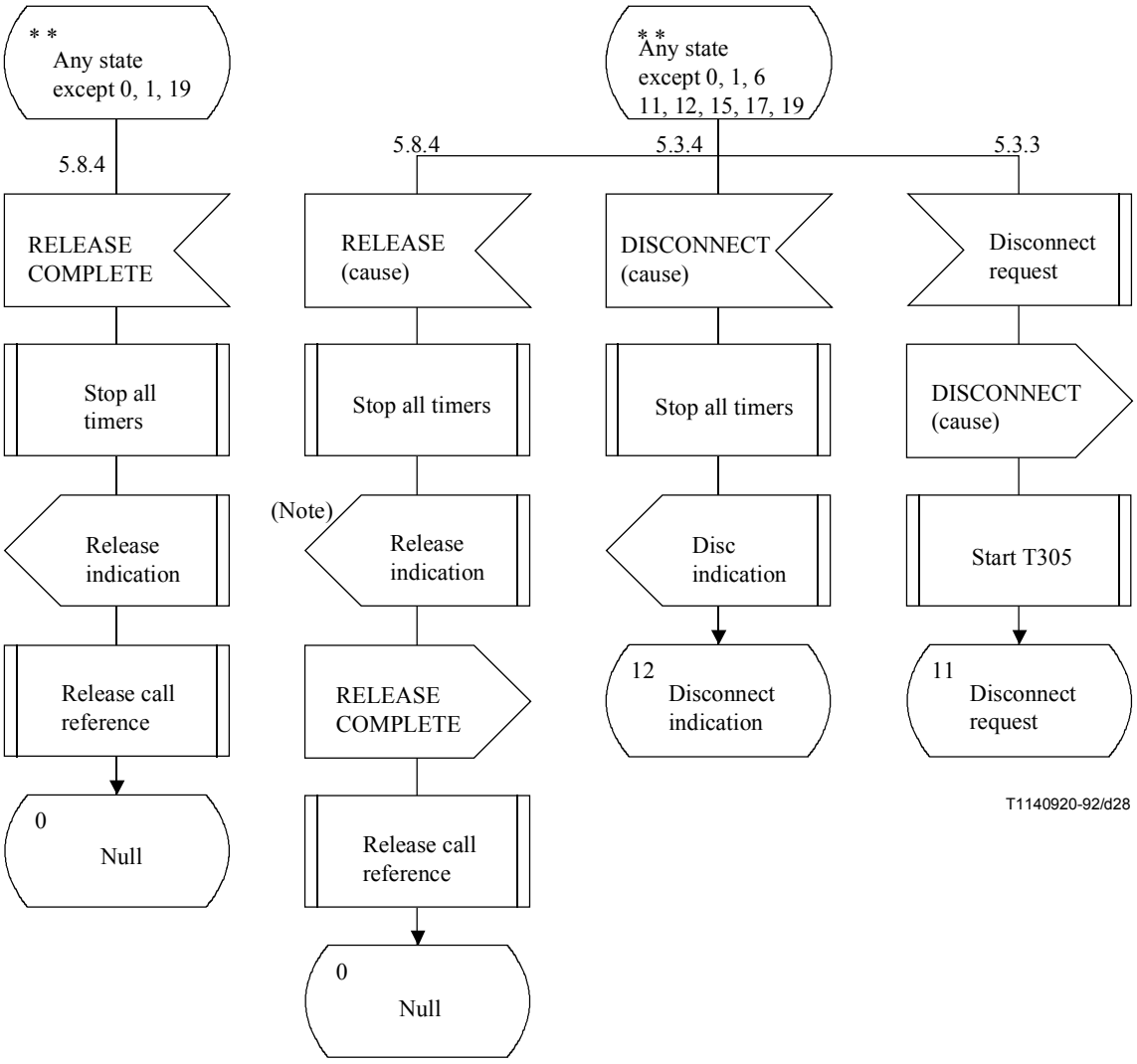
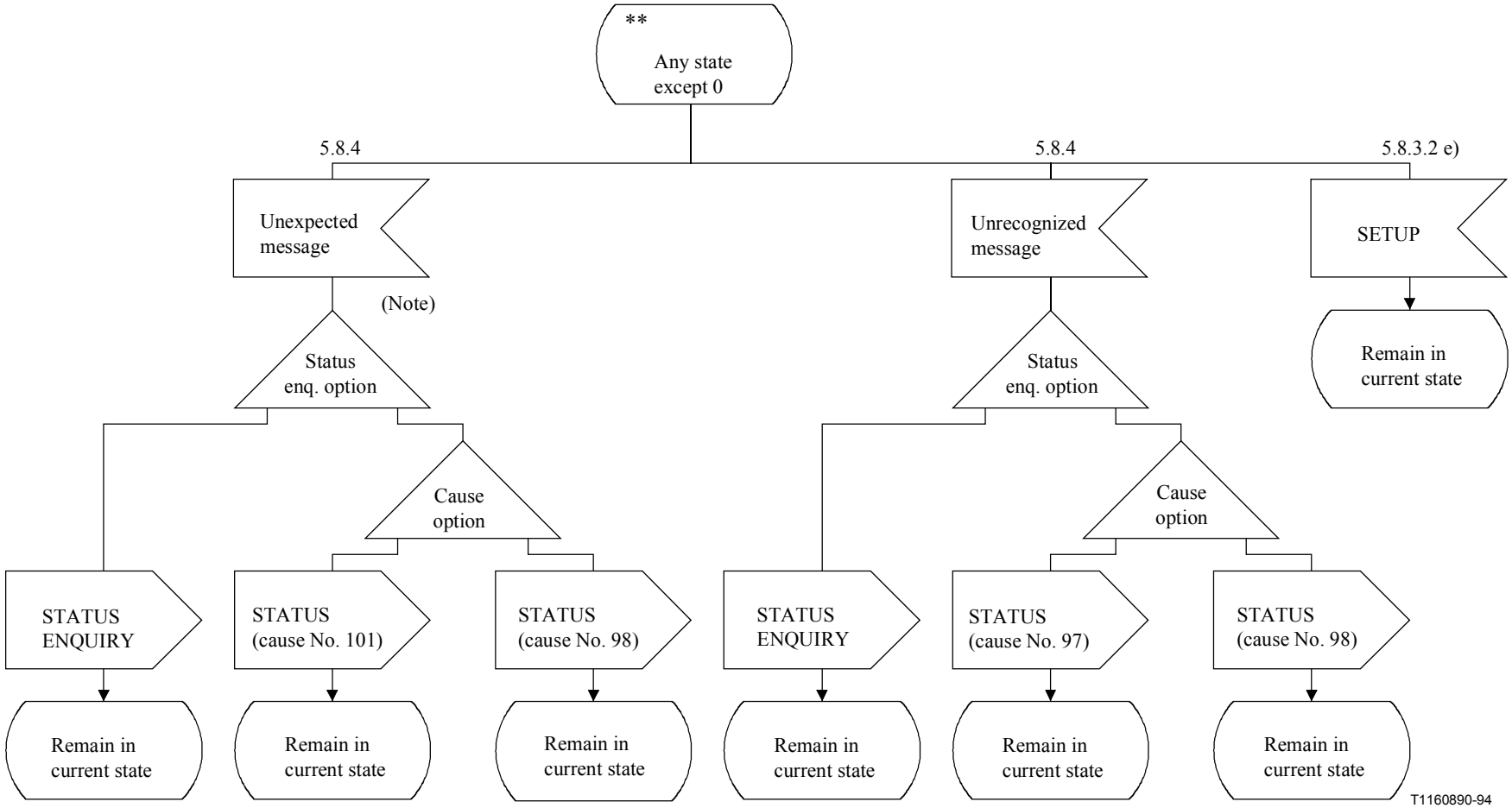


Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 18 of 25)***



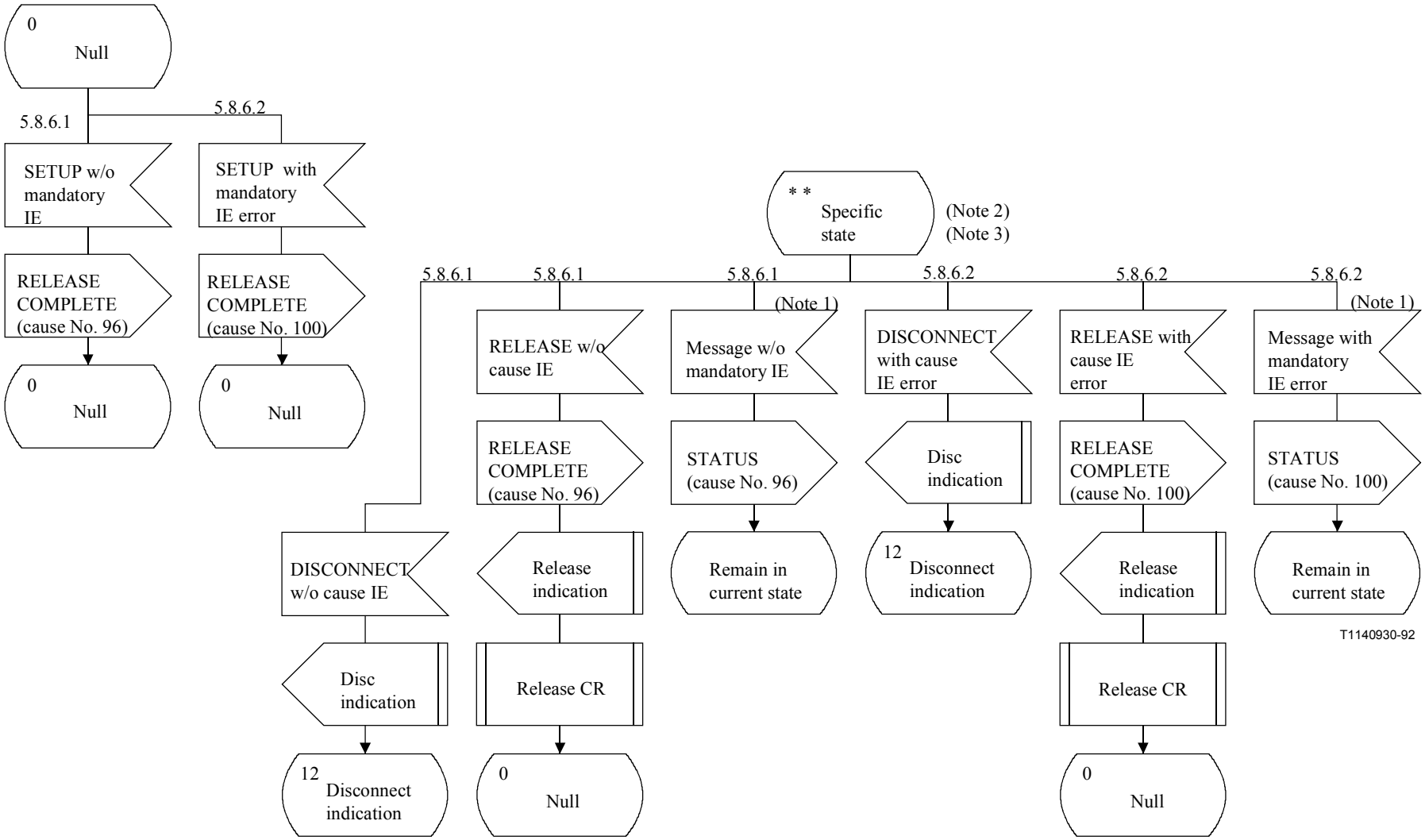
NOTE – After receiving this primitive, call control process should release B-channel.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 19 of 25)***



NOTE – Except RELEASE or RELEASE COMPLETE.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 20 of 25)***



NOTE 1 – Except SETUP, RELEASE, RELEASE COMPLETE and DISCONNECT.

NOTE 2 – These messages are recognized by the user as expected messages in the state. [See Figure A.3 (sheet 15 of 25)].

NOTE 3 – See 5.8.6 procedures for specific states.

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 21 of 25)***

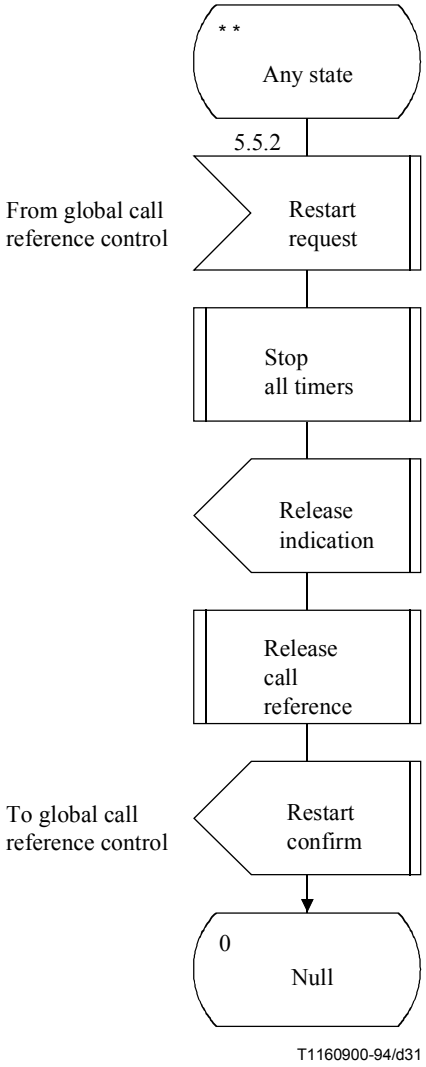
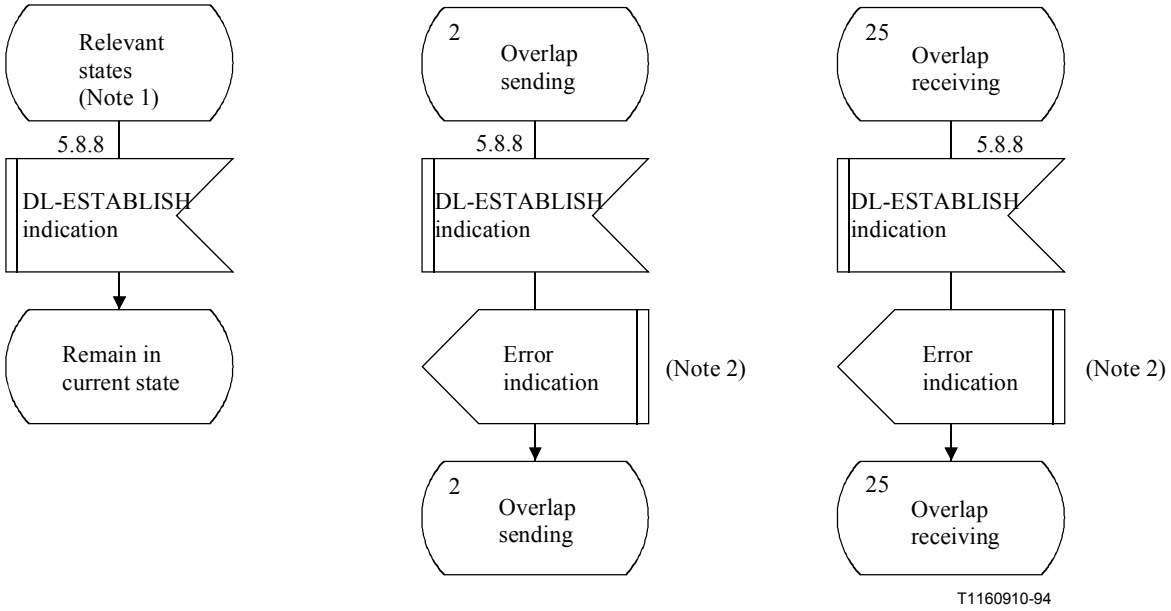


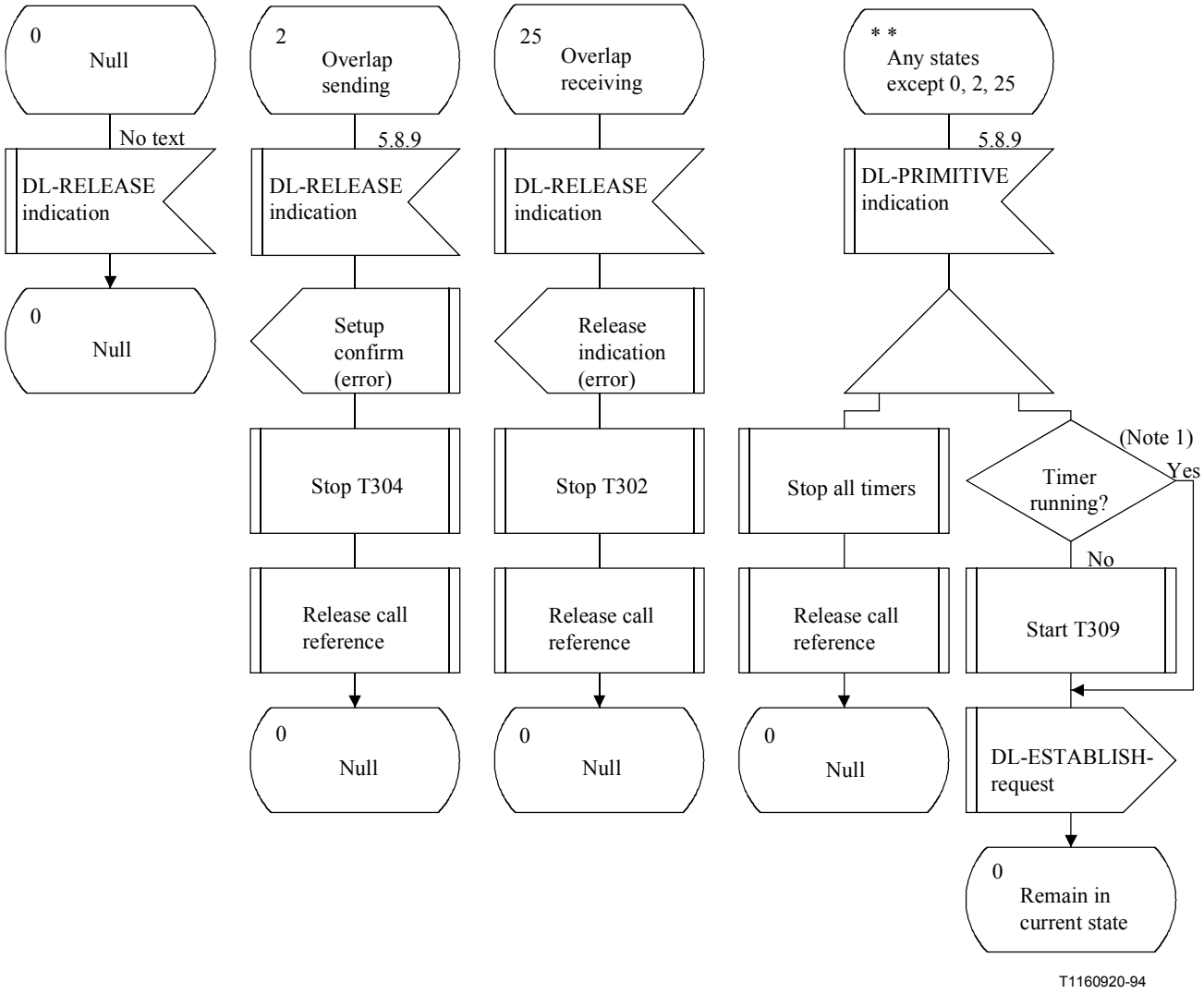
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 22 of 25)***



NOTE 1 – The relevant states are as follows: U1, U3, U4, U6 to U12, U15, U17, U19.

NOTE 2 – At the reception of this primitive, the call control should clear the call by sending disconnect request primitives.

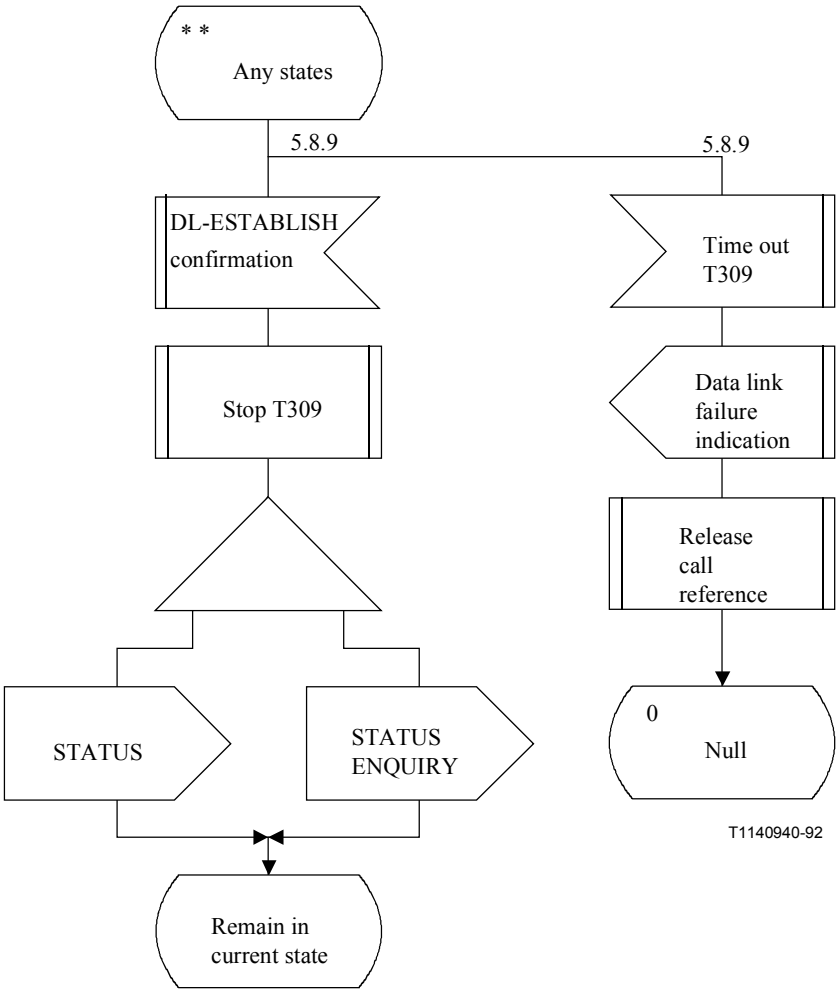
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 23 of 25)***



NOTE 1 – Any timers including T309.

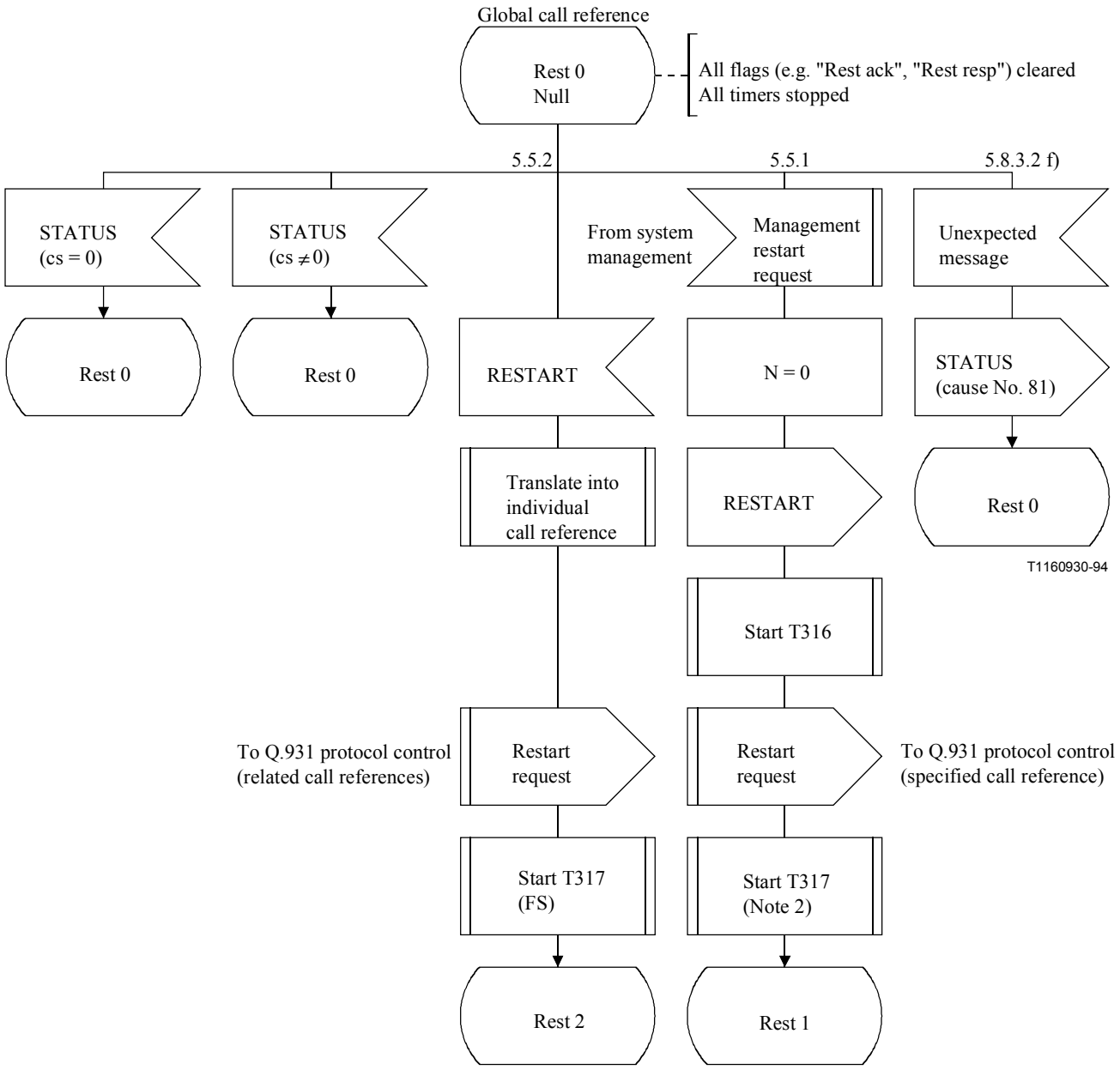
NOTE 2 – T309 is optional (see 9.2).

Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 24 of 25)***



NOTE – T309 is optional (see 9.2).

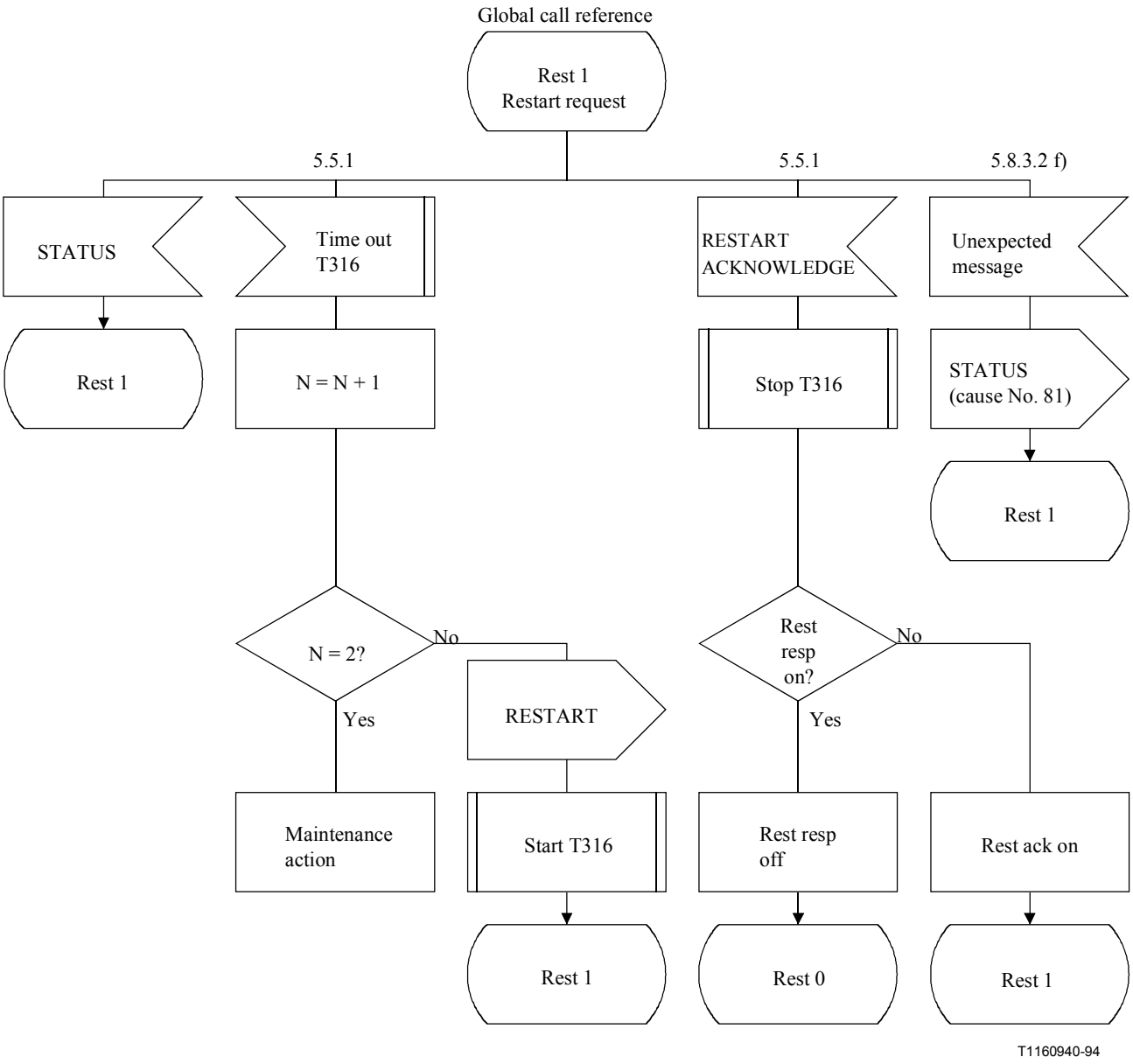
Figure A.3/Q.931 – Detailed protocol control (user side) ***(sheet 25 of 25)***



NOTE 1 – T316 and T317 are optional (see 9.2).

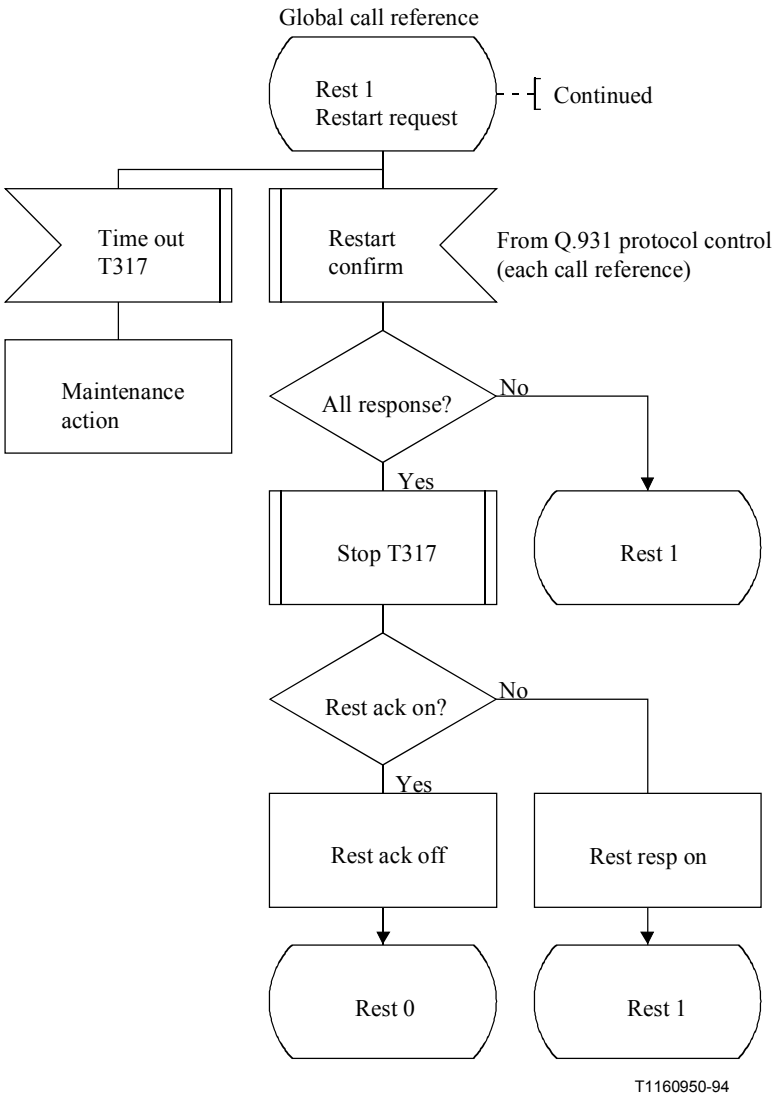
NOTE 2 – The value of T317 is implementation dependent.

Figure A.4/Q.931 – Detailed protocol control for the global call reference (user side) ***(sheet 1 of 4)***



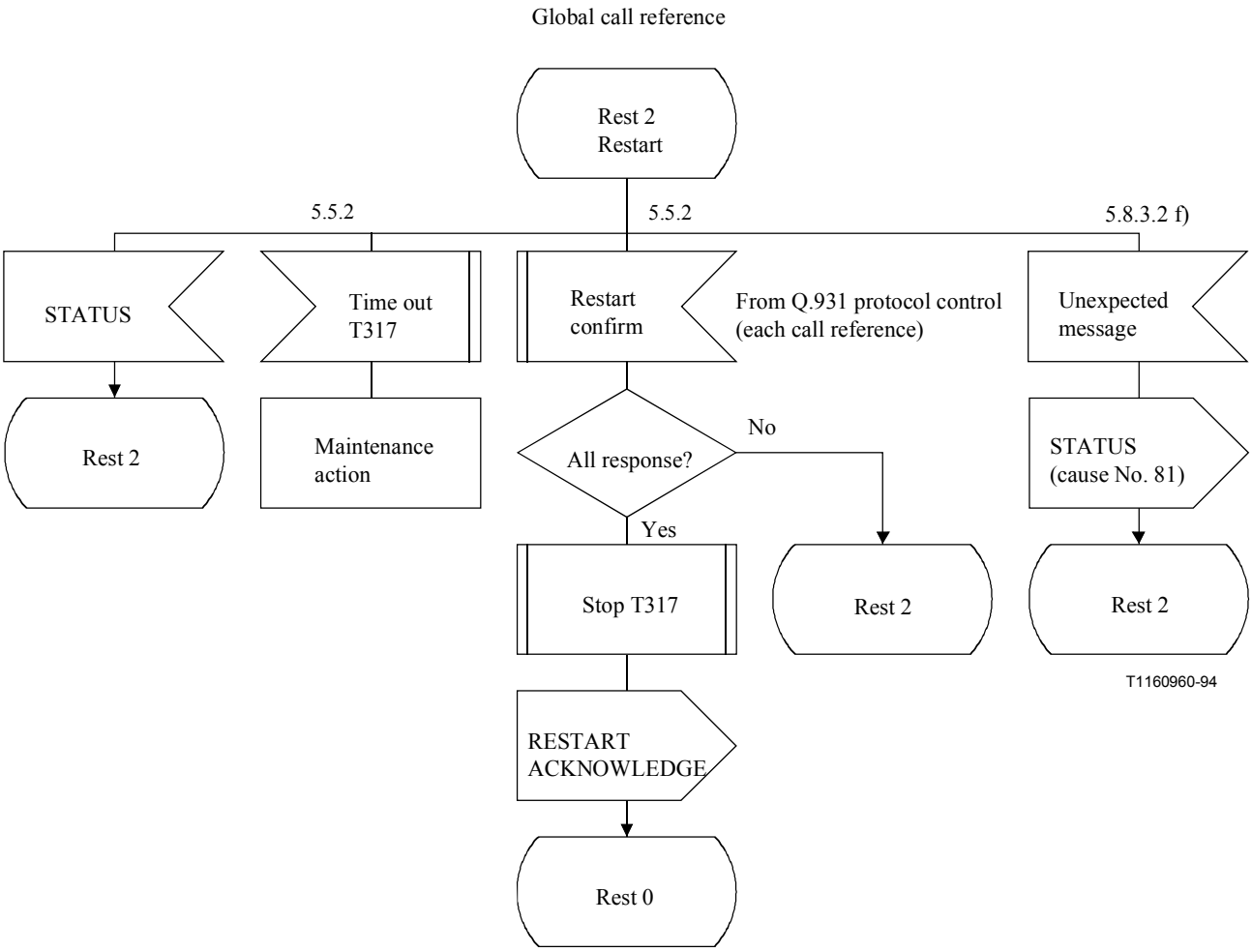
NOTE – T316 is optional (see 9.2).

Figure A.4/Q.931 – Detailed protocol control for the global call reference (user side) ***(sheet 2 of 4)***



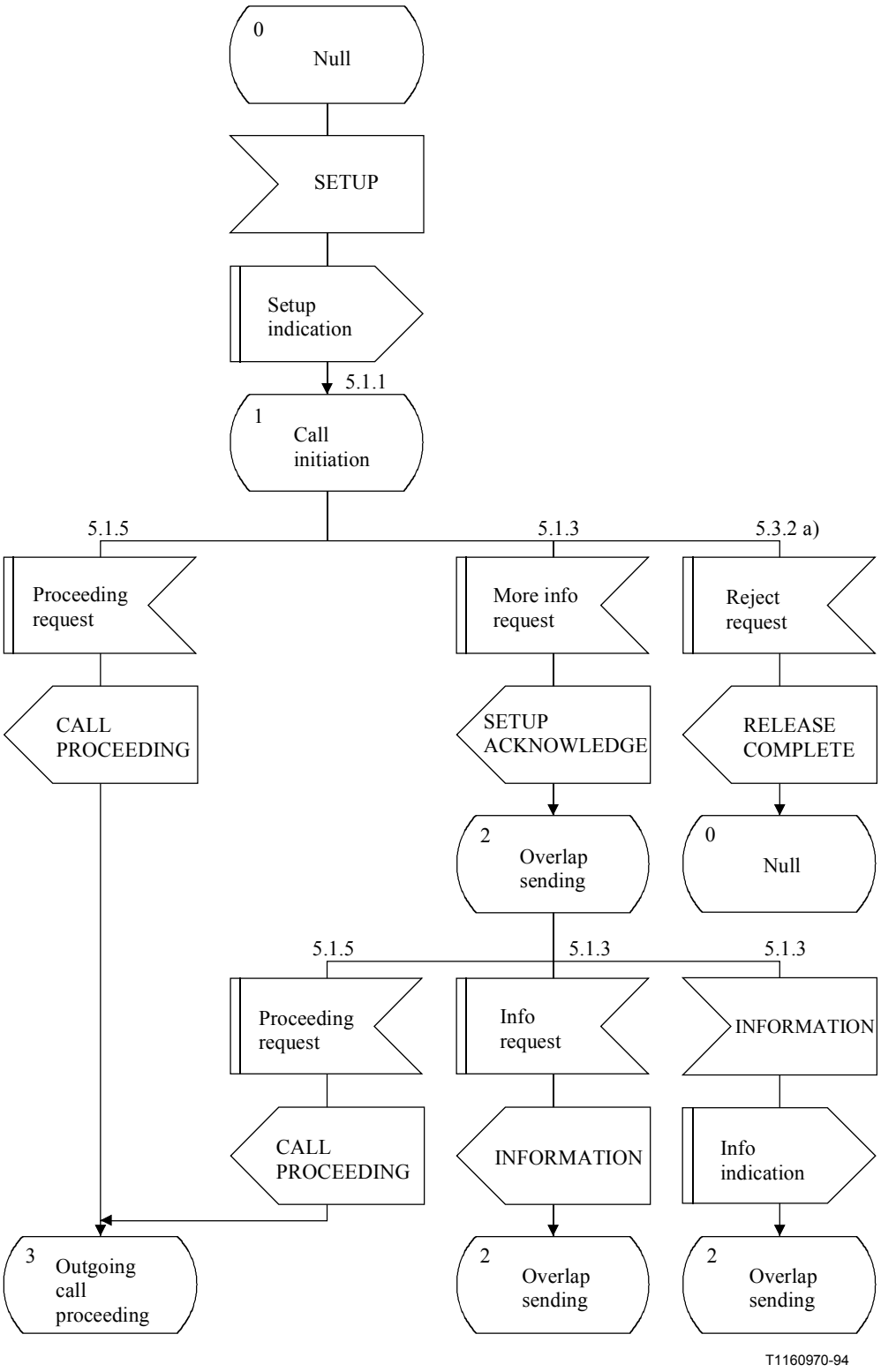
NOTE – T317 is optional (see 9.2).

Figure A.4/Q.931 – Detailed protocol control for the global call reference (user side) ***(sheet 3 of 4)***



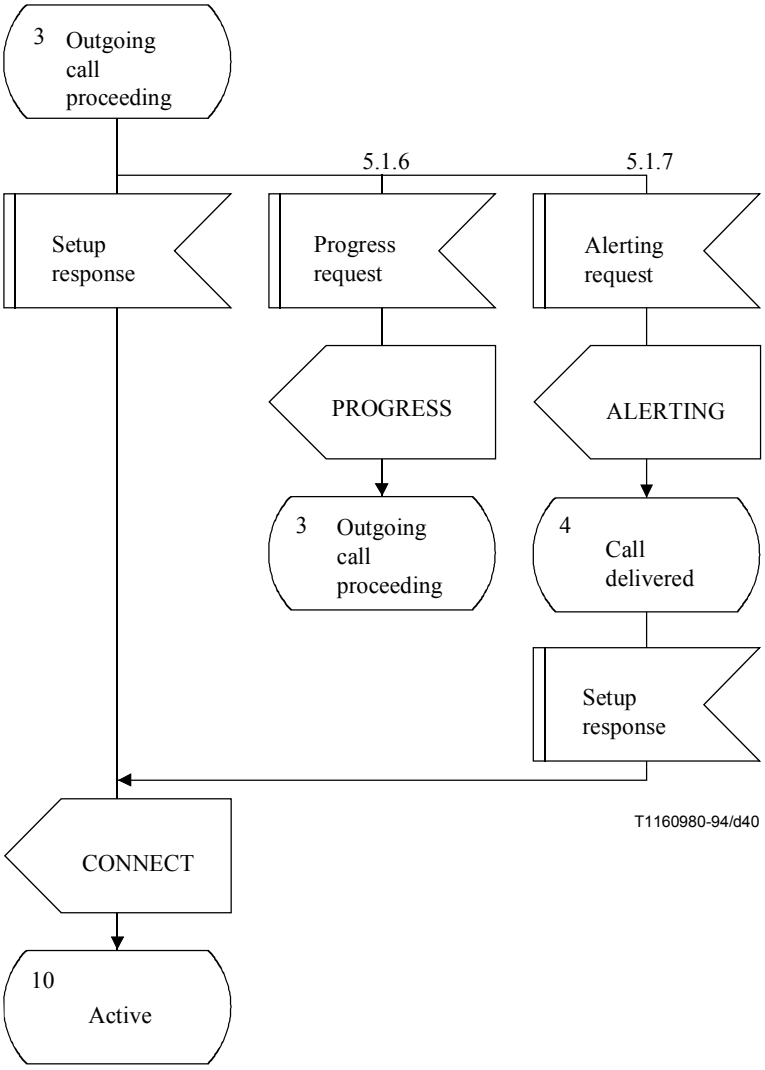
NOTE – T317 is optional (see 9.2).

Figure A.4/Q.931 – Detailed protocol control for the global call reference (user side) ***(sheet 4 of 4)***



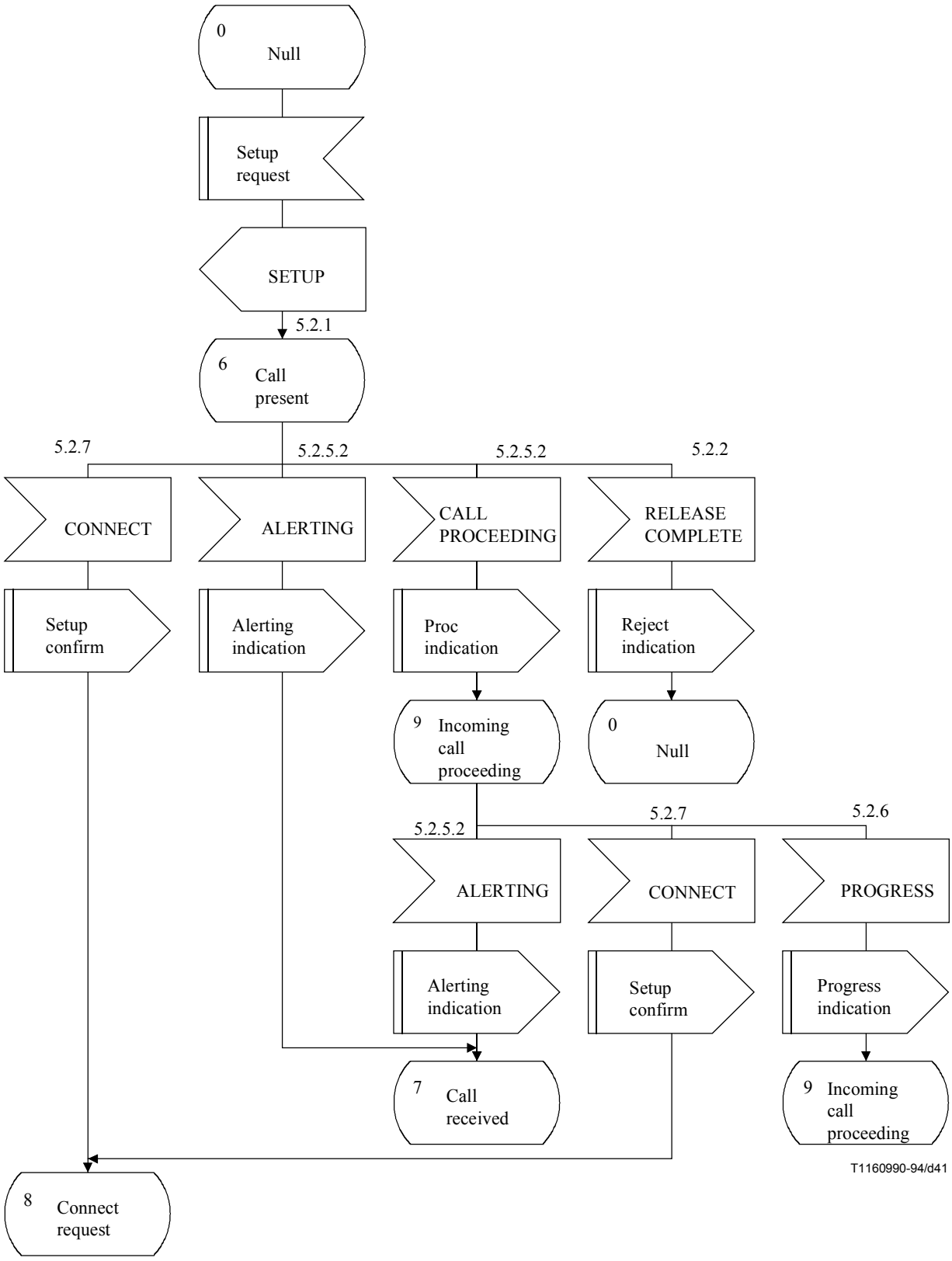
a) Outgoing set-up procedure (1 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 1 of 8)*



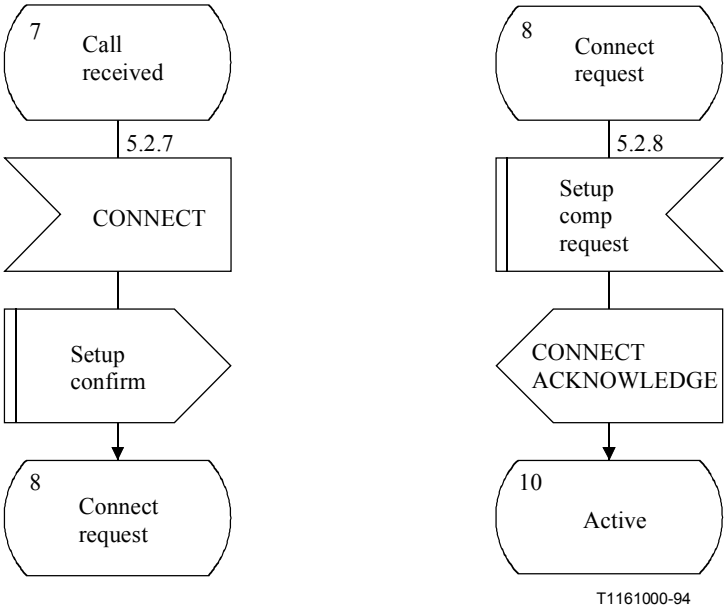
a) Outgoing set-up procedure (2 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 2 of 8)*



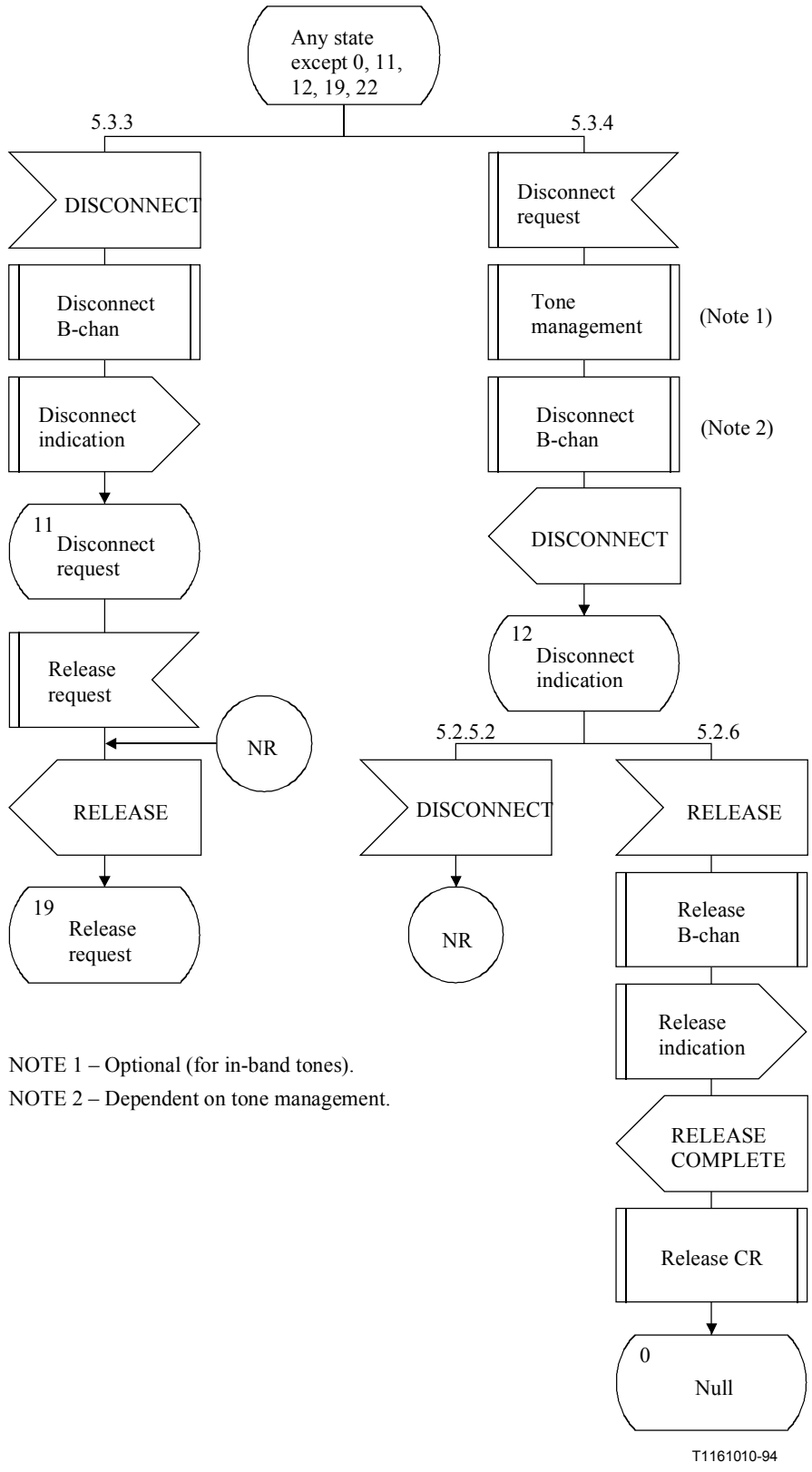
b) Incoming set-up procedure (1 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 3 of 8)*



b) Incoming set-up procedure (2 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 4 of 8)*

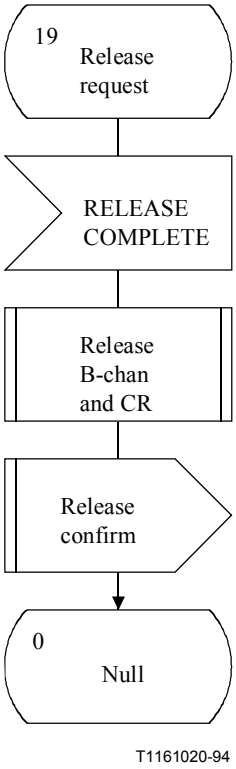


NOTE 1 – Optional (for in-band tones).

NOTE 2 – Dependent on tone management.

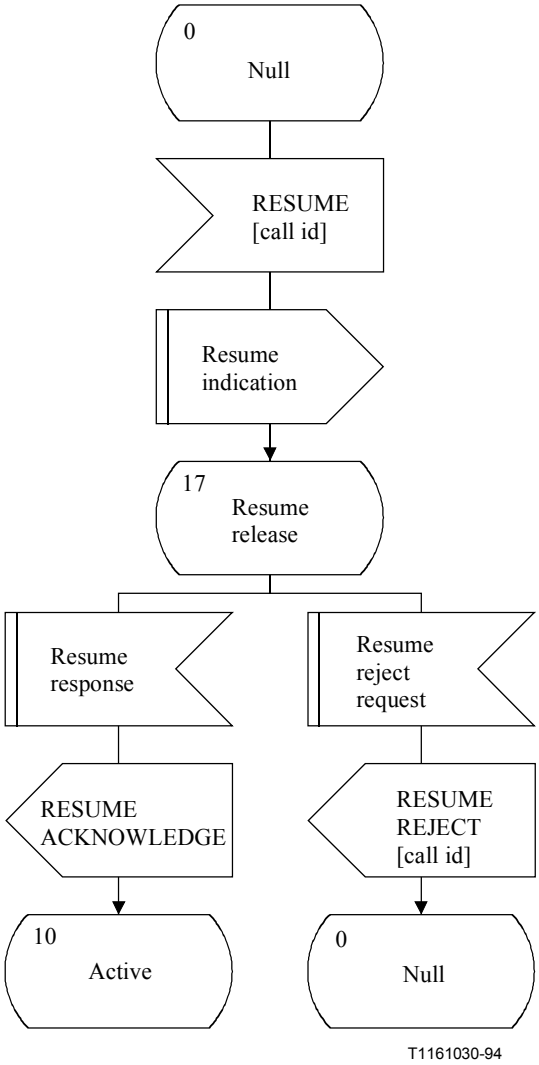
c) Clearing procedure (1 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 5 of 8)*



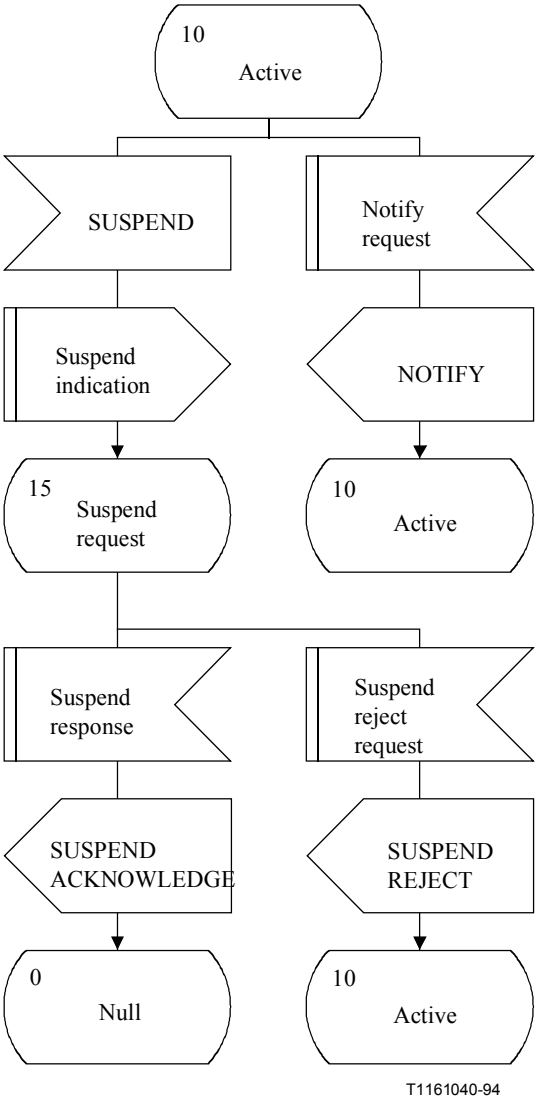
c) Clearing procedure (2 of 2)

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 6 of 8)*



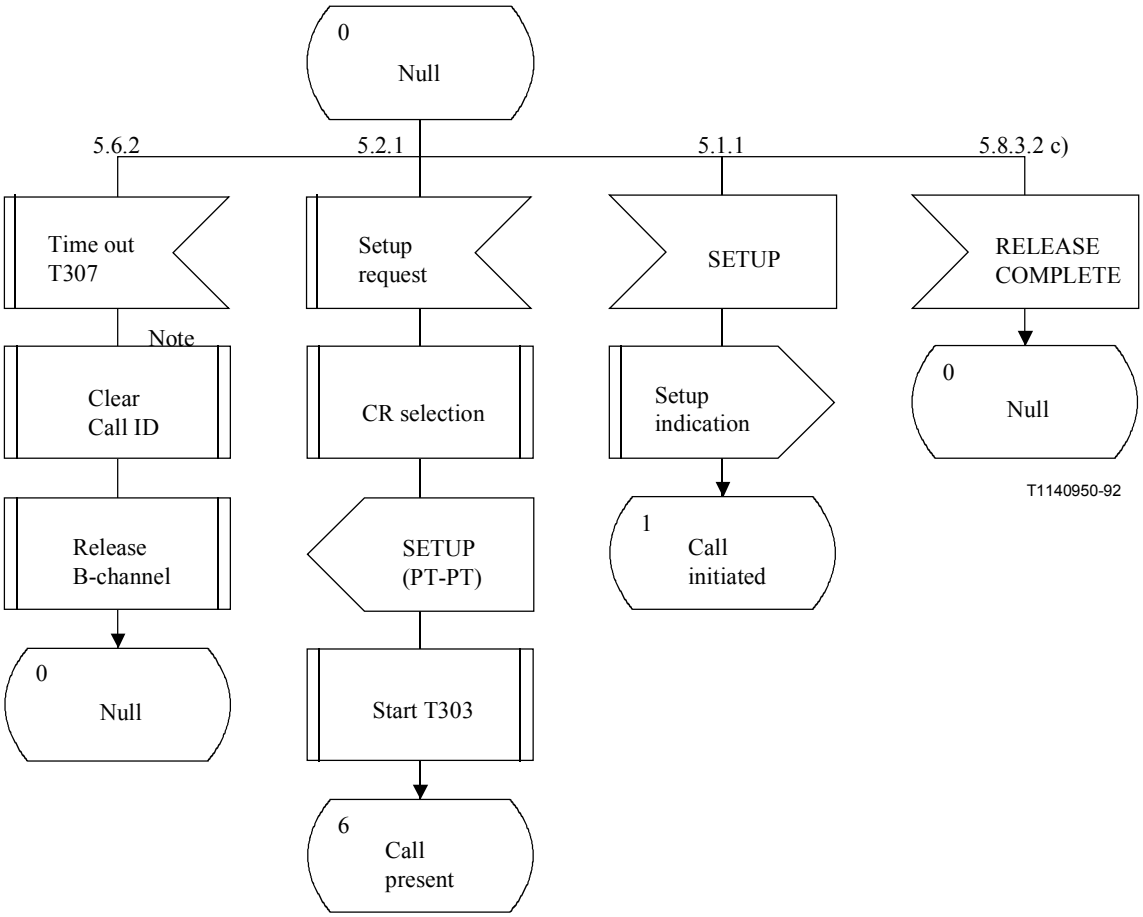
d) Resume procedure

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 7 of 8)*



e) Suspend procedure

Figure A.5/Q.931 – Overview protocol control (network side) point-point *(sheet 8 of 8)*



NOTE – No call reference is associated with T307.

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 1 of 28)*

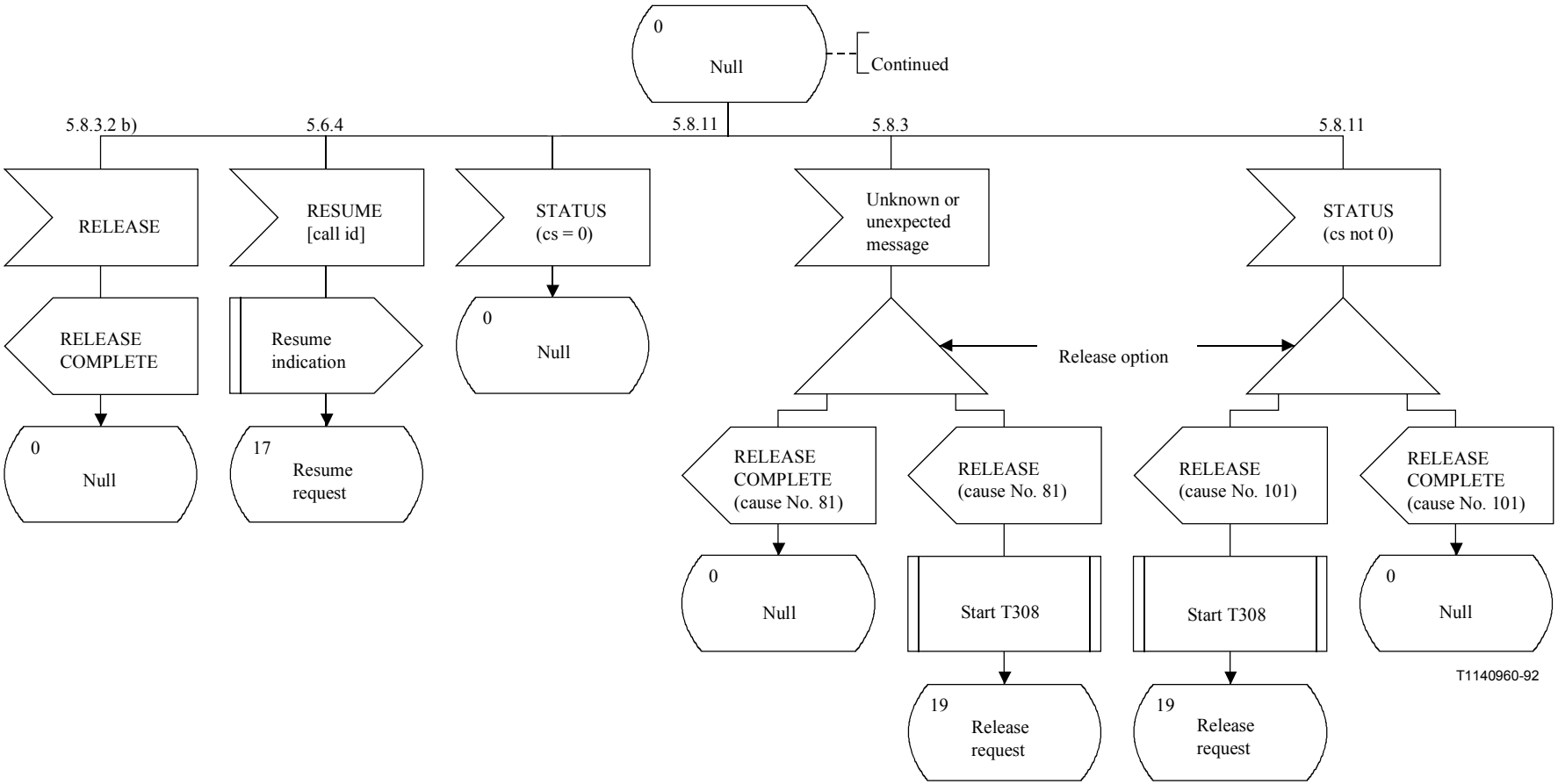


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 2 of 28)*

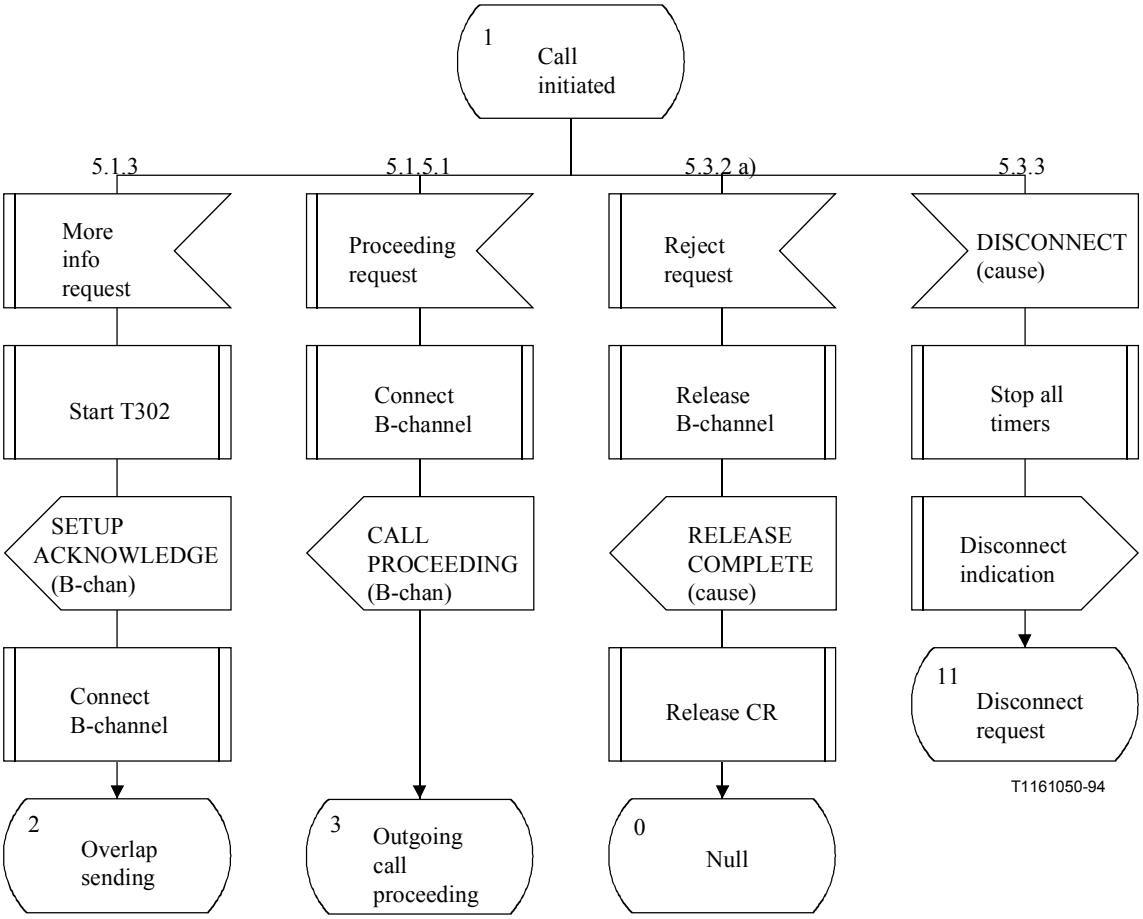
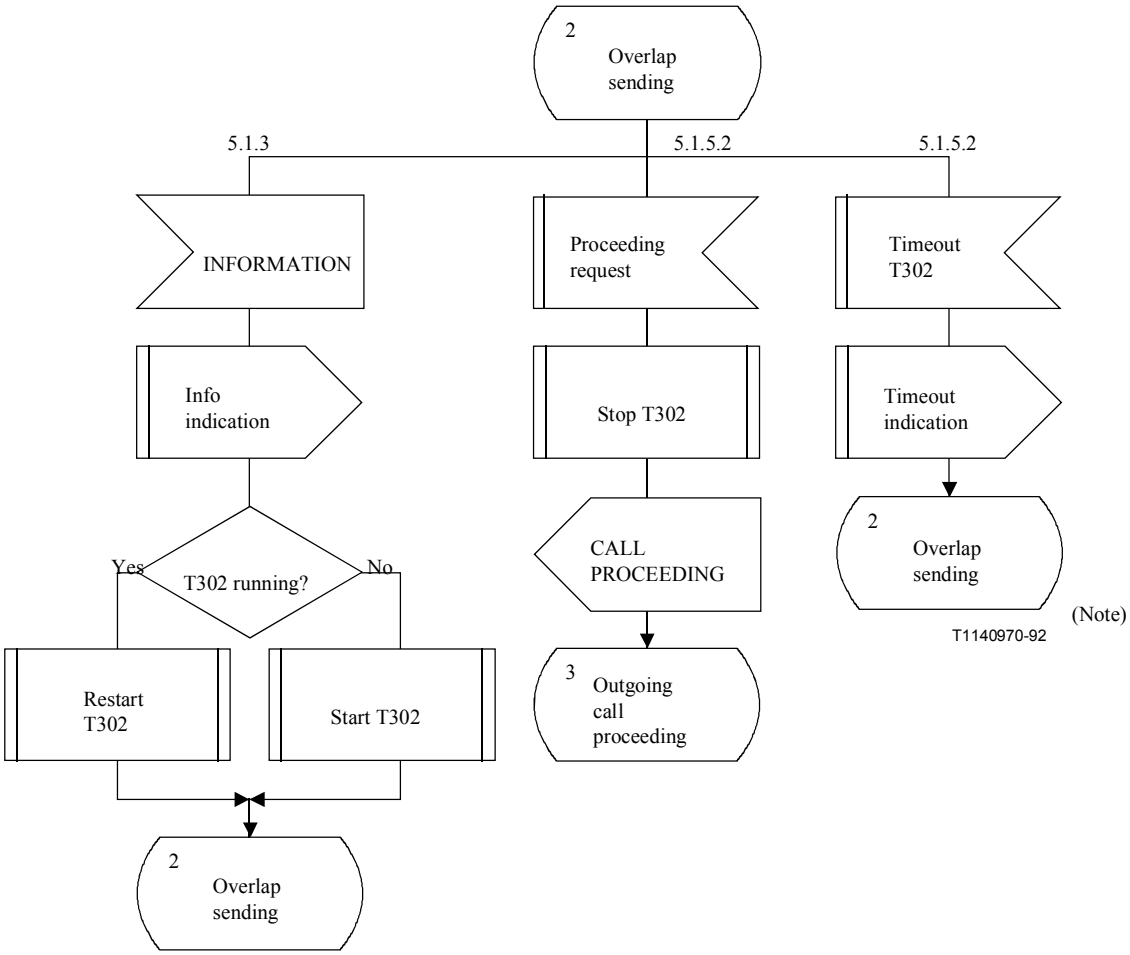


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 3 of 28)*



NOTE – It is assumed that the CC functional block will carry out the functions of 5.1.5.2 and 5.1.7.

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 4 of 28)*

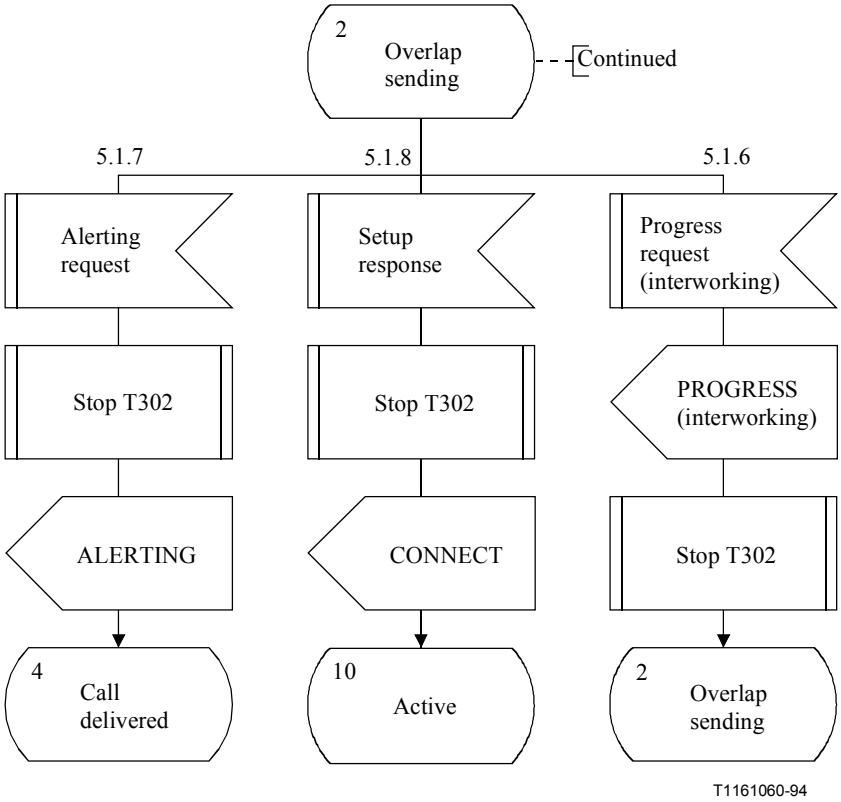


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 5 of 28)*

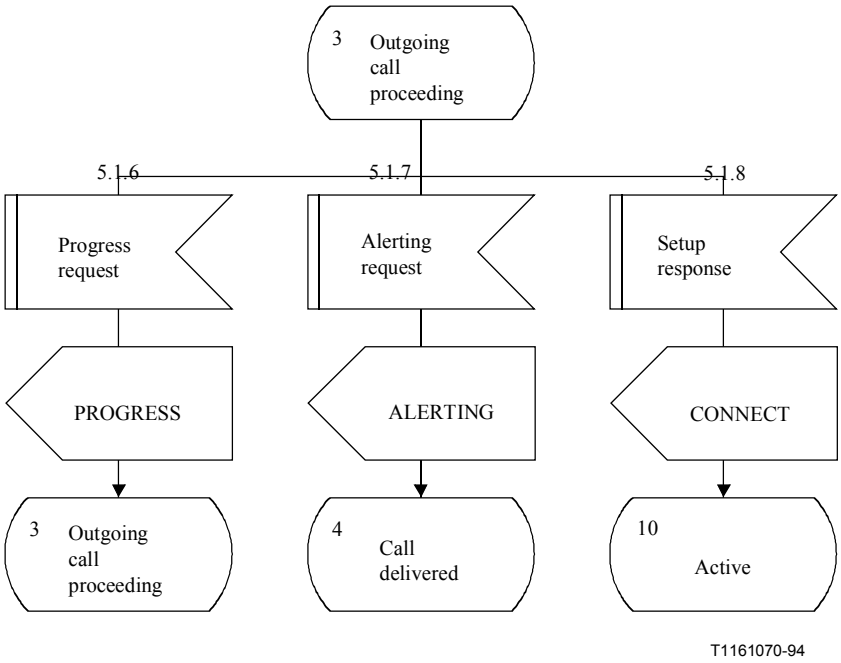


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 6 of 28)*

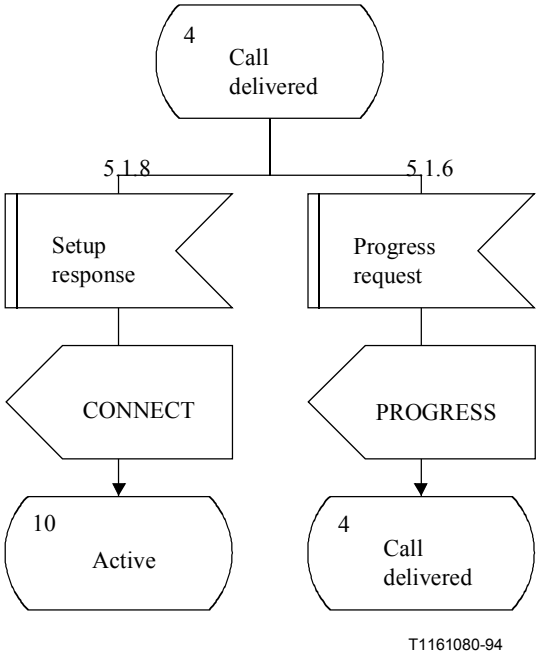
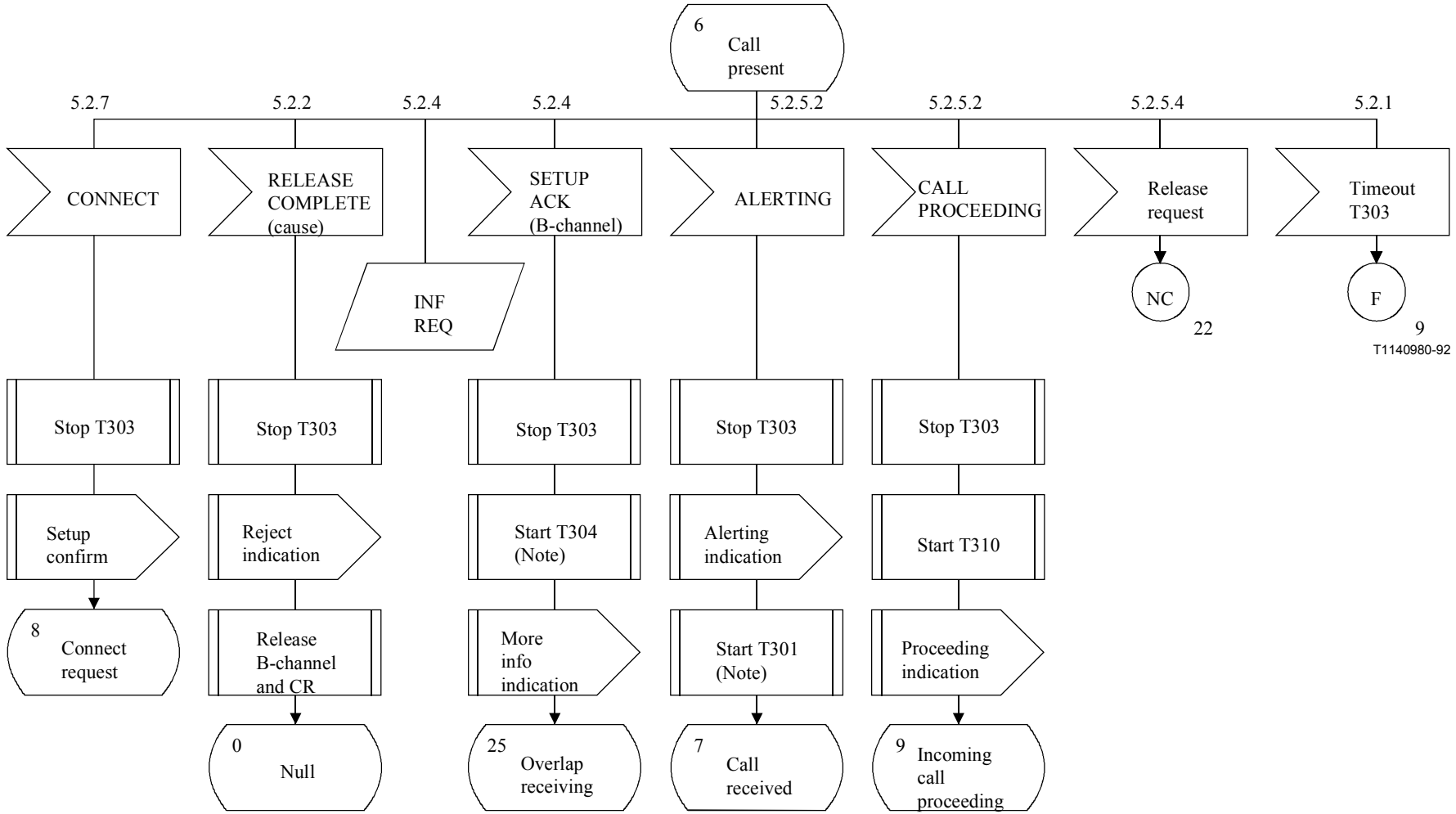


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 7 of 28)*



NOTE – T301 and T304 are optional (see 9.1).

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 8 of 28)*

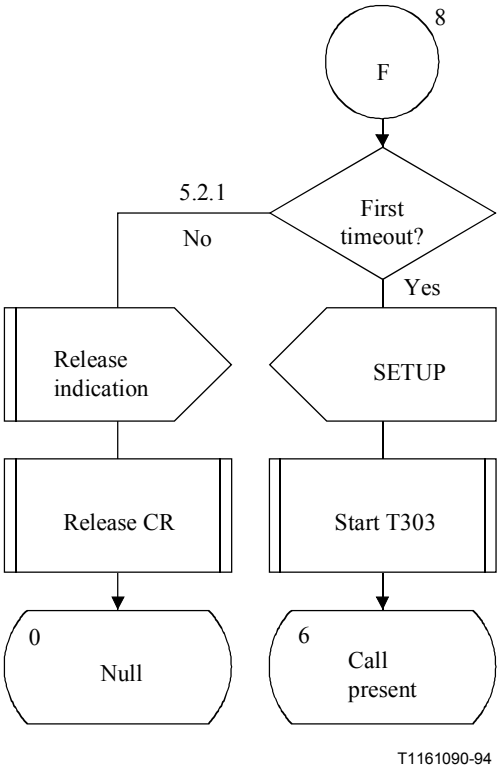
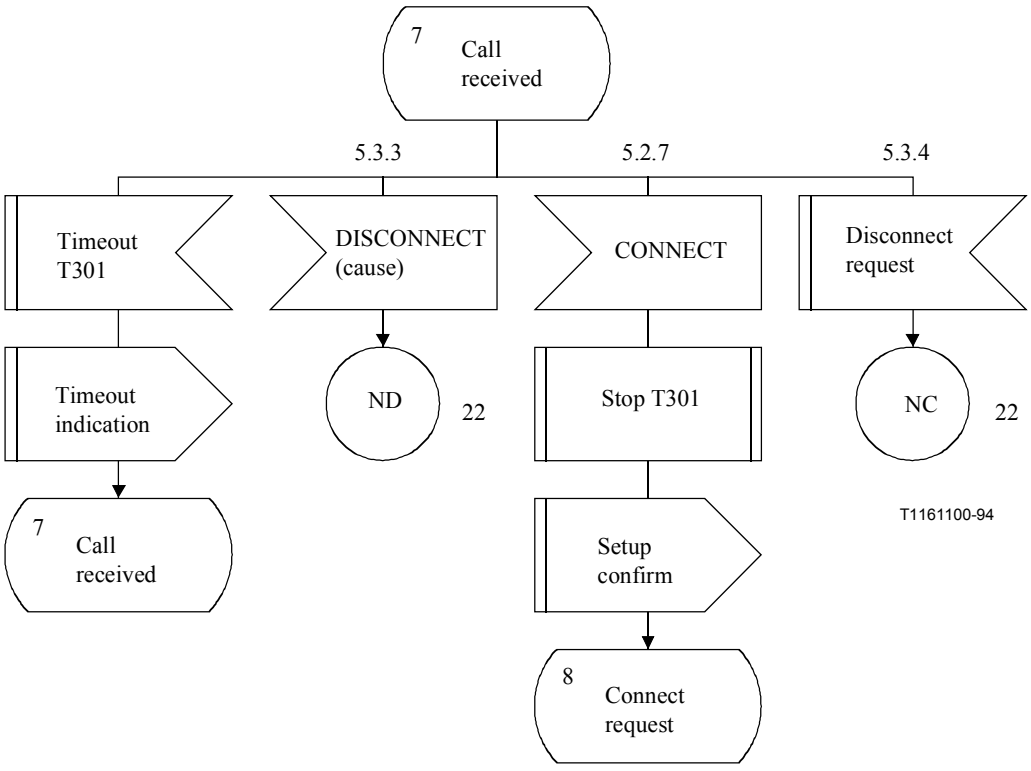


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 9 of 28)*



NOTE – T301 est facultatif (voir 9.1).

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 10 of 28)*

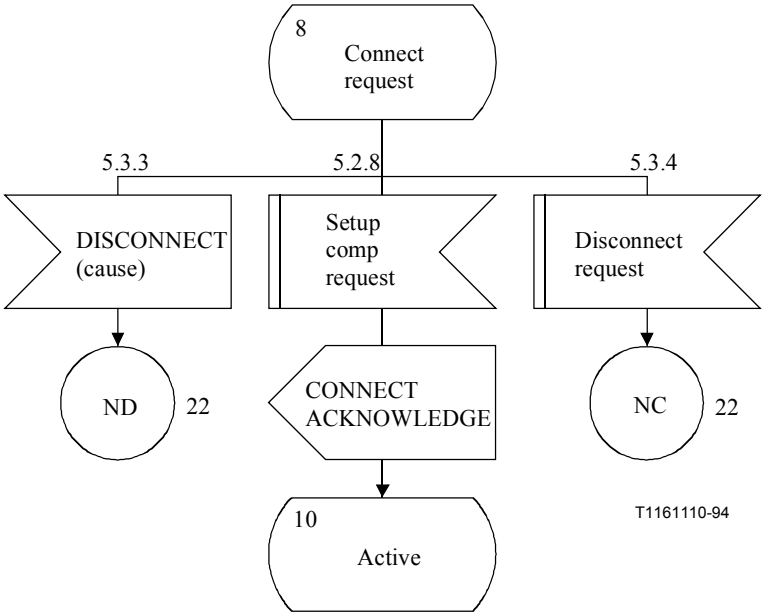
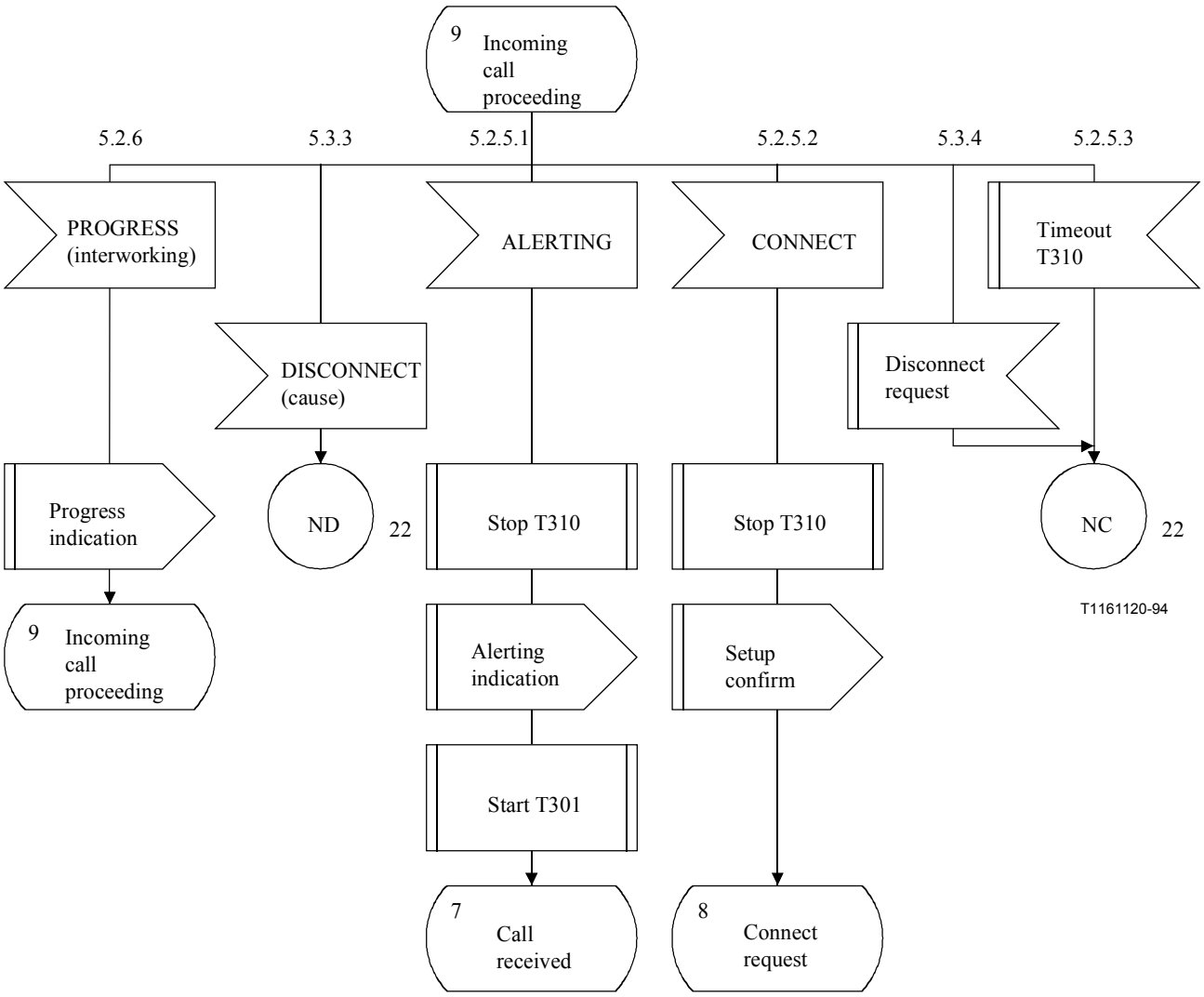


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 11 of 28)*



NOTE – T301 is optional (see 9.1).

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 12 of 28)*

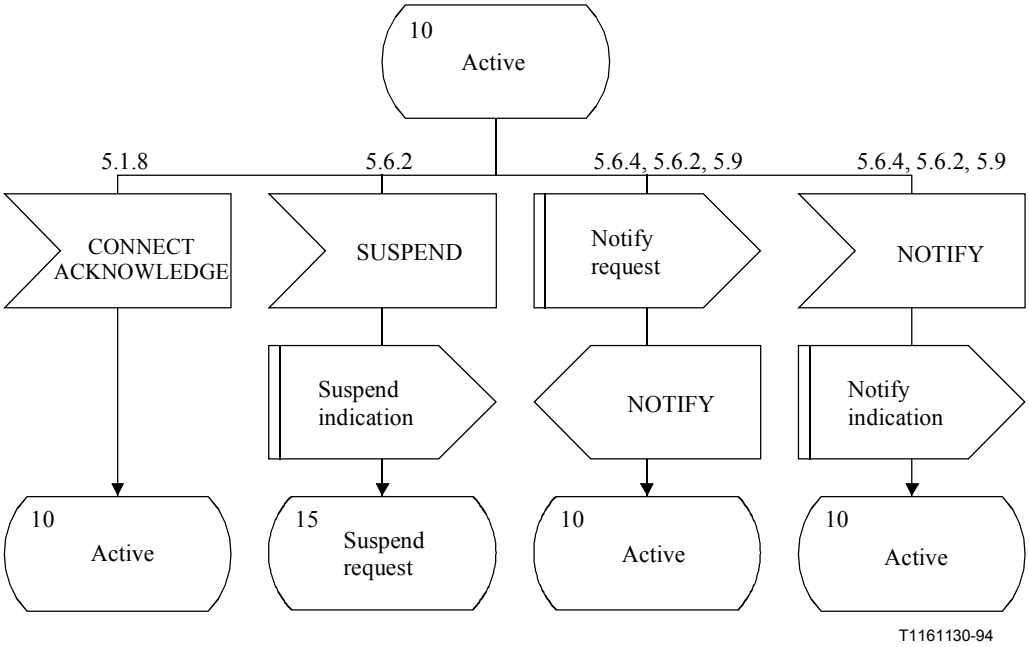


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 13 of 28)*

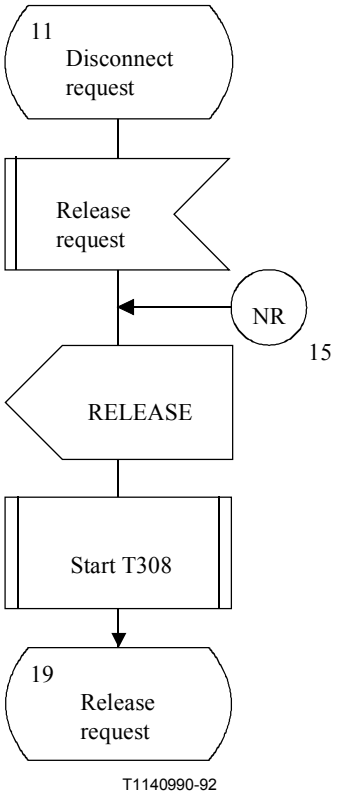


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 14 of 28)*

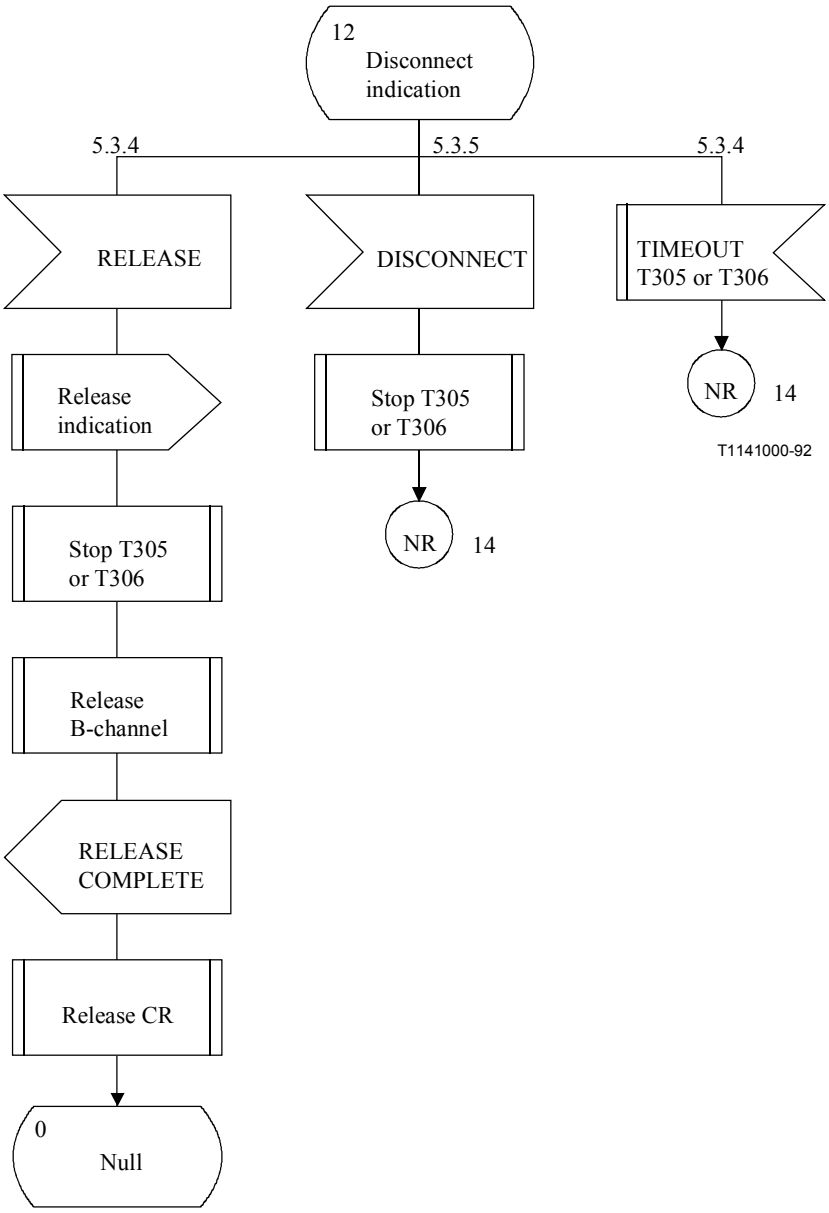


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 15 of 28)*

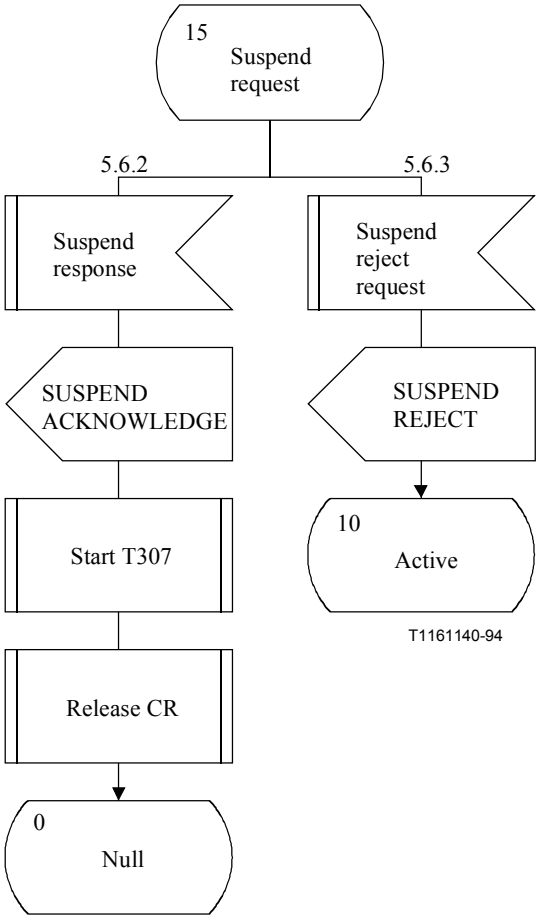


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 16 of 28)*

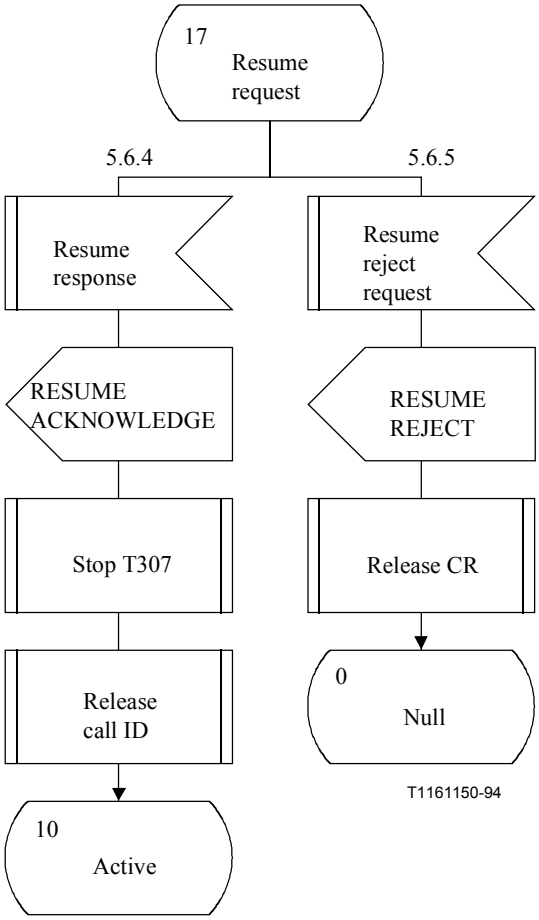


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 17 of 28)*

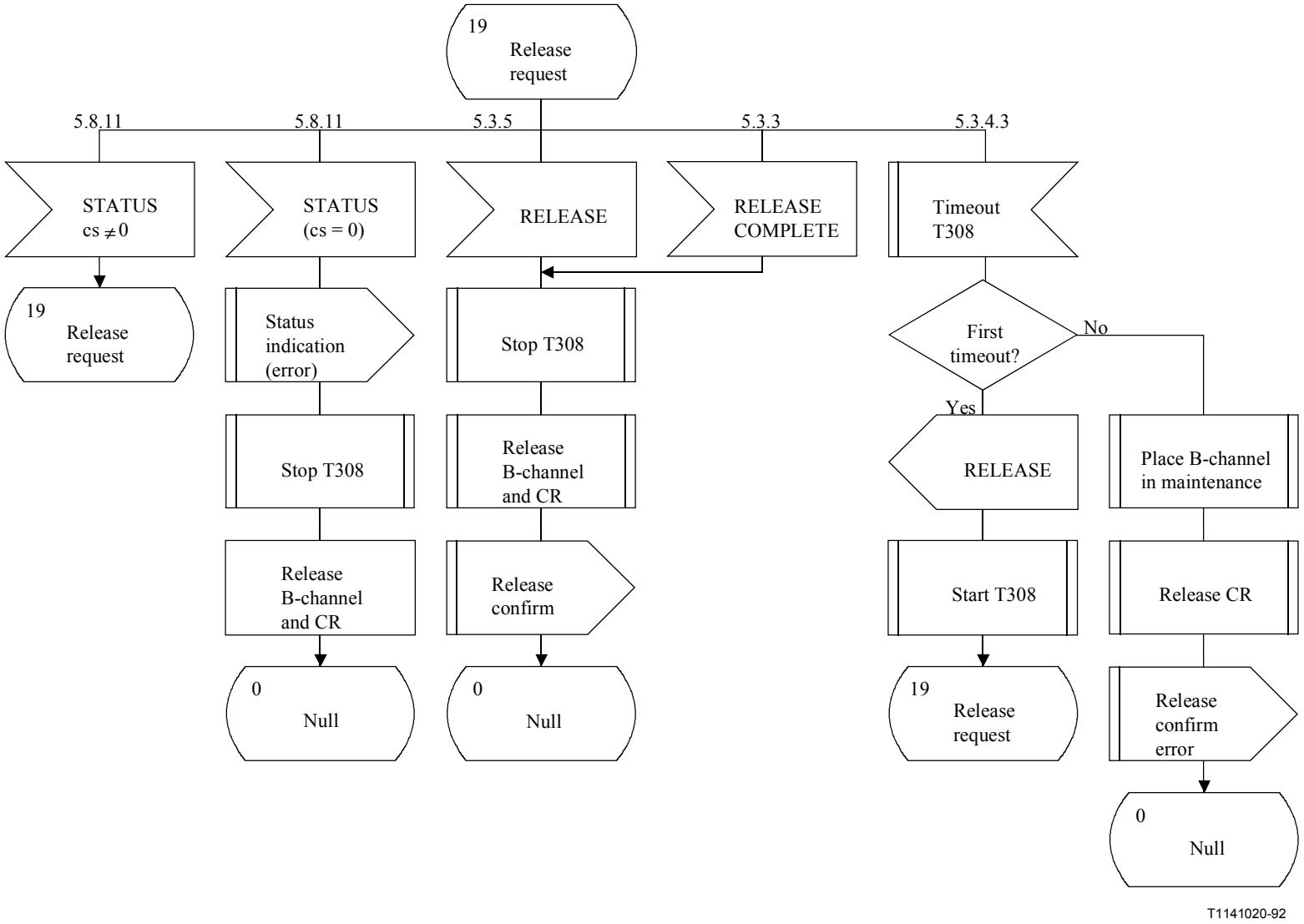
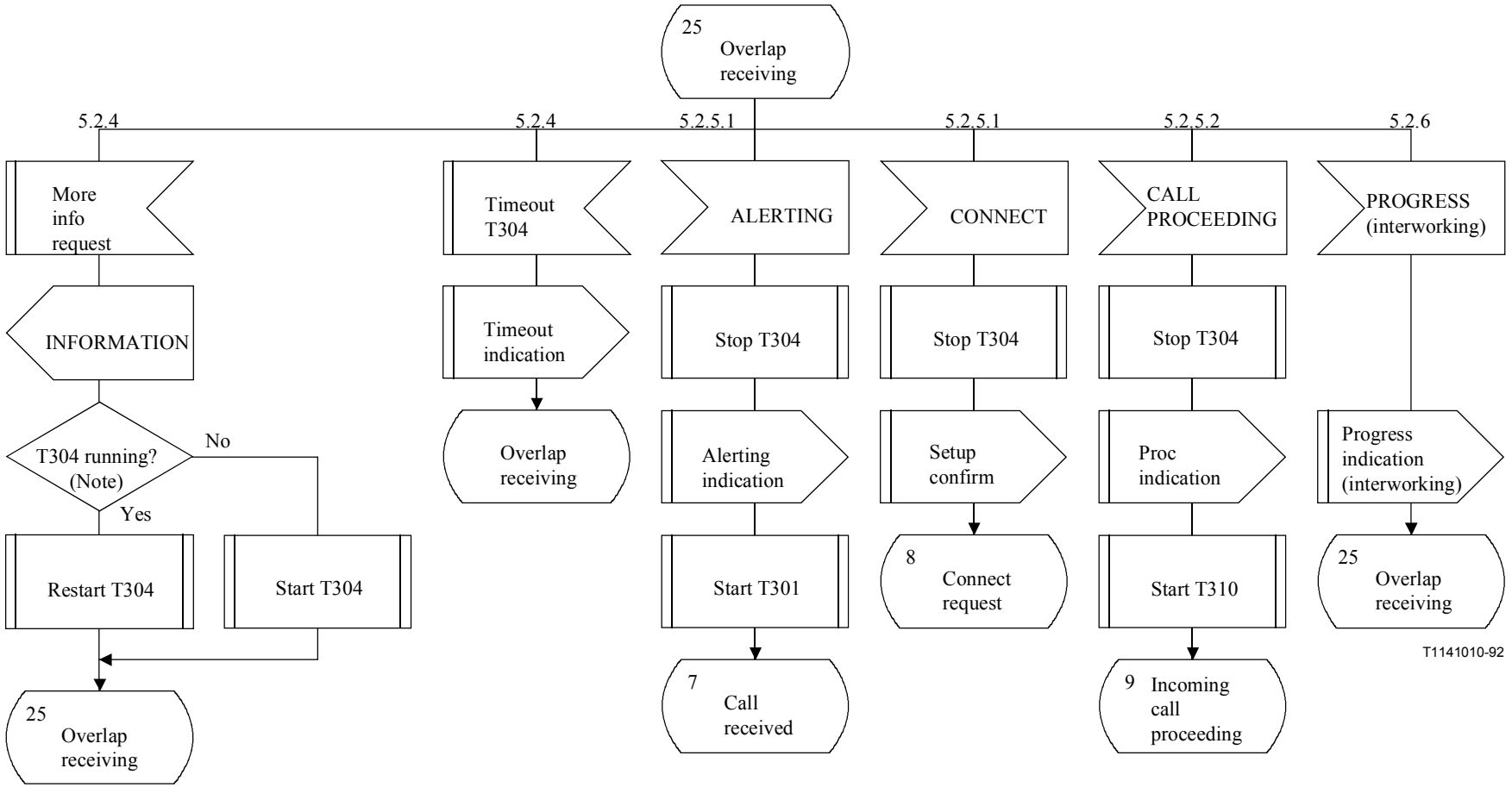
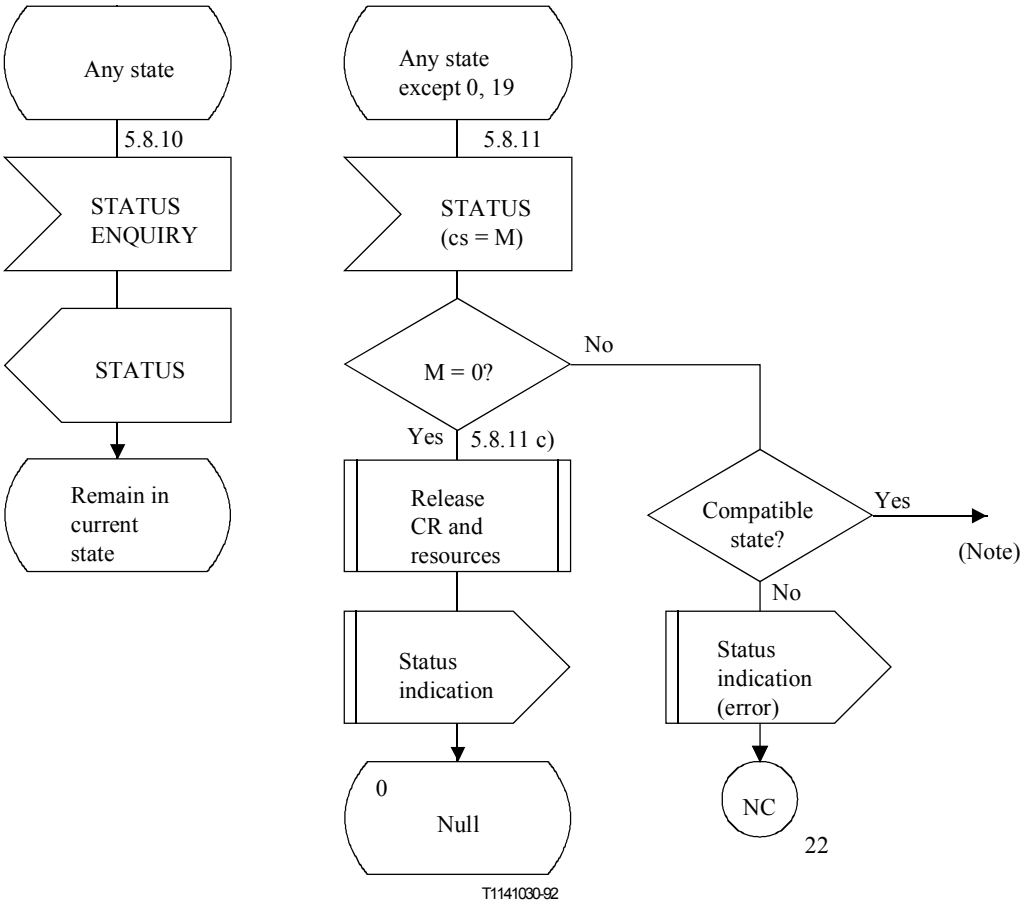


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 18 of 28)*



NOTE – T304 is optional (see 9.1).

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 19 of 28)*



NOTE – Action on receipt of STATUS indicating a compatible call state is implementation dependent. (See 5.8.11.)

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 20 of 28)*

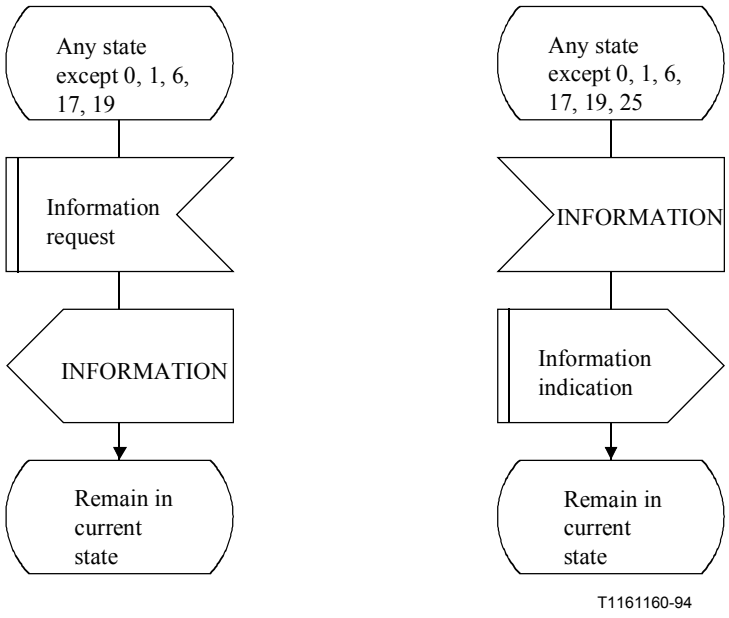
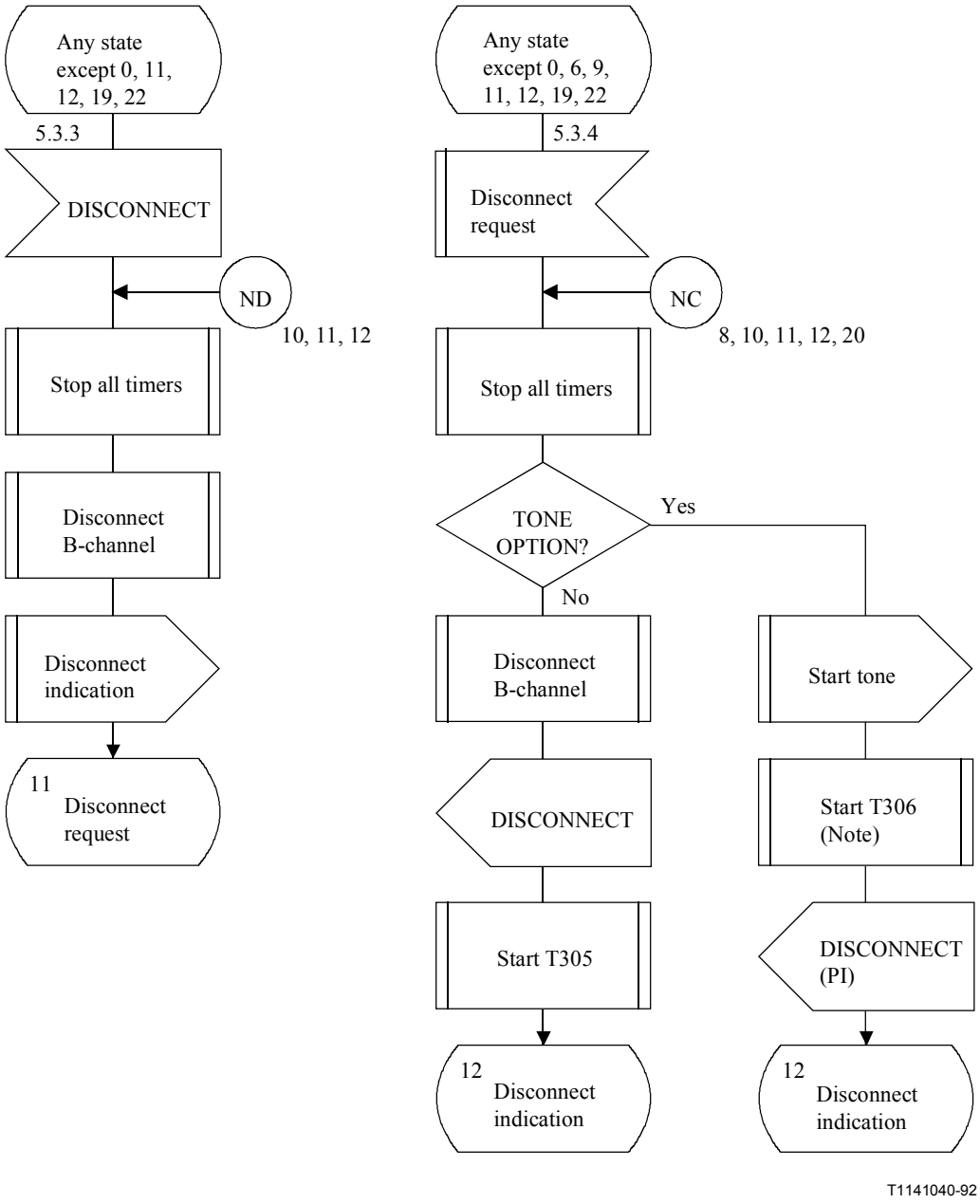
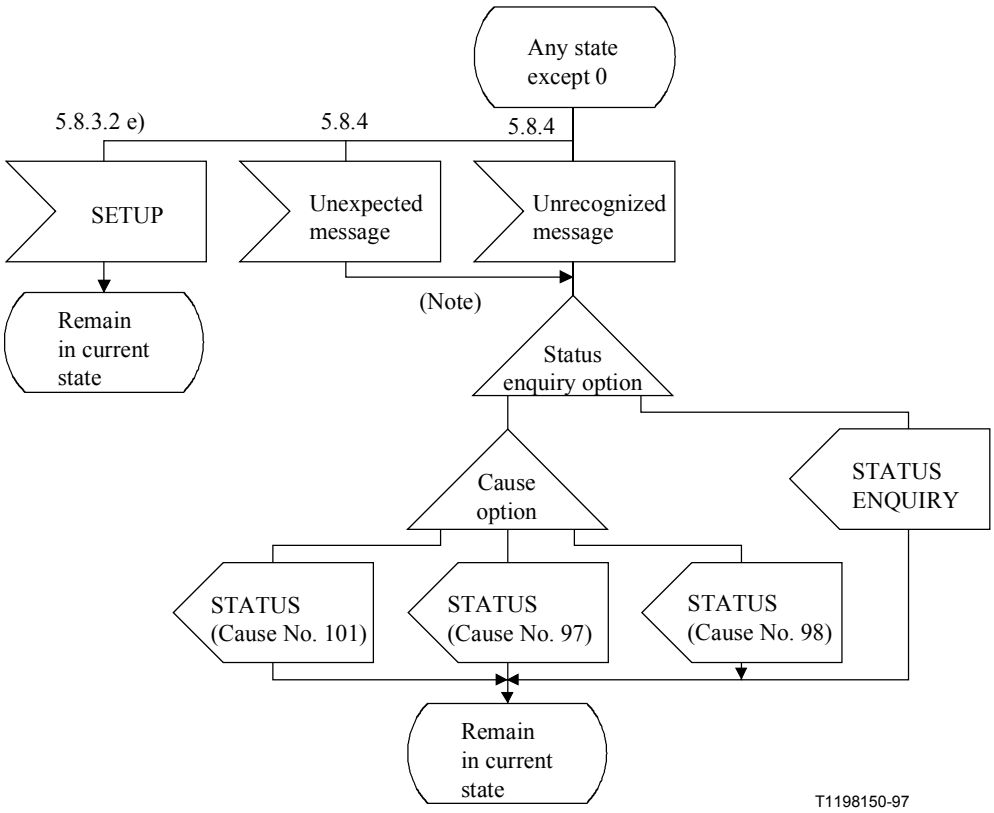


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 21 of 28)*



NOTE – See 9.1 for default values of T306.

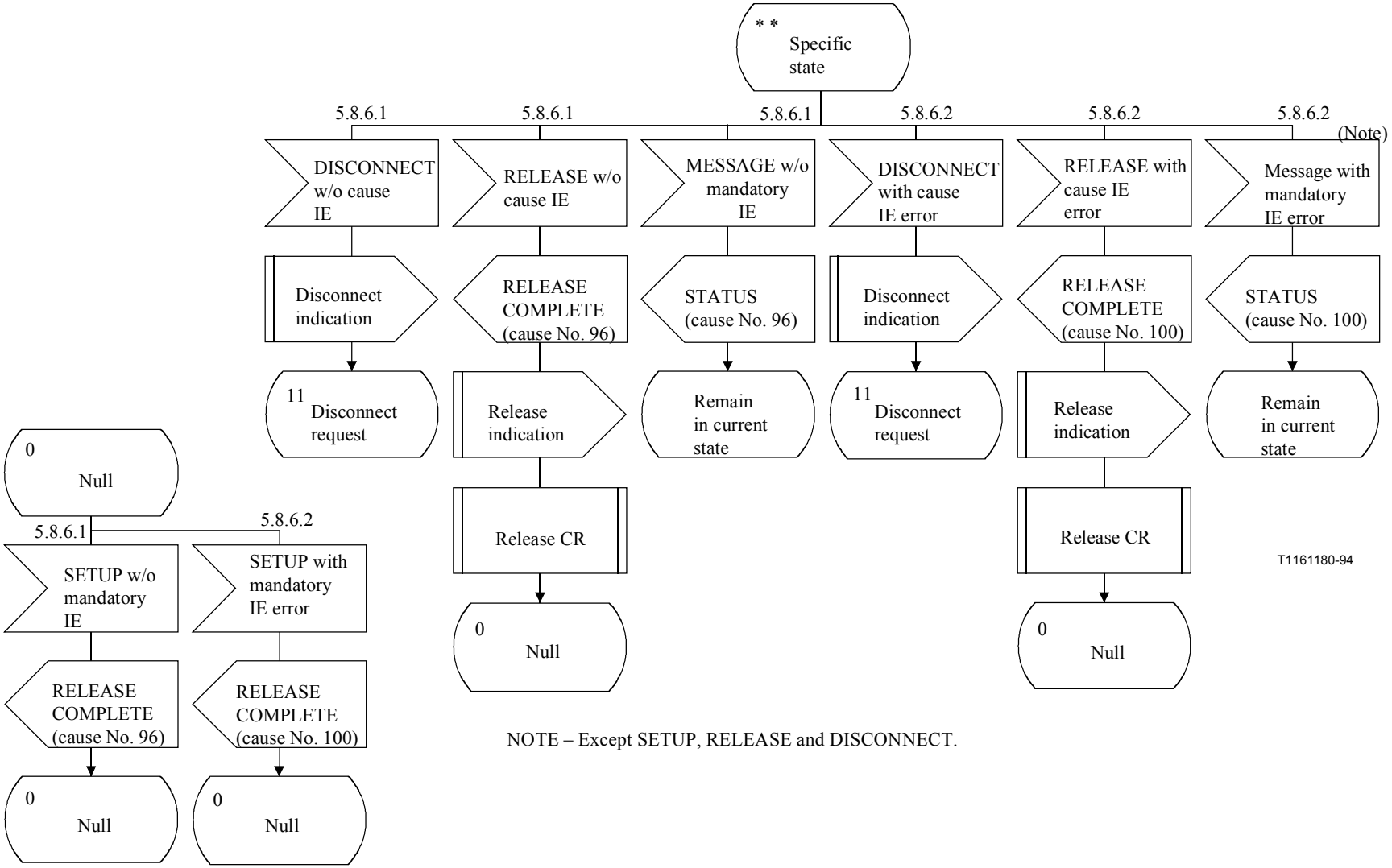
Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 22 of 28)*



a) Error handling SDL diagram (1 of 2)

NOTE – Except RELEASE or REALEASE COMPLETE.

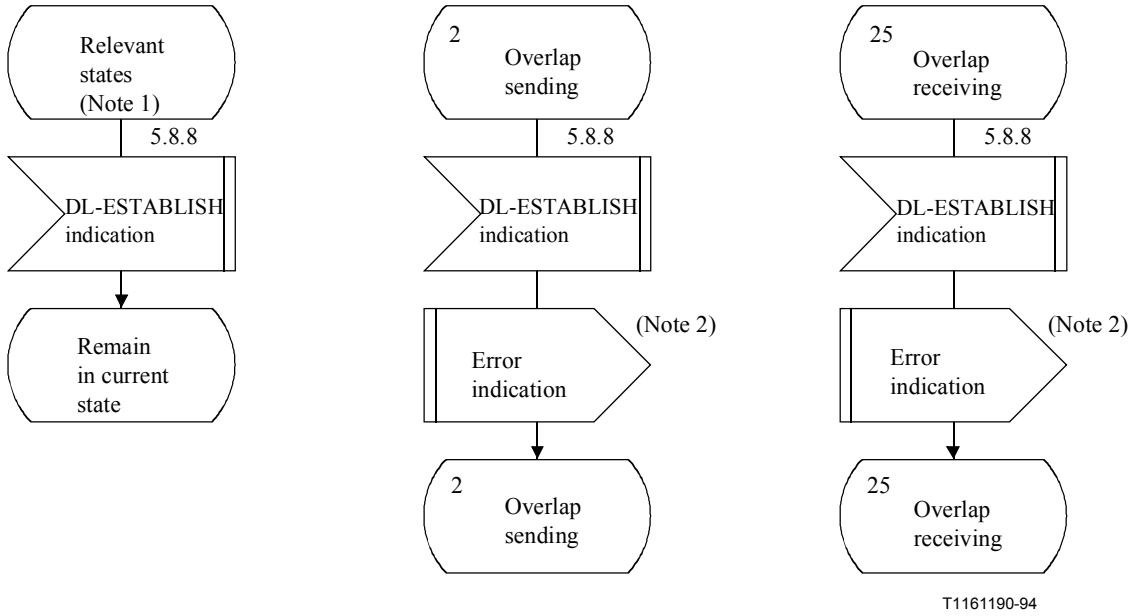
Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 23 of 28)*



NOTE – Except SETUP, RELEASE and DISCONNECT.

a) Error handling SDL diagram (2 of 2)

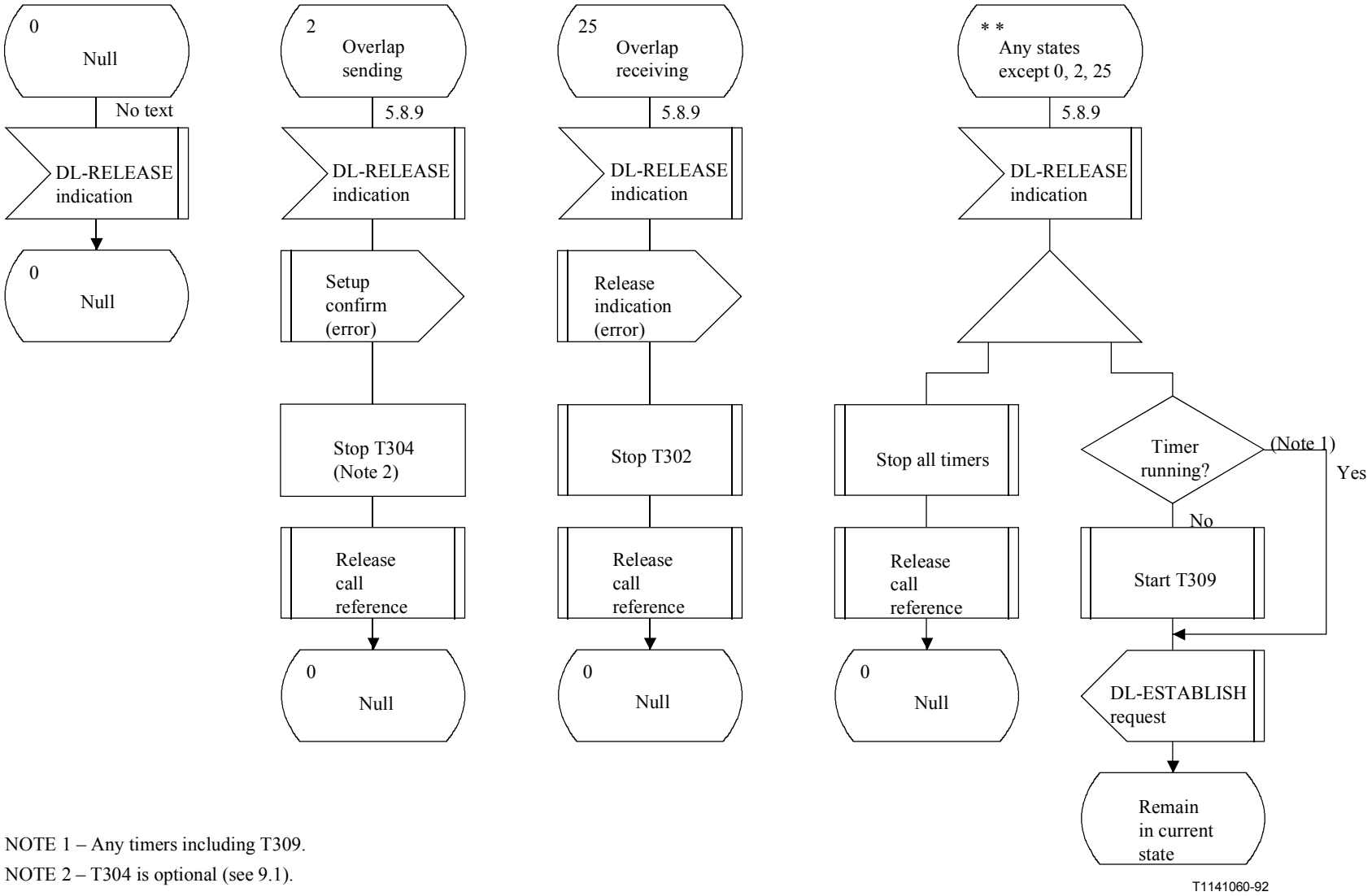
Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 24 of 28)*



NOTE 1 – The relevant states are as follows: N1, N3, N6 to N12, N15, N17, N19.

NOTE 2 – At the reception of this primitive, the call control should clear the call by sending disconnect request primitives.

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 25 of 28)*



NOTE 1 – Any timers including T309.

NOTE 2 – T304 is optional (see 9.1).

Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 26 of 28)*

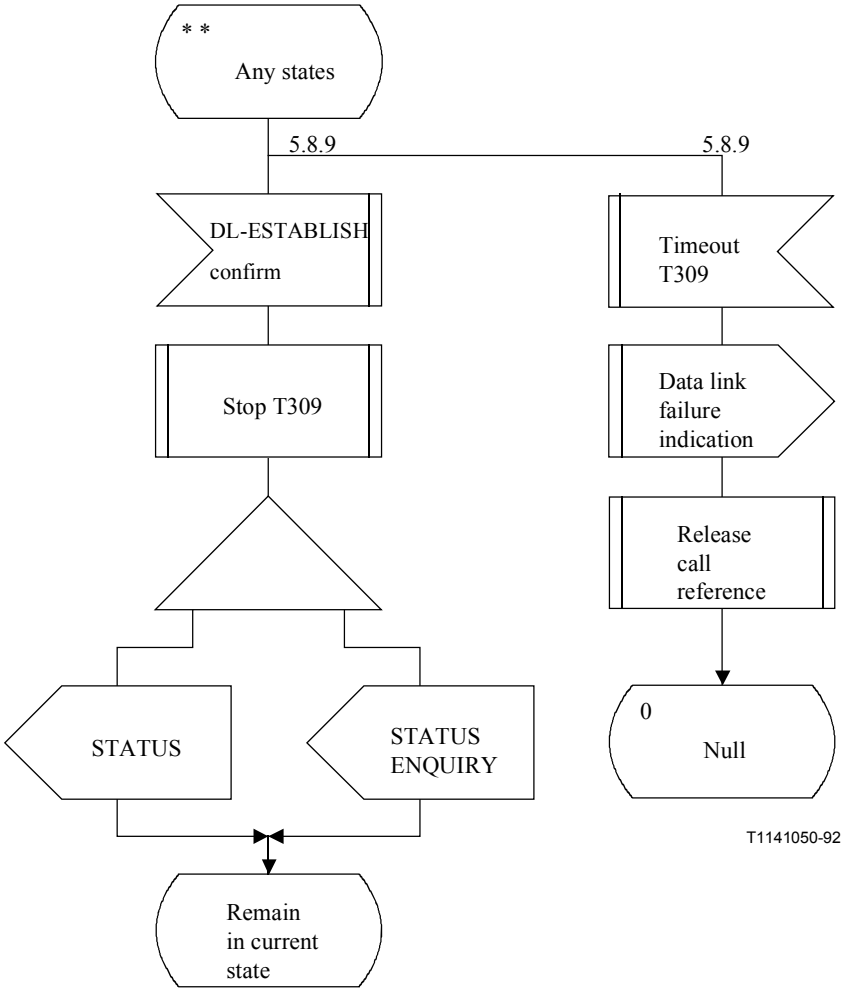


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 27 of 28)*

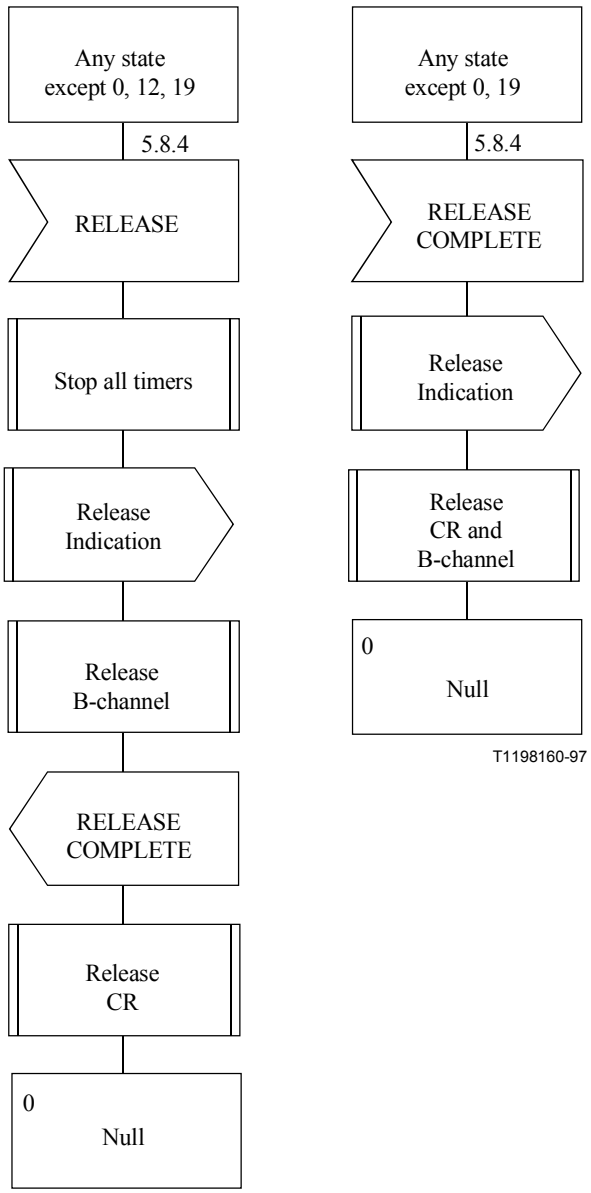


Figure A.6/Q.931 – Detailed protocol control (network side) point-point *(sheet 28 of 28)*

ANNEX B

Compatibility and address checking

B.1 Introduction

This Annex describes the various compatibility and address checks which should be carried out to ensure that best match of user and network capabilities is achieved on a call within an ISDN.

This Annex also covers interworking with existing networks.

Three different processes of checking shall be performed:

i) at the user-to-network interface on the calling side (see B.2);

ii) at the network-to-user interface on the called side (see B.3.2); and

iii) user-to-user (see B.3.3).

NOTE – In this context and throughout this Annex, the term "called user" is the endpoint entity which is explicitly addressed. This may be an addressed Interworking Unit (IWU); see the I.500-series Recommendations.

For details on the coding of the information required for compatibility checking, see Annex I.

B.2 Calling side compatibility checking

At the calling side, the network shall check that the bearer service requested by the calling user in the Bearer capability information element matches with the bearer services provided to that user by the network. If a mismatch is detected, then the network shall reject the call using one of the causes listed in 5.1.5.2.

Network services are described in Recommendations I.230 [47] and I.240 [48] as bearer services and teleservices, respectively.

B.3 Called side compatibility and address checking

In this subclause, the word "check" means that the user examines the contents of the specified information element.

B.3.1 Checking of addressing information

If an incoming SETUP message is offered with addressing information (i.e. either DDI or sub-addressing or the appropriate part of the called party number), the following actions will occur:

a) If a number or subaddress is assigned to a user, then the information in a Called party number or Called party subaddress information element of the incoming call shall be checked by the user against the corresponding part of the number assigned to the user or the user's own subaddress. In case of a mismatch, the user shall ignore the call. In the case of match, the compatibility checking described in B.3.2 and B.3.3 will follow.

b) If a user has no assigned number or subaddress, then the Called party number and Called party subaddress information element shall be ignored. Then, compatibility checking described in B.3.2 and B.3.3 will follow.

NOTE 1 – According to user's requirements, compatibility checking can be performed in various ways from the viewpoint of execution order and information to be checked, e.g. first assigned number/subaddress and then compatibility or vice versa.

NOTE 2 – If an incoming call, offered with addressing information, is always to be awarded to the addressed user, all users connected to the same passive bus should have an assigned number or subaddress.

B.3.2 Network-to-user compatibility checking

When the network is providing a bearer service at the called side, the user shall check that the bearer service offered by the network in the Bearer capability information element matches the bearer services that the user is able to support. If a mismatch is detected, then the user shall either ignore or reject the offered call using cause No. 88, *incompatible destination*, (see 5.2.2).

B.3.3 User-to-user compatibility checking

The called side terminal equipment shall check that the content of the Low layer compatibility information element is compatible with the functions it supports.

The Low layer compatibility information element (if available) shall be used to check compatibility of low layers (e.g. from layer 1 to layer 3, if layered according to the OSI model).

NOTE – The Bearer capability information element is also checked (see B.3.2). Therefore, if any conflict from duplication of information in the Bearer capability and the Low layer compatibility information elements is detected, this conflict shall be resolved according to Annex I, e.g. the conflicting information in the Low layer compatibility information element shall be ignored.

If the Low layer compatibility information element is not included in an incoming SETUP message, the Bearer capability information element shall be used to check the compatibility of low layers.

The called terminal equipment may check the High layer compatibility information element (if present) as part of user-to-user compatibility checking procedures, even if the network only supports bearer services.

If a mismatch is detected in checking any of the information elements above, then the terminal equipment shall either ignore or reject the offered call using cause No. 88, *incompatible destination*, (see 5.2.2).

B.3.3.1 User-to-user compatibility checking and bearer service selection

If the called side terminal does not support the semantics of bearer service selection, it will respond as if the fallback bearer capability were only the offered bearer capability in applying B.3.3.

If the called side terminal does support the semantics of bearer service selection, and if it is able to accept the call using either of the bearer capabilities (the fallback bearer capability or the preferred bearer capability), then it shall follow B.3.3 procedures in separately evaluating its compatibility with the offered call for each of the offered bearer capabilities (the fallback bearer capability and the preferred bearer capability).

a) If evaluations show the terminal to be compatible with the call, it shall respond to the call using the preferred bearer capability.

b) If the evaluations show the terminal to be incompatible with the call for one of the two offered bearer capabilities (either the fallback bearer capability or the preferred bearer capability), it shall not answer the call using that bearer capability.

c) If both evaluations show the terminal to be incompatible with the call, it shall follow B.3.3 procedures for incompatible calls.

B.3.4 User action tables

Tables B.1, B.2 and B.3 show the action which shall be carried out as a result of compatibility checking with the calling user's request for a bearer service and/or teleservice.

B.4 Interworking with existing networks

Limitations in network or distant user signalling (e.g. in the case of an incoming call from a PSTN or a call from an analogue terminal) may restrict the information available to the called user in the incoming SETUP message. A called user should accept limited compatibility checking (e.g. without the High layer compatibility information element) if a call is routed from an existing network which does not support High layer compatibility information element transfer.

In cases where the network cannot provide all incoming call information, or where the network is not aware of the existence or absence of some service information (such as a compatibility information), the incoming SETUP message includes a Progress indicator information element, containing progress indicator No. 1, *call is not end-to-end ISDN; further call progress information may be available in band*, or No. 3, *origination address is non-ISDN* (see Annex G).

The terminal equipment receiving a SETUP with a Progress indicator information element shall modify its compatibility checking, the terminal equipment should regard the compatibility as successful if it is compatible with the included information, which as a minimum, will be the Bearer

capability information element. A terminal equipment expecting information in addition to the Bearer capability information element in a full ISDN environment need not reject the call if such information is absent but a Progress indicator information element is included.

Table B.1/Q.931 – Bearer capability compatibility checking

BC mandatory info element	Point-to-point data link (Note 1)	Broadcast data link (Note 1)	
Compatible	Proceed	Proceed	
Incompatible	Reject (5.2.5.1)	Ignore [5.2.5.1 a)] (Note 2)	Reject [5.2.5.1 b)] (Note 2)

Table B.2/Q.931 – Low layer and high layer compatibility checking – Compatibility assured with the available description of the call

LLC/HLC Compatibility assured	Point-to-point data link (Note 1)		Broadcast data link (Note 1)		
Compatible	**Accept**		**Accept**		
Incompatible	Reject (5.2.5.1)	Attempt low layer compatibility negotiation (Annex J)	Ignore [5.2.5.1 a)] (Note 2)	Reject [5.2.5.1 b)] (Note 2)	Attempt low layer compatibility negotiation (Annex J)

Table B.3/Q.931 – Low layer and high layer compatibility checking – Compatibility not assured with the available description of the call

LLC/HLC Compatibility not assured	Point-to-point data link (Note 1)		Broadcast data link (Note 1)	
HLC or LLC Present	Accept or reject (Note 3)	Attempt low layer compatibility negotiation (Annex J)	Accept or reject (Note 3)	Attempt low layer compatibility negotiation (Annex J)

NOTES to Tables B.1, B.2 and B.3

NOTE 1 – For broadcast data link terminal equipment which is explicitly addressed using subaddressing or the appropriate part of the called party number, the point-to-point column in the above table shall be used.

NOTE 2 – When a terminal equipment on a broadcast data link is incompatible, an option of "ignore or reject" is permitted (see 5.2.2).

NOTE 3 – Some terminal equipment on this interface may understand the High layer compatibility or Low layer compatibility information elements and would reject the call if incompatible.

ANNEX C

Transit network selection

This Annex describes the processing of the Transit network selection information element.

C.1 Selection not supported

Some networks may not support transit network selection. In this case, when a Transit network selection information element is received, that information element is processed according to the rules for unimplemented non-mandatory information elements (see 5.8.7.1).

C.2 Selection supported

When transit network selection is supported, the user identifies the selected transit network(s) in the SETUP message. One Transit network selection information element is used to convey a single network identification.

The user may specify more than one transit network. Each identification is placed in a separate information element. The call would then be routed through the specified transit networks in the order listed in the SETUP. For example, a user lists networks A and B, in that order, in two Transit network selection information elements within a SETUP message. The call is first routed to network A (either directly or indirectly), and then to network B (either directly or indirectly), before being delivered.

As the call is delivered to each selected network, the corresponding transit selection may be stripped from the call establishment signalling, in accordance with the relevant internetwork signalling arrangement. The Transit network selection information element(s) is (are) not delivered to the destination user.

No more than four Transit network selection information elements may be used in a single SETUP message.

When a network cannot route the call because the route is busy, the network shall initiate call clearing in accordance with 5.3 with cause No. 34, *no circuit/channel available*.

If a network does not recognize the specified transit network, the network shall initiate call clearing in accordance with 5.3, with cause No. 2, *no route to specified transit network*. The diagnostic field shall contain a copy of the contents of the Transit network selection information element identifying the unreachable network.

A network may screen all remaining Transit network selection information elements to:

a) avoid routing loops; or

b) ensure that an appropriate business relationship exists between selected networks; or

c) ensure compliance with national and local regulations.

If the transit network selection is of an incorrect format, or fails to meet criteria a), b) or c), the network shall initiate call clearing in accordance with 5.3, with cause No. 91, *invalid transit network selection*.

When a user includes the Transit network selection information element, pre-subscribed default Transit network selection information (if any) is overridden.

ANNEX D

Extensions for symmetric call operation

D.1 Additional message handling

In symmetric applications, the SETUP message will contain a Channel Identification information element indicating a particular B-channel to be used for the call. A point-to-point data link shall be used to carry the SETUP message.

The procedure described in clause 5 for the user side should normally be followed. Where additional procedures are required, they are detailed below.

D.1.1 B-channel selection – Symmetric interface

Only B-channels controlled by the same D-channel will be the subject of the selection procedure. The selection procedure is as follows:

a) The SETUP message will indicate one of the following:
   1) channel is indicated, no acceptable alternative; or
   2) channel is indicated, any alternative is acceptable.

b) In cases 1) and 2), if the indicated channel is acceptable and available, the recipient of the SETUP message reserves it for the call. In case 2), if the recipient of the SETUP message cannot grant the indicated channel, it reserves any other available B-channel associated with the D-channel.

c) If the SETUP message included all information required to establish the call, the recipient of SETUP message indicates the selected B-channel in a CALL PROCEEDING message transferred across the interface and enters the Incoming Call Proceeding state.

d) If the SETUP message did not include all the information required to establish the call, B-channel is indicated in a SETUP ACKNOWLEDGE message sent across the interface. The additional call establishment information, if any, is sent in one or more INFORMATION messages transferred across the interface in the same direction as the SETUP message. When all call establishment information is received, a CALL PROCEEDING, ALERTING or CONNECT message, as appropriate, is transferred across the interface.

e) In case 1) if the indicated B-channel is not available, or in case 2) if no B-channel is available, a RELEASE COMPLETE message with a cause value of No. 44, *requested circuit/channel not available*, or No. 34, *no circuit/channel available*, respectively, is returned to the initiator of the call. The sender of this message remains in the Null state.

f) If the channel indicated in the CALL PROCEEDING or SETUP ACKNOWLEDGE message is unacceptable to the initiator of the call, it clears the call in accordance with 5.3.

D.1.2 Call confirmation

Upon receipt of a SETUP message, the equipment enters the Call Present state. Valid responses to the SETUP message are a SETUP ACKNOWLEDGE, an ALERTING, a CALL PROCEEDING, a CONNECT or a RELEASE COMPLETE message.

If the indicated channel is acceptable to the initiator of the call, the initiator shall attach to the indicated B-channel.

D.1.3 Clearing by the called user employing user-provided tones/announcements

In addition to the procedures described in 5.3.3, if the bearer capability is either audio or speech, the called user or private network may apply in-band tones/announcements in the clearing phase. When in-band tones/announcements are provided, the DISCONNECT message contains progress indicator No. 8, *in-band information or appropriate pattern is now available*, and the called user or private network proceeds similarly as stipulated in 5.3.4.1 for the network.

D.1.4 Active indication

Upon receipt of a CONNECT message, the initiator of the call shall respond with a CONNECT ACKNOWLEDGE message and enter the Active state.

D.2 Timers for call establishment

User endpoints implement the network side timers T301, T303 and T310 along with the corresponding network side procedures for actions taken upon expiration of these timers. See Table 9-2 for the call establishment user-side timers and procedures.

D.3 Call collisions

In symmetric arrangements, call collisions can occur when both sides simultaneously transfer a SETUP message indicating the same channel. In the absence of administrative procedures for assignment of channels to each side of the interface, the following procedure is employed.

First, one side of the interface will be designated the *network* and the other side of the interface will be designated the *user*. Second, for the three possible scenarios where the same channel is indicated by combinations of preferred and exclusive from the user and network sides, the following procedure is used:

a) *Network preferred, user preferred*

The network preferred channel is awarded and an alternate channel is indicated in the first response to the user SETUP message.

b) *Network exclusive, user exclusive*

The network exclusive channel is awarded and the user SETUP message is cleared with a RELEASE COMPLETE message with cause No. 34, *no circuit/channel available*.

c) *Network preferred, user exclusive, or network exclusive, user preferred*

The side of the interface with an exclusive indicator in a SETUP message is awarded the channel and an alternate channel is indicated in the first response to the side using a preferred indicator in the SETUP message.

Channel identification is allowed in both directions for ALERTING and CONNECT.

ANNEX E

Network-specific facility selection

This Annex describes the processing of the Network-specific facilities information element. The purpose of this information element is to indicate which network facilities are being invoked.

E.1 Default provider

When the length of the network identification field is set to zero in the Network-specific facilities information element, then the services identified in this information element are to be provided by the network side of the interface receiving the information element (default provider). If the Network-specific facilities information element is recognized but the network facilities are not understood, then this information element is processed according to rules for non-mandatory information element content error (see 5.8.7.2).

E.2 Routing not supported

Some networks may not support the routing to the remote network of the contents of the Network-specific facilities information element. In this case, when a Network-specific facilities information element is received, that information element is processed according to the rules for unrecognized information elements (see 5.8.7.1).

E.3 Routing supported

When Network-specific facility information element routing is supported, the user identifies the network provider in this information element in the Q.931 SETUP message. One Network-specific facility information element is used to identify a network provider.

The user may specify more than one network provider by repeating the Network-specific facilities information element. Each identification is placed in a separate information element. The information is routed to the indicated network provider as long as the call is also handled by the network provider (see Annex C, Transit network selection). For example, if the user lists network providers A and B in separate Network-specific facilities information elements in a call control message, there must be corresponding Transit network selection information elements in the SETUP message identifying those networks (or default call routing via A and B that was established prior to call establishment).

As the signalling messages containing Network-specific facilities information elements are delivered to the indicated remote network, they may be stripped from the signalling messages, in accordance with the relevant internetworking signalling arrangement. The Network-specific facilities information elements may be delivered to the identified user.

No more than four Network-specific facilities information elements may be used in a SETUP message. When the information element is repeated, the order of presentation of the elements in a message is not significant. Further, there does not have to be a one-to-one correspondence between Network-specific facilities information elements and Transit network selection information elements.

If a network cannot pass the information to the indicated network provider, either due to:

– the network indicated is not part of the call path; or

– no mechanism exists for passing the information to identified network,

the network shall initiate call clearing in accordance with 5.3, with cause No. 2, *no route to specified transit network*. The diagnostic field may optionally contain a copy of the first 5 octets of the Network-specific facilities information element.

When the user includes the Network-specific facilities information element in the SETUP message, pre-subscribed default service treatment (if any) is overridden.

ANNEX F

D-channel backup procedures

F.0 Foreword

The procedure defined in this Annex can be used when non-associated signalling is applied to multiple primary rate access arrangements. This feature can be provided on a subscription basis and is network dependent.

F.1 General

In associated signalling, the D-channel signalling entity can only assign calls to channels on the interface containing the D-channel. When the D-channel signalling entity can assign calls to channels on more than one interface (including the one containing the D-channel), this is called non-associated signalling. Figure F.1 is an example of associated signalling used on each of the three interfaces between a user (e.g. a PABX) and a network. Replacing associated signalling with non-associated signalling on these interfaces results in the example shown in Figure F.2.

When non-associated signalling is employed, the reliability of the signalling performance for the ISDN interfaces controlled by the D-channel may be unacceptable. To improve the reliability, a D-channel backup procedure employing a standby D-channel is necessary. The next subclause describes the backup procedure which is optional for endpoints that use non-associated signalling.

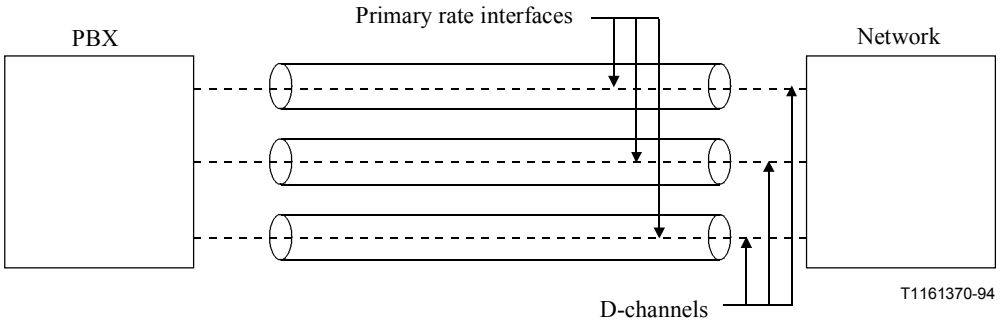


Figure F.1/Q.931 – Example of associated signalling on each of the three primary rate interfaces

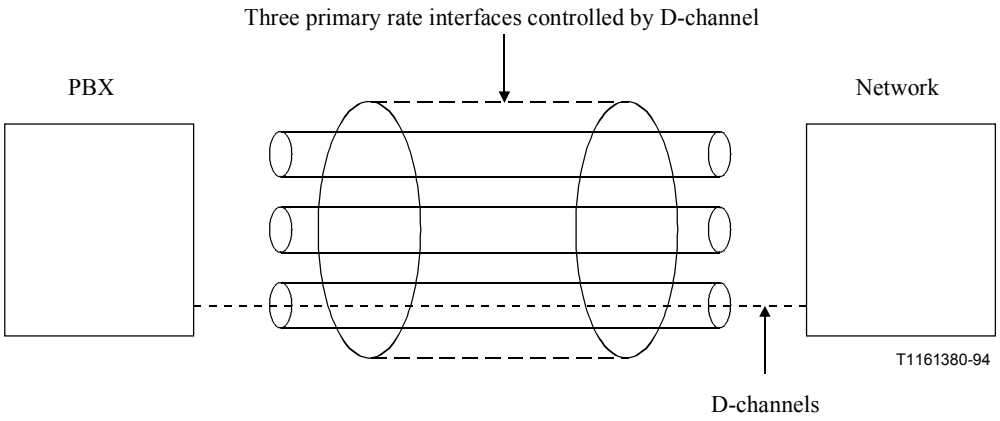


Figure F.2/Q.931 – Example of non-associated signalling controlling three primary rate interfaces

F.2 D-channel backup procedure

F.2.1 Role of each D-channel

When two or more interfaces connect a network and a user, a primary D-channel (labelled "one") is always present on one interface. On a different interface, a secondary D-channel (labelled "two") is present that can also send signalling packets. Figure F.3 shows the addition of a secondary (i.e. backup) D-channel to the arrangement shown in Figure F.2.

D-channel one is used to send signalling packets across the user-network interface for multiple interfaces including the interface containing D-channel two. D-channel two is in a standby role and is active at layer 2 only. All SAPI groups (e.g. 0, 16 and 63) are alive and can send packets. At periodic intervals determined by the appropriate layer 2 timer associated with SAPI 0, a link audit frame will be sent on the point-to-point signalling link with DLCI = 0 of D-channel two.

Since D-channel two is in a standby role, load sharing between D-channels one and two is not possible. Furthermore, D-channel two cannot serve as a B-channel when it is in a standby role. Lastly, D-channel two can only back up the signalling functions provided by D-channel one and not some other D-channel on a different interface.

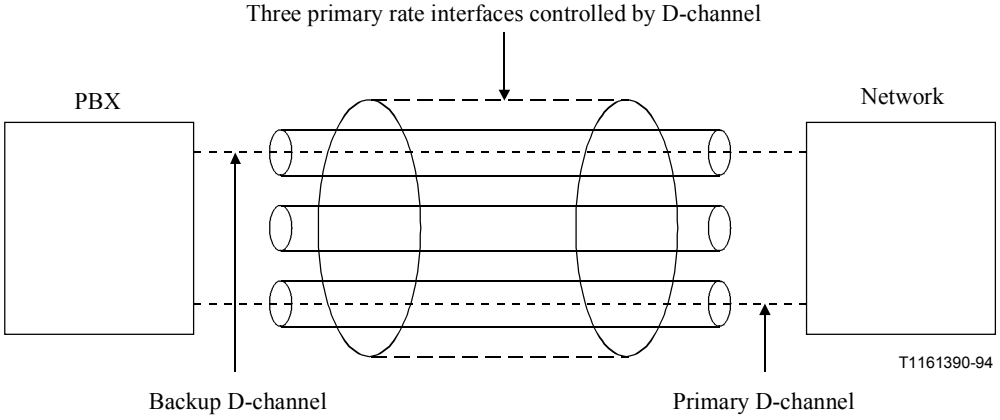


Figure F.3/Q.931 – Example of non-associated signalling with backup D-channel controlling three primary rate interfaces

F.2.2 Switch-over of D-channels

Failure of D-channel one is determined by the receipt of a DL-RELEASE indication primitive from the data link layer. At this point, optionally additional attempts to re-establish this D-channel may be initiated. Otherwise, it is assumed that D-channel one has failed.

Two states are defined for any D-channel in a backup arrangement. A D-channel is termed out-of-service when layer 2 remains in the TEI-assigned state, after being periodically requested by layer 3 to establish multiple-frame operation. A D-channel is termed maintenance busy when layer 2 is held in the TEI-assigned state by layer 3. While in the maintenance busy condition, the response to an invitation for link establishment is met with the transmission of a DM (Disconnected Mode).

When the D-channel one has failed and if D-channel two is not in an out-of-service condition, the layer 3 shall place D-channel one in a maintenance busy condition, start timer T321 and then issue a DL-ESTABLISH request primitive to re-initialize SAPI 0 link 0 of D-channel two. Upon receipt of this primitive, the data link layer issues an SABME command. Timer T200 is started. The end receiving the SABME command on D-channel two follows the remainder of the Q.921 procedures for establishing logical link with DLCI = 0.

Once the logical link with DLCI = 0 in D-channel two is in the Link Established state, the procedure to establish layer 3 call control signalling can begin on the link.

To establish the backup D-channel for carrying call control signalling, layer 3 should issue an appropriate layer 3 message (e.g. a STATUS ENQUIRY on stable call reference numbers). Once a response to that layer 3 message is received, D-channel two is declared to be the active D-channel, normal layer 3 call control signalling may proceed, timer T321 is stopped, and D-channel one is moved to the out-of-service condition. If the maintenance busy timer T321 expires before a response is received to the layer 3 message, D-channel one is moved to the out-of-service condition and an attempt is made to establish the logical link with DLCI = 0 on D-channel one and D-channel two.

If the logical link with DLCI = 0 of both D-channel one and D-channel two are initialized simultaneously, the designated primary shall be chosen as the D-channel for carrying call control signalling. The designated primary D-channel is agreed upon at subscription time by both sides of the interface.

After a switch-over, old D-channel two becomes the new D-channel one and old D-channel one becomes the new D-channel two.

Upon completion of appropriate maintenance activity to D-channel two, the logical links for SAPI = 0 and 63 are made active at layer 2 and the D-channel is removed from the out-of-service condition.

D-channels may only be switched again by a failure of D-channel one or a routing or maintenance request from a peer entity.

ANNEX G

Use of progress indicators

This Annex describes the use of the different progress indicator values defined in 4.5.22. Examples of use are given.

– **Progress indicator No. 1** – Indicates that interworking with a non-ISDN has occurred within the network or networks through which the call has traversed.

– **Progress indicator No. 2** – Indicates that the destination user is not ISDN.

– **Progress indicator No. 3** – Indicates that the origination user is not ISDN.

– **Progress indicator No. 4** – Indicates that a call which had left the ISDN has returned to the ISDN at the same point it had left due to redirection within the non-ISDN. This progress indicator would be employed when a prior Q.931 message resulted in a progress indicator No. 1 (*call is not end-to-end ISDN*), being delivered to the calling user.

The use of progress indicators Nos. 1, 2 and 3 is exemplified in the following.

Four interworking situations are identified in Figure G.1:

a) interworking with another network;

b) interworking with a non-ISDN user connected to ISDN;

c) interworking with non-ISDN equipment within the calling or called user's premises;

d) interworking with another network behind the T reference point.

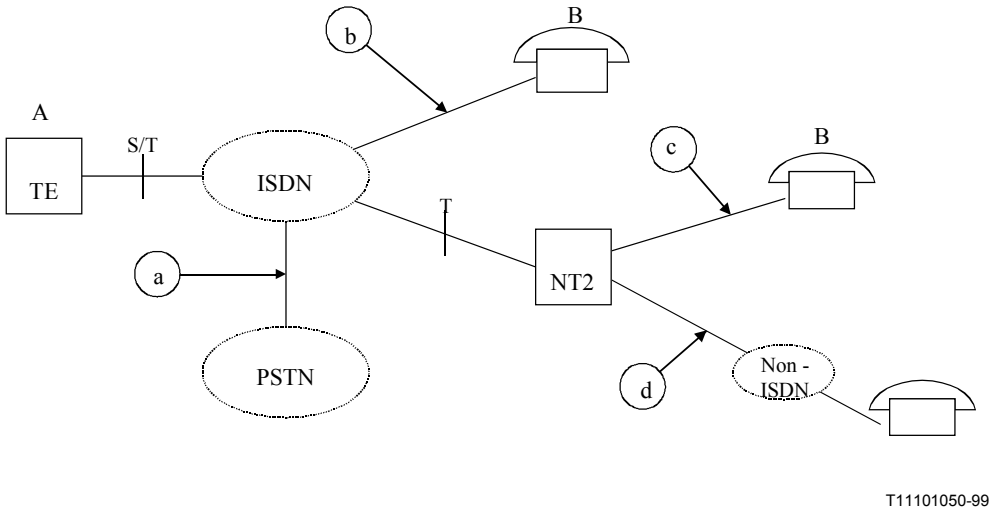


Figure G.1/Q.931

As regards calls from A the following applies:

- case a) – progress indicator No. 1 sent to A;
- case b) – progress indicator No. 2 sent to A;
- case c) – progress indicator No. 2 sent to A (location sub-field = private network);
- case d) – progress indicator No. 1 sent to A (location sub-field = private network).

As regards calls towards A the following applies:

- case a) – progress indicator No. 1 sent to A;
- case b) – progress indicator No. 3 sent to A;
- case c) – progress indicator No. 3 sent to A (location sub-field = private network);
- case d) – progress indicator No. 1 sent to A (location sub-field = private network).

The use of progress indicator No. 4 is exemplified in the following scenarios associated with the Call Forwarding supplementary service. If a call is originated from user A to user B, then as stated above, in the interworking cases b) and c) (see Figure G.1), progress indicator No. 2 shall be sent to user A to indicate that interworking has occurred. If subsequently the call is forwarded from user B to user C, and user C is an ISDN user, progress indicator No. 4 shall be sent to user A.

The use of progress indicator No. 8, *in-band information or appropriate pattern is now available*, is described in clause 5.

ANNEX H

Message segmentation procedures

This optional procedure is used on the basis of bilateral agreement between the user and the network.

H.1 Introduction

Layer 3 messages that are longer than the length of frames that the data link layer can support may be partitioned into several segments.

Message segmentation shall only be used when the message length exceeds N201 (defined in Recommendation Q.921 [3]).

The architectural relationship to other Q.931 functions is shown in Figure H.1. These procedures apply only within a specific data link connection and do not impact the procedures in operation on other parallel data link connections.

In order to support expressed needs for applications requiring message lengths of 10 000 octets, or greater, procedures to support those applications are under study. These procedures will consider backward compatibility and methods to allow information on other call references to be interleaved with segments of a long message. The specifics of these procedures are for further study.

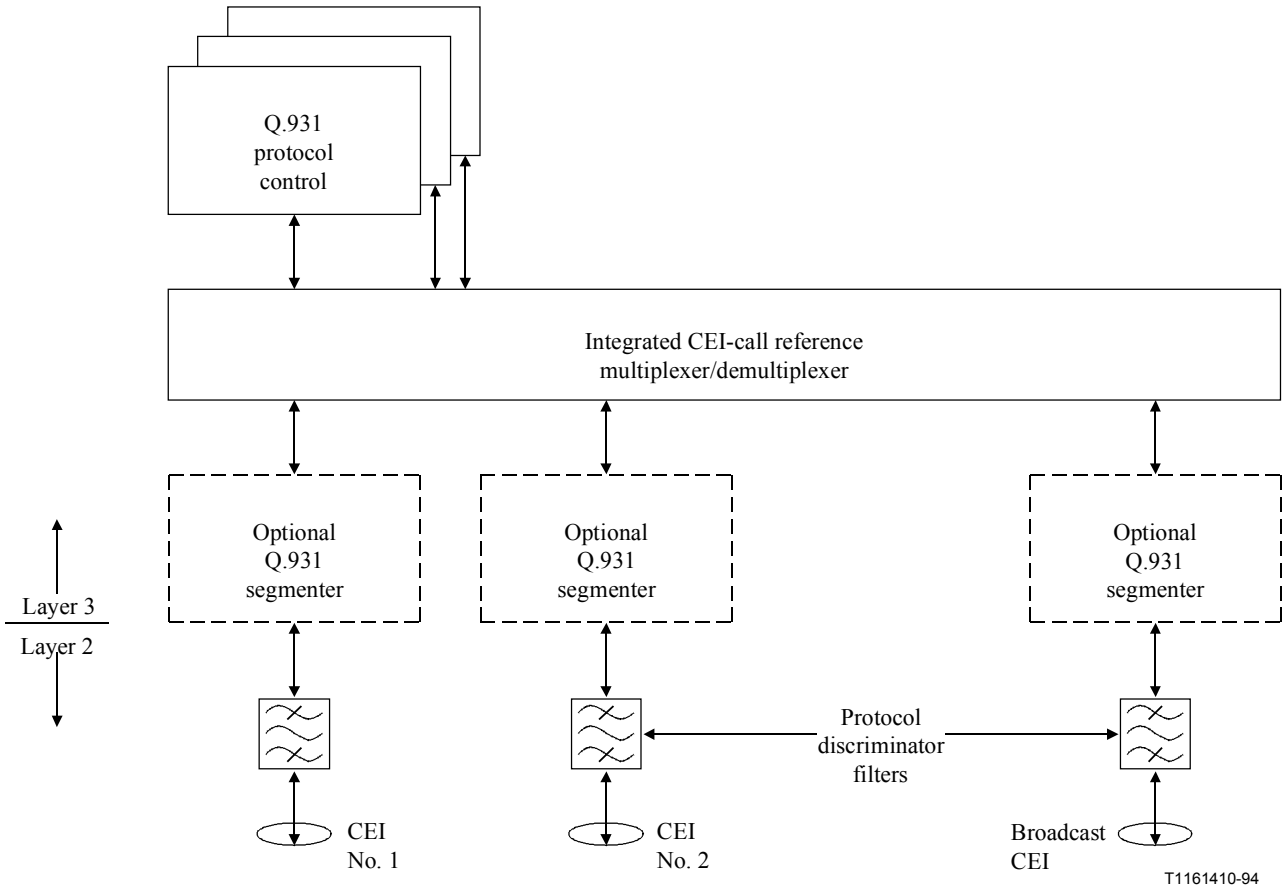


Figure H.1/Q.931 – Logical architecture containing segmentation function

H.2 Message segmentation

The following rules apply when Q.931 messages are to be segmented for transmission:

a) The default maximum number of message segments is 8. If the message is too long to be segmented, then a local maintenance activity shall be notified.

b) The first message segment shall begin with the Protocol discriminator information element immediately followed by the Call reference information element, the segment message type, the Segmented message information element, and octets starting with the first octet following the message type of the message being segmented, subject to the maximum length of the segment not exceeding the maximum size of the data link layer information field.

c) Each subsequent message segment shall begin with the Protocol discriminator information element immediately followed by the Call reference information element, the segment message type, the Segmented message information element, and one or more octets starting with the first octet following the message type of the message being segmented, subject to the maximum length of the segment not exceeding the maximum size of the data link layer information field.

d) The first segment indicator field of the Segmented message information element shall be set to indicate the first segment of a segmented message, and not set in any other segment.

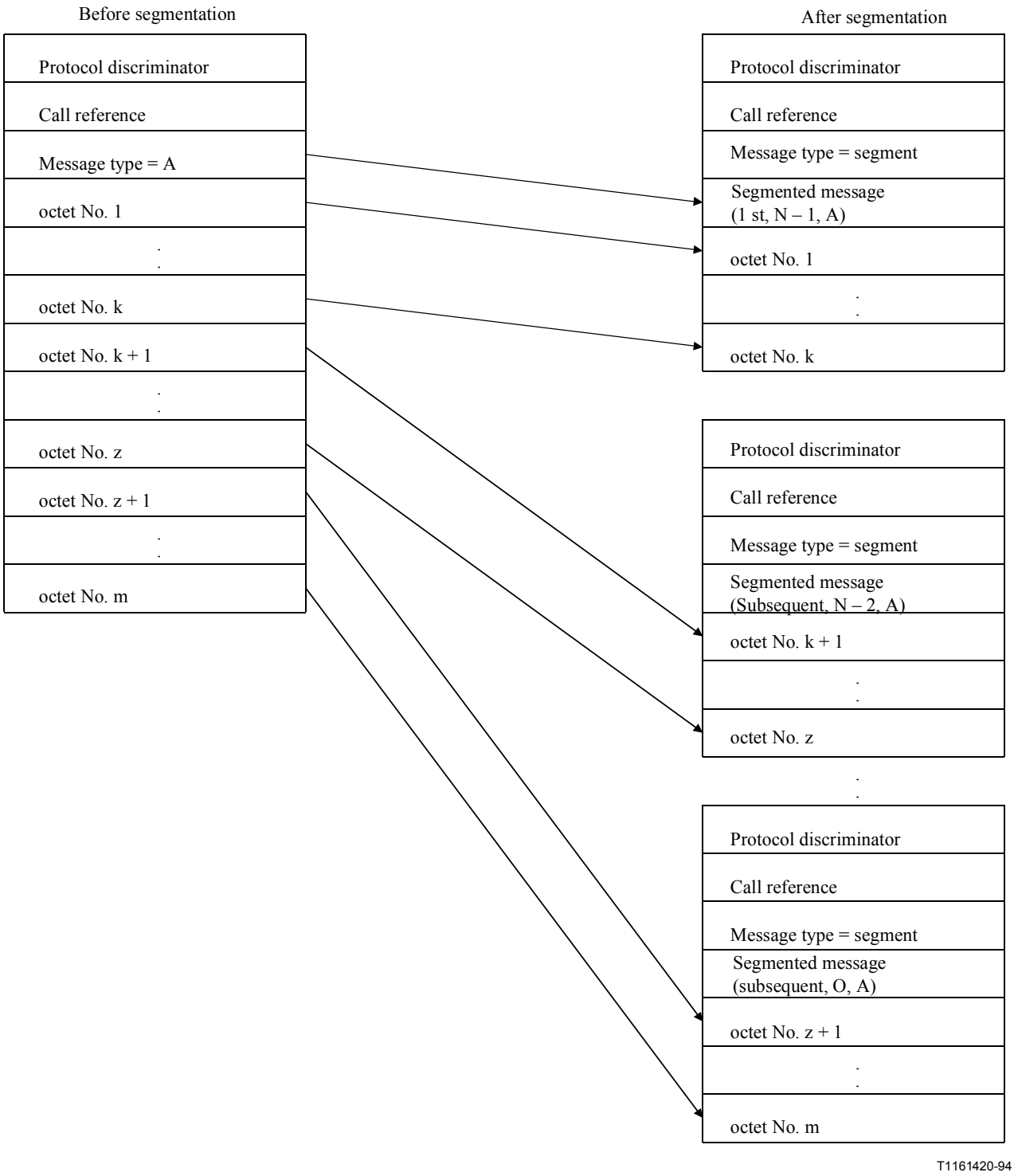
e) The number of segments remaining field of the Segmented message information element shall be set to indicate how many more segments are to be sent, see Figure H.2.

f) The Message type information element shall be coded to indicate a segment message, and the Segmented message information element shall indicate the message type of the original message.

g) Once the first segment has been transmitted on a particular data link connection, then all remaining segments of that message shall be sent (in order) before any other message (segmented or not) for any other call reference is sent on that data link connection, i.e. a segmented message cannot be interleaved with any other messages.

h) In exceptional circumstances, the transmission of a segmented message may be aborted by sending a message or message segment containing a different call reference; sending a message with the message type not coded "segment message", or stopping the transmission of subsequent message segments pertaining to the same message.

i) The octet order for the segmented message shall be preserved regardless of segment boundary.



NOTE – Segmentation may occur at any octet boundary.

Figure H.2/Q.931 – Relation between message and segments

H.3 Reassembly of segmented messages

The following rules apply to the receipt and reassembly of segmented Q.931 messages:

a) A reassembly function, on receiving a message segment containing the Segmented message information element with the first segment indicator indicating "first message", and containing the call reference and message type (coded as "segment message") shall enter the Receiving Segmented Message state and accumulate message segments.

b) Timer T314 shall be initialized or re-initialized upon receipt of a message segment containing the Segmented message information element with a non-zero number of segments remaining field. Timer T314 shall be stopped upon receipt of the last segment, i.e. a message segment containing the Segmented message information element with the number of segments remaining field coded zero. Timer T314 shall not be initialized or re-initialized if error procedures as identified in rules below are initiated.

c) A reassembly function receiving a message segment with a Segmented message information element should wait for receipt of the last message segment pertaining to the same message, i.e. containing the Segmented message information element with the number of segments remaining field coded zero before delivering the message for further Q.931 processing as specified in 5.8. The reassembly function shall enter the Null state.

d) Upon expiry of timer T314, the reassembly function shall discard all segments of this message so far received, notify the layer 3 management entity for the data link connection that message segments have been lost and enter the Null state.

NOTE 1 – Subsequent message segments relating to the same message shall be discarded according to rule f).

e) A reassembly function, upon receiving eight message segments of the same segmented message without receiving a message segment with a number of segments remaining field of the Segmented message information element coded zero, shall discard all message segments so far received, notify the layer 3 management entity for the data link connection that messages have been discarded and enter the Null state.

NOTE 2 – Subsequent message segments relating to the same message shall be discarded according to rule f).

f) A reassembly function, on receiving a message segment containing a Segmented message information element, but with no call reference or Message type information element, while in the Null state shall discard that message segment and remain in the Null state.

g) A reassembly function, on receiving a message segment containing a Segmented message information element, while in the Receiving Segmented Message state with the number of segments remaining field that is not decremented from the number of segments remaining field in the Segmented message information element of the previous message segment, shall discard all segments of this message so far received and enter the Null state.

NOTE 3 – Subsequent message segments relating to the same message shall be discarded according to rule f).

h) If there is a DL-RELEASE indication primitive or DL-ESTABLISH indication primitive received while in the Receiving Segmented Message state, the reassembly function shall discard all received message segments so far received, forward the DL-RELEASE indication primitive or DL-ESTABLISH indication primitive for further Q.931 processing and enter the Null state.

i) A reassembly function, upon receiving a message segment with the first segment indicator of the Segmented message information element indicating "subsequent", while in the Null state, shall discard that message segment and remain in the Null state.

j) A receiving entity, on receiving a message with a different call reference while in the Receiving Segmented Message state, shall discard all segments of the segmented message so far received and enter the Null state. The message received with the new call reference shall receive normal processing.

NOTE 4 – Subsequent message segments relating to the same message shall be discarded according to rule f).

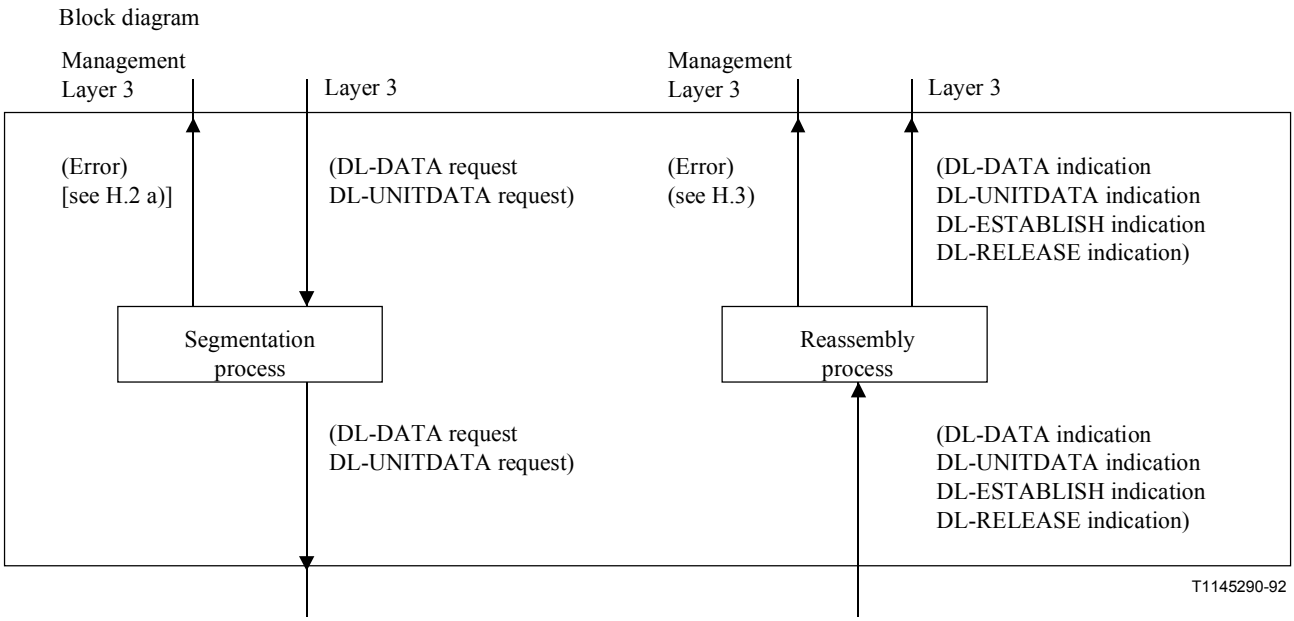


Figure H.3/Q.931 – Segmentation functional interaction diagram

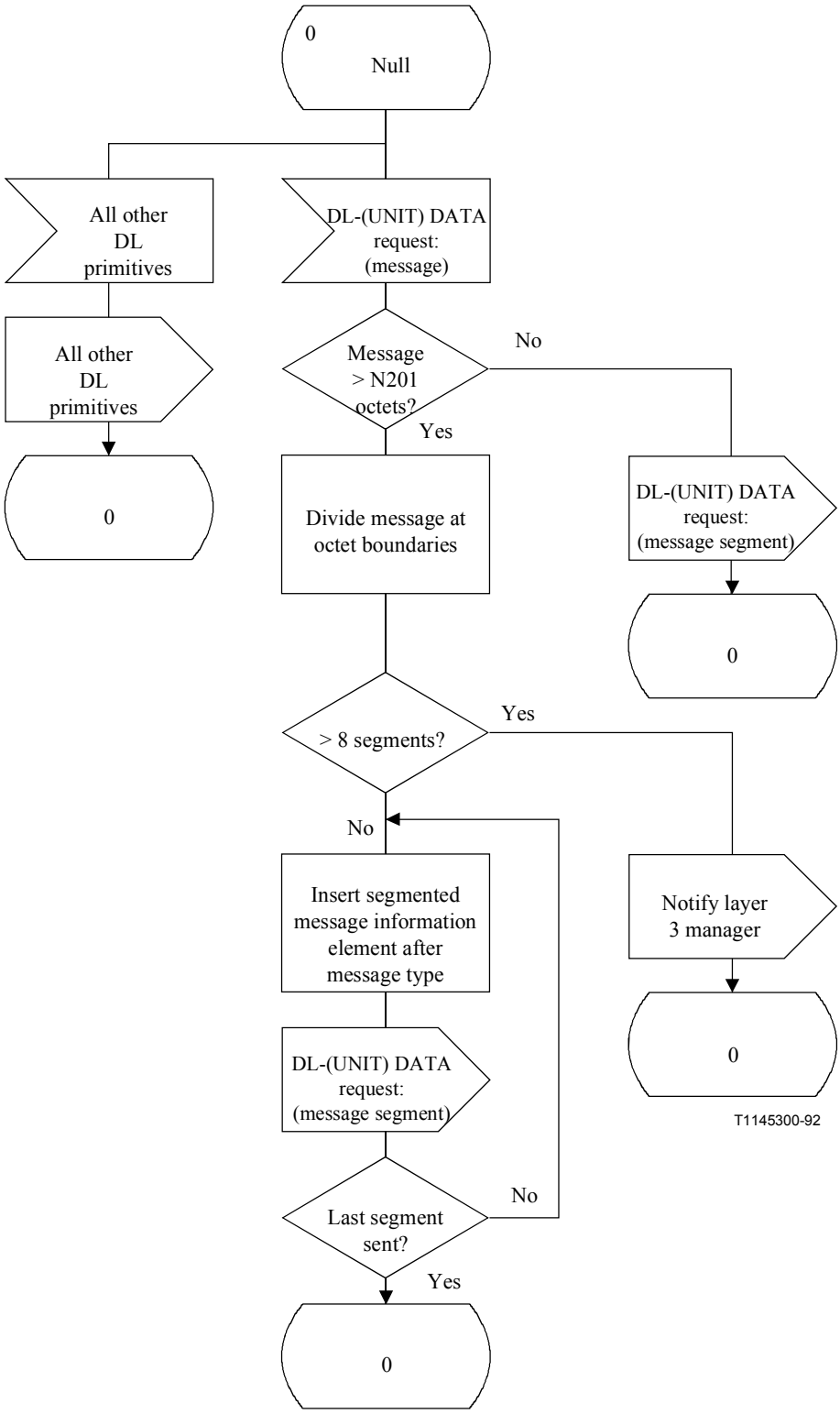


Figure H.4/Q.931 – Message segmenter SDL

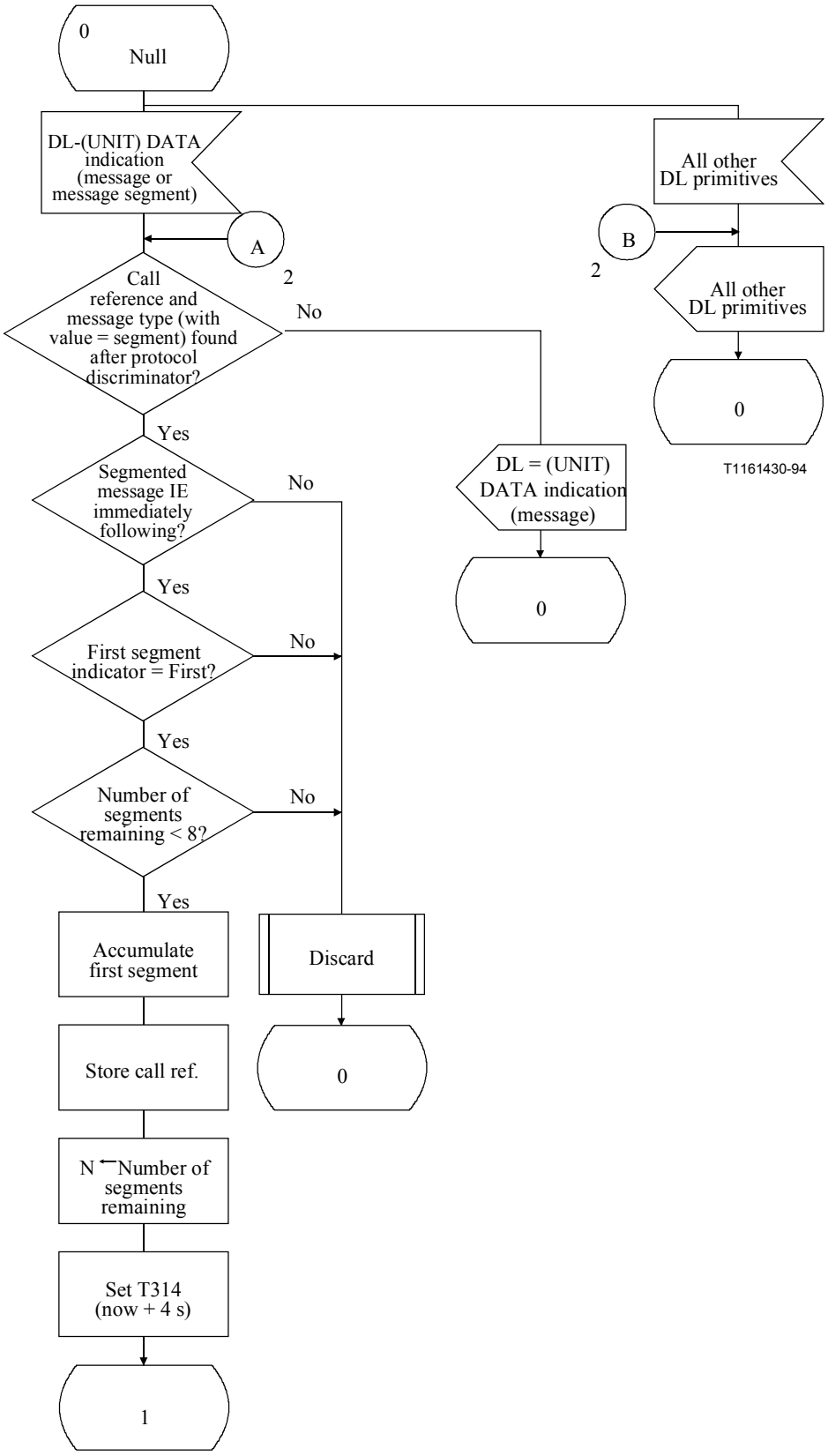


Figure H.5/Q.931 – Message reassembler SDL *(sheet 1 of 3)*

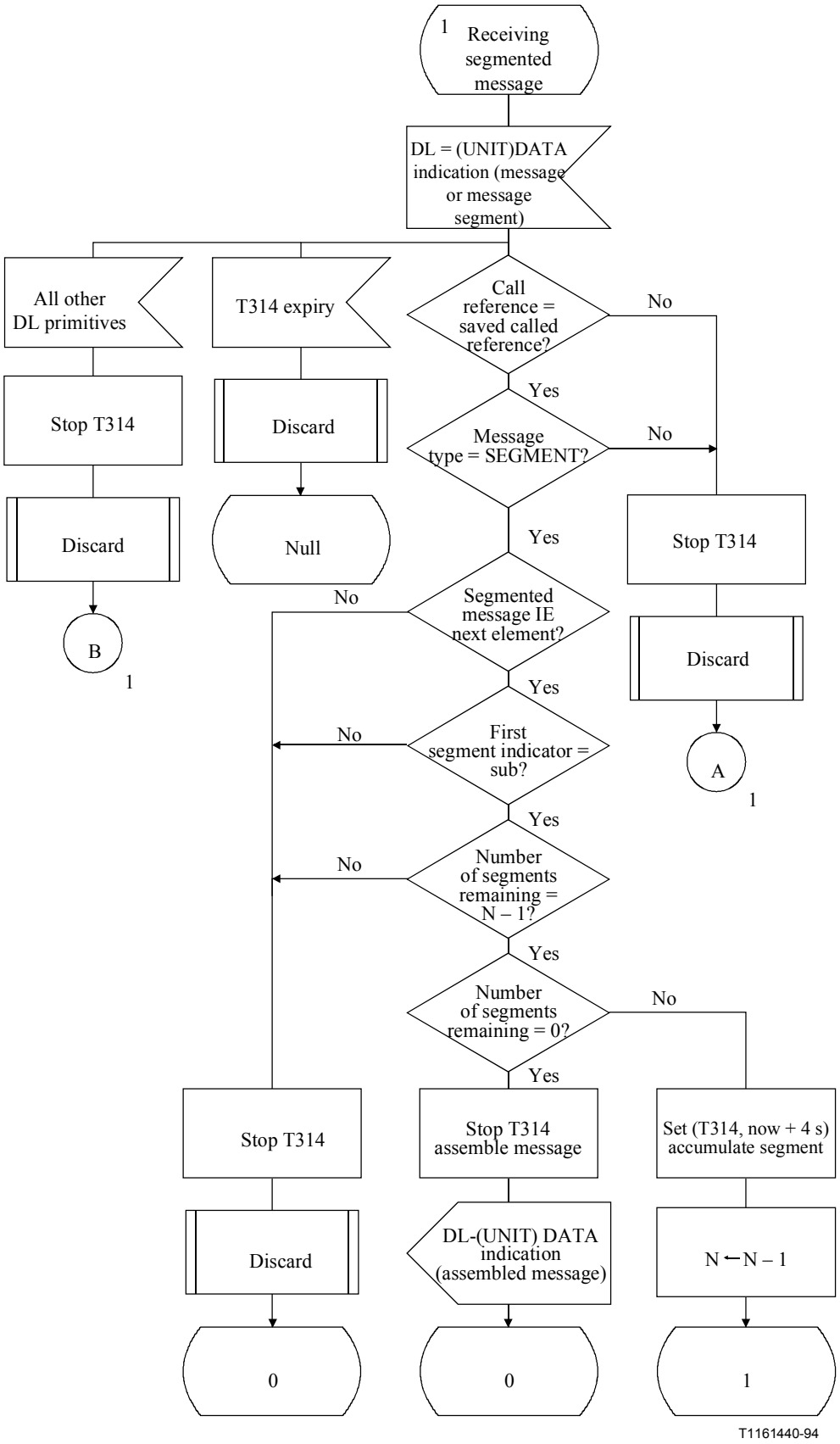


Figure H.5/Q.931 – Message reassembler SDL *(sheet 2 of 3)*

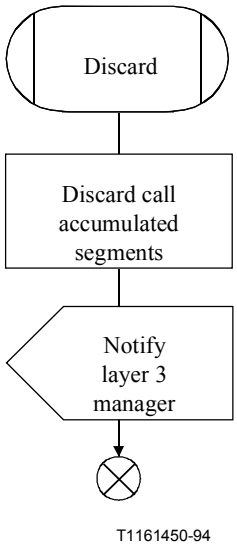


Figure H.5/Q.931 – Message reassembler SDL *(sheet 3 of 3)*

ANNEX I

Low layer information coding principles

I.1 Purpose

This Annex describes principles that shall be used when the calling user specifies information during call set-up regarding low layer capabilities required in the network and by the destination terminal.

NOTE – In this context and throughout this Annex, the term "called user" is the endpoint entity which is explicitly addressed. This may be an addressed Interworking Unit (IWU) (see I.500-series Recommendations and Recommendation X.31 [14] case A).

I.2 Principles

I.2.1 Definitions of types of information

There are three different types of information that the calling ISDN user may specify during call set-up to identify low layer capabilities needed in the network and by the destination terminal:

a) **Type I information** is information about the calling terminal which is only used at the destination end to allow a decision regarding terminal compatibility. An example would be modem type. This information is encoded in octets 5 to 7 of the Low layer compatibility information element.

b) **Type II information** is the selection of bearer capability from the choices of bearer capabilities offered by the network to which the calling user is connected. This type of information is present even if no interworking occurs. An example is Unrestricted Digital Information (UDI). This information is coded in

   i) octets 3 and 4 of the Bearer capability information element when the transfer mode required by the calling user is circuit mode;

   ii) octets 3, 4, 6 and 7 of the Bearer capability information element when the transfer mode required by the calling user is packet mode.

c) **Type III information** is information about the terminal or intended call which is used to decide destination terminal compatibility and possibly to facilitate interworking with other ISDNs or other dedicated networks. An example is A-law encoding. Type III information is encoded in octet 5, 6 and 7 of the Bearer capability information element.

I.2.2 Examination by network

Type I information is user-to-user (i.e. not examined by network) while both types II and III should be available for examination by the destination user and the network. The Low layer compatibility information element is an information element which is not examined by the network while the Bearer capability information element is an information element which is examined by the user and the network.

I.2.3 Location of type I information

Type I information (i.e. terminal information only significant to the called user) shall, when used, be included in the Low layer compatibility information element.

I.2.4 Location of type II and III information

Type II (i.e. bearer selection) information shall be included in the Bearer capability information element. Type III information, when used, is included in the Bearer capability information element. The network may use and modify the information (e.g. to provide interworking). The rationale for the user including some terminal related information in the type III information (interworking related) is shown by the following example.

Normally with UDI, the rate adaption technique chosen is related to the terminal. The specification of a particular rate adaption scheme with a UDI bearer service could allow a compatibility decision by the destination terminal in a purely ISDN situation. However, it could also conceivably be used to allow interworking with a PSTN, assuming that the appropriate functions (i.e. data extraction, modem pool) are available at the interworking unit.

If the rate adaption information is carried in the Low layer compatibility information element, and not in the Bearer capability information element, then interworking by the network providing the bearer capability would not be possible. However, if the rate adaption information is carried in the Bearer capability information element, interworking would be possible.

Hence, there is some terminal related information which may be considered interworking related. The consequence for the calling user of not including such terminal related information in the Bearer capability information element is that the call may not be completed if an interworking situation is encountered.

When type III information is included for any user protocol layer, the following network involvement is permitted within the indicated user protocol:

– **layer 1**: mapping of the user protocol to other protocols, and encapsulation of the user protocol within another protocol;

– **layer 2**: relaying of layer 2 PDUs across different layer 1 environments, and encapsulation of the user protocol within another protocol. Full termination of the user protocol is not provided, and in particular routing or destination identification information within the user protocol is not analysed until the entity addressed by the Called party number information element is reached;

– **layer 3**: relaying of layer 3 PDUs across different layer 2 environments, and encapsulation of the user protocol within another protocol. Full termination of the user protocol is not provided, and in particular routing or destination identification information within the user

protocol is not analysed until the entity addressed by the Called party number information element is reached.

When type II information is included for any user protocol layer, and in addition to the identification of the telecommunication service requested by the user, the following network involvement is permitted within the indicated user protocol:

– **layer 1**: mapping of the user protocol to another protocol, and encapsulation of the user protocol within other protocols;

– **layer 2**: relaying of layer 2 PDUs across different layer 1 environments, and encapsulation of the user protocol within another protocol. Full termination of the user protocol can be provided, and in particular routing or destination identification information within the user protocol is analysed and utilised to reach the destination entity. When the user protocol is terminated, the Called party number information element (if included) is ignored at this point;

– **layer 3**: relaying of layer 3 PDUs across different layer 2 environments, and encapsulation of the user protocol within another protocol. Full termination of the user protocol can be provided, and in particular routing or destination identification information within the user protocol is analysed and utilised to reach the destination entity. When the user protocol is terminated, the Called party number information element (if included) is ignored at this point.

For both type II and type III information, if the network involvement modifies (interworks) the user protocols described by the bearer capability, the bearer capability relayed to the destination is modified appropriately. If no network involvement occurs, the bearer capability relayed to the destination is not modified.

The interworking arrangements with other appropriate bearer capabilities (e.g. packet mode, frame mode) or other networks (e.g. PSTN, B-ISDN) which may be accommodated by some networks are beyond the scope of this Recommendation.

I.2.5 Relationship between Bearer capability and Low layer capability information elements

There shall be no contradiction of information between the Low layer compatibility and the Bearer capability at the originating side. However, as some Bearer capability codepoints may be modified during the transport of the call, this principle implies that there should be minimal duplication of information between Bearer capability information element and Low layer compatibility information element.

NOTE – If as a result of duplication, a contradiction occurs between the Bearer capability information element and the Low layer compatibility information element at the terminating side, the receiving entity shall ignore the conflicting information in the Low layer compatibility information element.

The following example, dealing with the specification of the encoding scheme used by the terminal for the speech or 3.1 kHz audio bearer services, shows the consequences of duplication.

It is expected that some ISDNs will support only A-law and some only μ-law, with conversion provided by the μ-law network. (See Recommendation G.711). If the encoding scheme is specified in both the Bearer capability information element and the Low layer compatibility information element, interworking between two ISDNs might require a change of the user information layer 1 protocol in the Bearer capability information element (e.g. from A-law to μ-law), while the encoding scheme specified in the Low layer compatibility information element would presumably be forwarded to the destination unchanged. Since, to determine compatibility, the destination terminal examines both the Bearer capability information element and the Low layer compatibility information element, it would receive conflicting information regarding the encoding scheme used.

I.3 Information classification

The following are the examples of classifying low layer information currently identified. This information is provided to facilitate understanding of the characteristics of types II and III information.

I.3.1 Examples for speech and 3.1 kHz audio bearer services

a) *Type II information (common to all applications using these bearer services)*

- information transfer capability = speech or 3.1 kHz audio;
- information transfer mode = circuit;
- information transfer rate = 64 kbit/s;
- user information layer 1 protocol = A-/μ-law.

b) *Type III information for interworking with CSPDN (3.1 kHz audio applications are assumed) – Figure I.1*

- user information layer 1 protocol = rate adaption + user rate (see Note);

  NOTE – Only those profiles conforming to ITU-T standardized rate adaption are allowed when only the above information is provided.

c) *Type III information for interworking with PSTN*

i) voice applications – Figure I.2:
   - user information layer 1 protocol = A-/μ-law;

ii) voice band data applications – Figure I.3:
   - user information layer 1 protocol = A-/μ-law.

I.3.2 Examples for 64 kbit/s UDI circuit mode bearer service

a) *Type II information (common)*

- information transfer capability = unrestricted digital information;
- information transfer mode = circuit;
- information transfer rate = 64 kbit/s.

b) *Type III information for interworking with PSPDN (packet applications) – Figure I.4*

- no type III information is required.

c) *Type III information for interworking with PSTN*

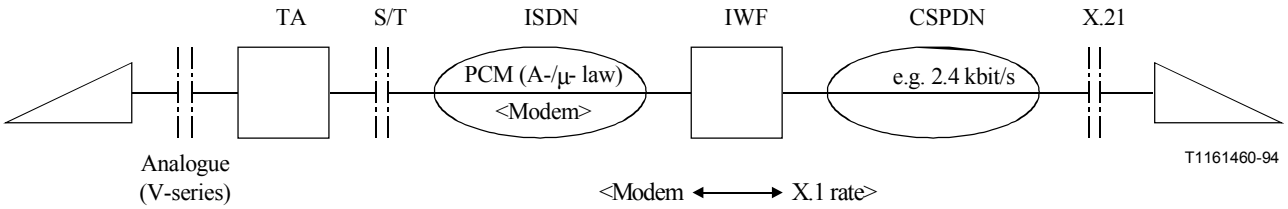
i) voice applications – Figure I.5:
   - no type III information is required;

ii) rate-adapted data applications – Figure I.6:
   - no type III information is required.

d) *Type III information for interworking with PSTN with end-to-end digital connectivity (data applications) – Figure I.7*

- user information layer 1 protocol = rate adaption + user rate (see Note).

  NOTE – The profile described in Recommendation I.463 [52] is allowed.



NOTE – Is user rate sufficient to specify the type of modem at IWF?

Figure I.1/Q.931 – BC = 3.1 kHz audio – Voice → CSPDN

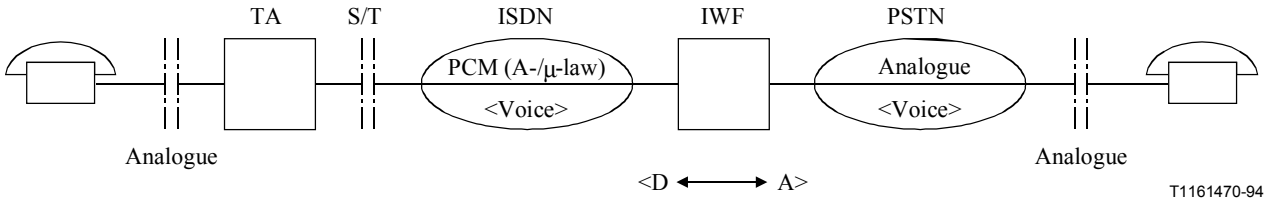


Figure I.2/Q.931 – BC = 3.1 kHz audio – Voice → PSTN

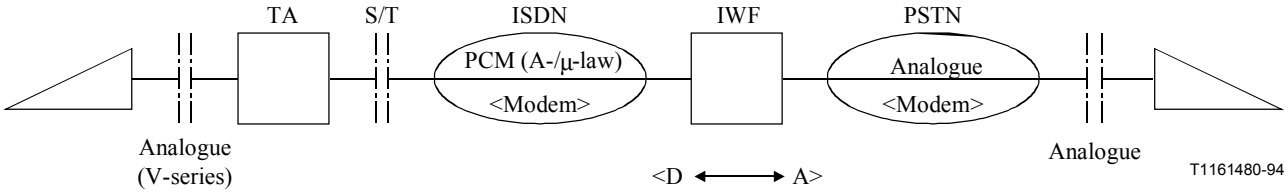


Figure I.3/Q.931 – BC = 3.1 kHz audio – Voice → PSTN

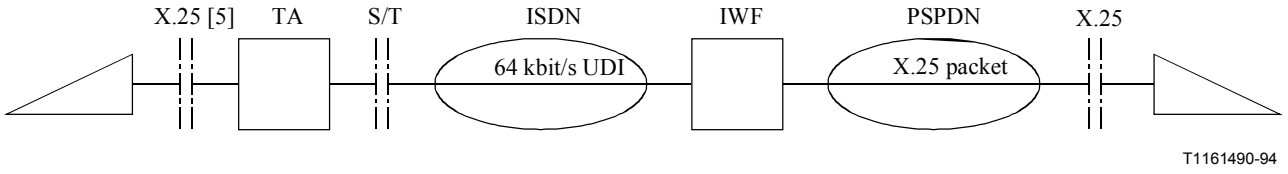


Figure I.4/Q.931 – BC = 64 kbit/s UDI – Packet application → PSPDN

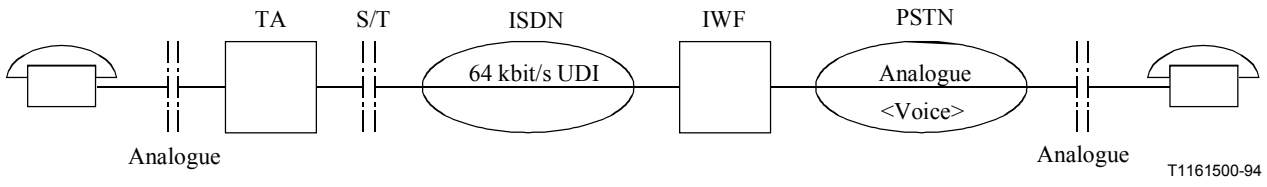


Figure I.5/Q.931 – BC = 64 kbit/s UDI – Voice → PSTN

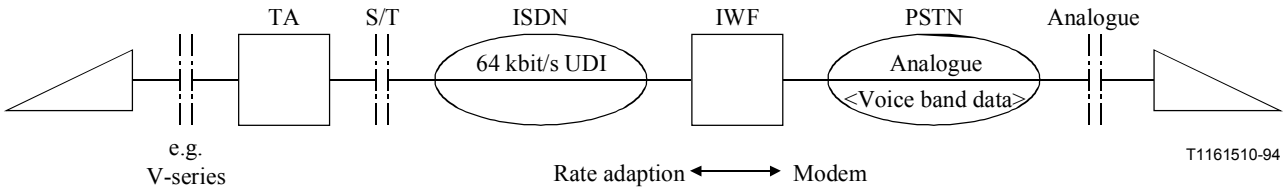


Figure I.6/Q.931 – BC = 64 kbit/s UDI – Rate adapter data → PSTN

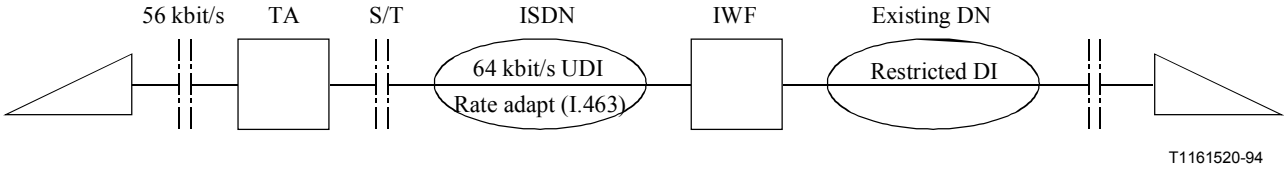


Figure I.7/Q.931 – BC = 64 kbit/s UDI – Existing digital network

I.3.3 Examples for ISDN virtual-circuit bearer service

a) *Type II information (common)*

- information transfer capability = unrestricted digital information;
- information transfer mode = packet;
- information transfer rate = -- -- --;
- user information layer 1 protocol = rate adaption + user rate (see Note 1);
- user information layer 2 protocol = LAPB (see Note 2);
- user information layer 3 protocol = X.25 [5] packet layer protocol (see Note 2).

NOTE 1 – This parameter is included only when user packet information flow is rate adapted. Only those profiles conforming to Recommendation X.31 are allowed when only the above information is provided for layer 1 protocol.

NOTE 2 – Only those profiles conforming to Recommendation X.31 are used. See Figures I.8 to I.10.

b) *Type III information for interworking with PSPDN, CSPDN, PSTN*

- no type III information is necessary.

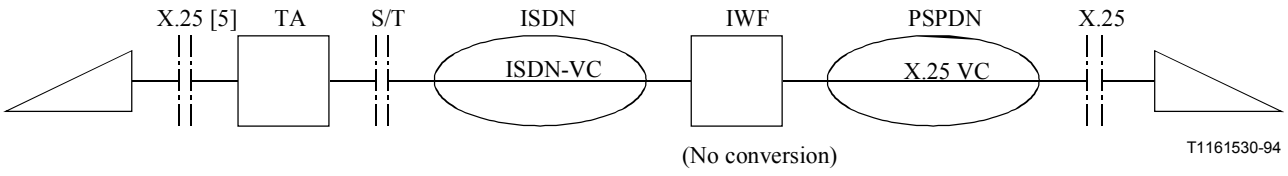


Figure I.8/Q.931 – BC = ISDN Virtual Circuit (VC) → PSPDN

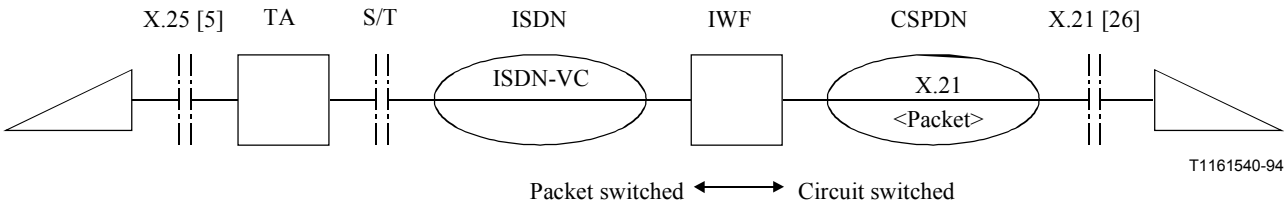


Figure I.9/Q.931 – BC = ISDN Virtual Circuit (VC) → CSPDN

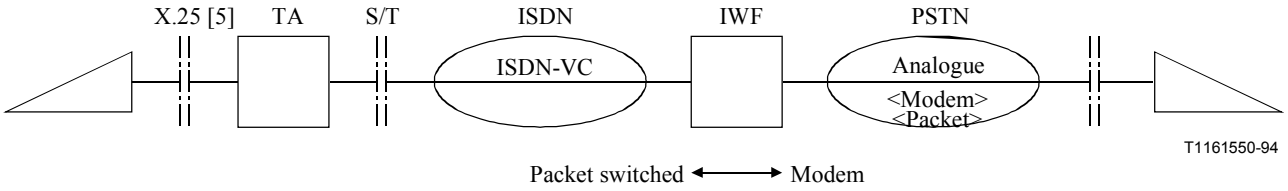


Figure I.10/Q.931 – BC = ISDN Virtual Circuit (VC) → PSTN

I.4 Scenarios outside the scope of ISDN standardization

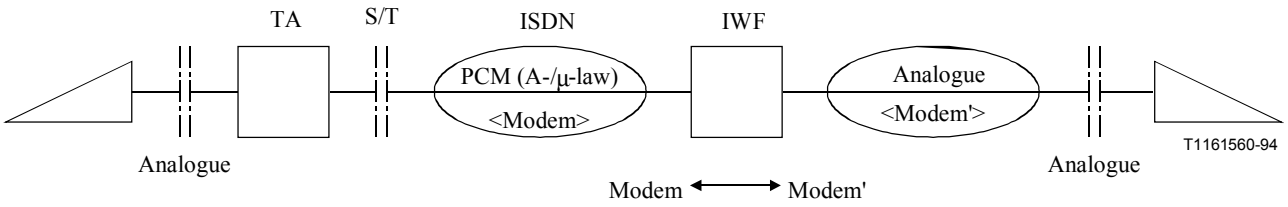
I.4.1 Examples for speech and 3.1 kHz audio bearer services

a) *Type II information (common)*

- information transfer capability = speech or 3.1 kHz audio;
- information transfer mode = circuit;
- information transfer rate = 64 kbit/s;
- user information layer 1 protocol = A-/μ-law.

b) *Type III information for interworking with PSTN – Voice band data applications – Modem type conversion occurs – Figure I.11*

- user information layer 1 protocol = rate adaption + user rate + other attributes (if required).



NOTE – This scenario seems to be a part of PSTN services.

Figure I.11/Q.931 – BC = 3.1 kHz audio – Voice band data → PSTN

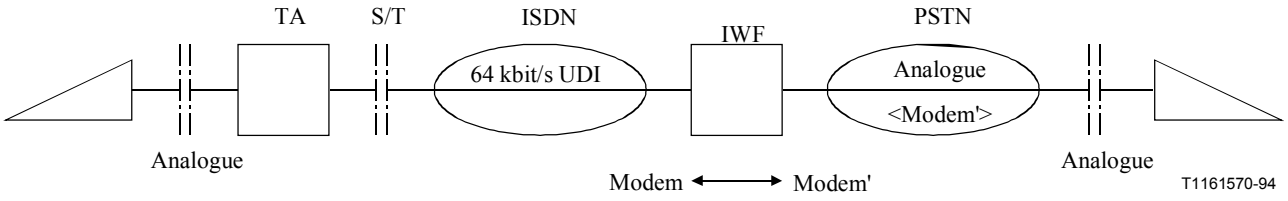
I.4.2 Examples for 64 kbit/s UDI circuit-mode bearer services

a) *Type II information (common)*

- information transfer capability = unrestricted digital information;
- information transfer mode = circuit;
- information transfer rate = 64 kbit/s.

b) *Type III information for interworking with PSTN – Voice band data applications – Figure I.12*

- no type III information is required.



NOTE – This scenario seems to be a combination of interworking with PSTN and a part of PSTN services.

Figure I.12/Q.931 – BC = 64 kbit/s UDI – Voice band data → PSTN

ANNEX J

Low layer compatibility negotiation

This Annex describes an additional low layer compatibility checking procedure that may be applied by the user.

J.1 General

The purpose of the Low layer compatibility information element is to provide a means which should be used for compatibility checking by an addressed entity (e.g. a remote user or an interworking unit or high layer function network node addressed by the calling user). The Low layer compatibility information element is transferred transparently by an ISDN between the call originating entity (e.g. the calling user) and the addressed entity.

The user information protocol fields of the Low layer compatibility information element indicate the low layer attributes at the call originating entity and the addressed entity. This information is not interpreted by the ISDN, and therefore the bearer capability provided by the ISDN is not affected by this information. The call originating entity and the addressed entity may modify the low layer attributes by the negotiation described below if that can be supported by the bearer capability actually provided by the ISDN.

The Low layer compatibility information element is coded according to 4.5.19.

J.2 Low layer capability notification to the called user

When the calling user wishes to notify the called user of its information transfer attributes (type II information – octets 3 and 4) or any low layer protocol (type I information – octets 5 to 7) to be used during the call and not already identified in the Bearer capability information element, then the

calling user shall include a Low layer compatibility information element in the SETUP message; this element is conveyed by the network and delivered to the called user. However, if the network is unable to convey this information element, it shall act as described in 5.8.7.1 (unrecognized information element).

J.3 Low layer compatibility negotiation between users

If the negotiation indicator (see 4.5) of the Low layer compatibility information element included in the SETUP message is set to "Out-band negotiation is possible (octet 3a, bit 7)", then one or more of the low layer protocol attribute(s) may be negotiated. In this case, the called user responding positively to the call may include a Low layer compatibility information element in the CONNECT message. This element will be conveyed transparently by the network and delivered to the calling user in the CONNECT message.

NOTE – Only the low layer protocol attributes may be negotiated and therefore the information transfer attributes (octets 3 to 4), if returned by the called user in the CONNECT message, will be identical to the ones received in the Low layer compatibility information element contained in the SETUP message.

If, for any reason, the network is unable to convey this information element, it shall act as described in 5.8.7.1 (unrecognized information element). Users are advised not to include in the Low layer compatibility information element sent from the called user to the calling user, attributes which would have the same value as the ones contained in the Low layer compatibility information element received from the calling party.

If bearer capability selection applies, and if a Low layer compatibility information element is returned by the called user in the CONNECT message, then the called user shall ensure that the information transfer attributes in that returned Low layer compatibility information element is the same as the information transfer attributes of the selected bearer capability.

J.4 Low layer compatibility negotiation options

The Low layer compatibility information element contains a negotiation indicator which may have one of the following values:

a) *Out-band negotiation not possible (default)* – Then the called user shall not invoke negotiation, according to J.3.

b) *Out-band negotiation possible* – The called user may then invoke low layer compatibility negotiation, as needed, according to J.3.

c) *In-band negotiation possible* – The called user may then invoke low layer compatibility negotiation using the supported in-band negotiation, according to service or application requirements.

d) *Either in-band or out-band negotiation allowed* – The called user may invoke one or the other low layer compatibility negotiation procedures according to its requirements. If the call is end-to-end ISDN, and the out-band low layer compatibility negotiation is supported by both parties, then this method of negotiation is preferred.

J.5 Alternate requested values

If the user wishes to indicate alternative values of low layer compatibility parameters (e.g. alternative protocol suites or protocol parameters), the Low layer compatibility information element is repeated in the SETUP message. Up to four Low layer compatibility information elements may be included in a SETUP message. The first Low layer compatibility information element in the message is preceded by the Repeat indicator information element specifying "priority list for selection". The order of

appearance of the Low layer compatibility information elements indicates the order of preference of end-to-end low layer parameters.

Alternatively, the network may discard the lower priority Low layer compatibility information element(s) depending on the signalling capability of the network.

If the network or called user does not support repeating of the Low layer compatibility information element, and therefore discards the Repeat indicator information element and the subsequent Low layer compatibility information elements, only the first Low layer compatibility information element is used in the negotiation.

The called user indicates a single choice from among the options offered in the SETUP message by including the Low layer compatibility information element in the CONNECT message. Absence of a Low layer compatibility information element in the CONNECT message indicates acceptance of the first Low layer compatibility information element in the SETUP message.

If bearer capability selection applies, then for each individual Low layer compatibility information element included in the SETUP message, the user shall ensure that there is no contradiction between the information contained in that Low layer compatibility information element and the information contained in at least one of the included Bearer capability information elements, either the fallback bearer capability, the preferred bearer capability, or both.

ANNEX K

Procedures for establishment of bearer connection prior to call acceptance

K.1 General

For some applications, it is desirable to allow the completion of the transmission path associated with a bearer service prior to receiving call acceptance. In particular, the completion of the backward direction of the transmission path prior to receipt of a CONNECT message from the called user may be desirable to:

a) allow the called user to provide internally-generated tones and announcements that are sent in-band to the calling user prior to answer by the called user; or

b) avoid speech clipping on connections involving an NT2 where delays may occur in relaying the answer indication within the called user equipment.

The procedures described in this Annex are only applicable to the speech and 3.1 kHz audio bearer services.

NOTE – The definition of necessary mechanisms (if any) with Signalling System No. 7 to avoid any potential undesirable charging implications remains for further study.

K.2 Procedures

As a network option, completion of the transmission path prior to receipt of a call acceptance indication may be provided in one of three ways:

a) on completion of successful channel negotiation at the destination interface; or

b) on receipt of a message containing an indication that in-band information is being provided; or

c) not at all, i.e. this option is not supported by the network.

When criteria a) is used to determine that transmission path should be established, the network shall connect, as a minimum, the backward side of the transmission path upon receipt of either a CALL PROCEEDING message or an ALERTING message containing an acceptable B-channel indication.

When criteria b) is used to establish the transmission path, the network shall connect, as a minimum, the backward side of the transmission path upon receipt of either an ALERTING message or a PROGRESS message containing progress indicator No. 8, *in-band information or appropriate pattern is now available*, or progress indicator No. 1, *call is not end-to-end ISDN; further call progress information may be available in-band*, respectively.

The network providing the early completion of the transmission path in the backward direction may choose to support only one of methods a) or b) above. The network may choose to further restrict which message(s) will result in establishment of the transmission path. These restrictions may be imposed on a per interface basis to provide an administrative means for limiting potential misuse of the early connection capabilities.

ANNEX L

Optional procedures for bearer service change

The procedure for bearer service change may not be provided on all networks. On those networks that support it, a user may use this procedure after making a suitable subscription-time arrangement.

When a bearer service requested in an originator's SETUP message cannot be provided by the network, the network would reject the call or, under some circumstances, the network may change the bearer service and provide bearer service change notification. These procedures are currently applicable only to a change from 64 kbit/s unrestricted to 64 kbit/s restricted, and from 64 kbit/s restricted to 64 kbit/s restricted with rate adaption.

NOTE – During an interim period some networks may only support restricted 64 kbit/s digital information transfer capability, i.e. information transfer capability solely restricted by the requirement that the all-zero octet is not allowed. For interworking, the values given in Appendix I/I.340 should apply. The interworking functions have to be provided in the network restricted capability. The ISDN with 64 kbit/s transfer capabilities will not be offered by this interworking, other than by conveying the appropriate signalling message to or from the ISDN terminal.

Up to two Bearer capability information elements may be present in the SETUP message from the originating user, corresponding to the allowed bearer service modifications given above. The Bearer capability information element shall be immediately preceded by the Repeat indicator information element with the meaning field specifying *Prioritized list for selecting one possibility*. Hence, the order of Bearer capability information elements would indicate order of bearer service preference.

If the SETUP message contains Bearer capability information elements not agreeing with any of the permissible ordered combinations listed above, the network will reject the call attempt.

After sending a CALL PROCEEDING message, when the originating network or terminating premises equipment determines that the preferred bearer service cannot be provided, it sends a NOTIFY message toward the call originator. The NOTIFY message contains a Notification indicator information element with a coding which indicates to the originating party the change in bearer service and also contains a Bearer capability information element specifying the attributes of the new bearer service.

Receipt of the NOTIFY message is not acknowledged. The call originator may allow the call to continue or may initiate call clearing in accordance with clause 5.

ANNEX M

Additional basic call signalling requirements for the support of private network interconnection for Virtual Private Network applications

M.1 Introduction

This Annex covers the application of a public ISDN providing Virtual Private Network (VPN) services to Private Integrated Services Network Exchanges (PINX).

This Annex contains only additional requirements to those in the main body of this Recommendation.

This Annex specifies additional protocol elements and call control procedures for the handling of calls in a VPN context at the T reference point of a public ISDN. The functionality provided by the public network may be:

– the emulation of an Originating PINX;
– the emulation of a Terminating PINX;
– the emulation of a Transit PINX;
– the emulation of a Relay Node;
– the emulation of an Incoming Gateway PINX;
– the emulation of an Outgoing Gateway PINX;
– the emulation of a combination of two or more of the above.

The support of these capabilities is a network option.

The public network can support the coexistence of multiple CNs in parallel, i.e. the resources of the public network are shared by multiple CNs. Each CN should be considered as a separate network.

The minimum requirement of the virtual Transit PINX and the virtual Gateway PINX is to be able to uniquely identify the CN to which a particular attached PINX belongs in order to ensure correct routeing of a particular call.

In addition, a physical PINX may support multiple CNs. Thus the mechanism for identifying a CN needs to be conveyed between a physical PINX and the public network.

This Annex does not cover the requirements for providing VPN services to terminals directly connected to the public network.

The specification included in this Annex does not imply any specific implementation technology or platform.

M.2 Scope

This Annex specifies the extensions required to the basic call control signalling protocol defined in the main body of this Recommendation to support calls within a Corporate telecommunications Network (CN) and to support calls which enter or exit the CN via Gateway PINX functionality performed by the public network. The protocol is applicable at the T reference points to which VPN services are provided. The support of these additional signalling capabilities is a network option. These DSS1 extensions are made available to PINXs on the basis of bilateral agreements at subscription time.

The additional basic call signalling capabilities identified in this Annex are to provide information flows that are functionally identical to the information flows provided by the PSS1 basic call control protocol (as defined by ISO/IEC 11572).

In the context of this Annex, the public network (providing VPN services) can be seen, from the private network perspective, as providing an interconnection between a PINX supporting DSS1 extensions for VPN and another PINX supporting PSS1 information flows. This second PINX may be a physical PINX connected to the public network or may be an emulation of an end PINX function provided by the public network.

An emulation of Originating PINX functionality by the public network should, as a minimum, meet the requirements of an Originating PINX Call Control as defined in ISO/IEC 11572 for the circuit-switched call control and the ISO supplementary services associated with the basic call (CLIP, CLIR, COLP, COLR, SUB). In addition, the Transit counter ANF (defined in ISO/IEC 15056) may be supported by these DSS1 extensions.

An emulation of Terminating PINX functionality by the public network should, as a minimum, meet the requirements of a Terminating PINX Call Control as defined in ISO/IEC 11572 for the circuit-switched call control and the ISO supplementary services associated with the basic call (CLIP, CLIR, COLP, COLR, SUB). In addition, the Transit counter ANF (defined in ISO/IEC 15056) may be supported by these DSS1 extensions.

An emulation of Transit PINX functionality by the public network should, as a minimum, meet the requirements of a Transit PINX Call Control as defined in ISO/IEC 11572 for the circuit-switched call control and the ISO supplementary services associated with the basic call (CLIP, CLIR, COLP, COLR, SUB). In addition, the Transit counter ANF (defined in ISO/IEC 15056) may be supported by these DSS1 extensions.

A Relay Node in a VPN includes the following functions:

– minimal routing capability;

– transparent handling of private networking information (e.g. transit counter).

An emulation of Incoming Gateway PINX functionality by the public network should, as a minimum, meet the requirements of an Incoming Gateway PINX Call Control as defined in ISO/IEC 11572 for the circuit-switched call control and the ISO supplementary services associated with the basic call (CLIP, CLIR, COLP, COLR, SUB). In addition, the Transit counter ANF (defined in ISO/IEC 15056) may be supported by these DSS1 extensions.

An emulation of Outgoing Gateway PINX functionality by the public network should, as a minimum, meet the requirements of an Outgoing Gateway PINX Call Control as defined in ISO/IEC 11572 for the circuit-switched call control and the ISO supplementary services associated with the basic call (CLIP, CLIR, COLP, COLR, SUB). In addition, the Transit counter ANF (defined in ISO/IEC 15056) may be supported by these DSS1 extensions.

The attached PINX acts as the user within the DSS1 protocol defined by this Annex.

M.2.1 Acronyms used in this Annex

ANF Additional Network Feature

BCD Binary Coded Decimal

CLIP Calling Line Identification Presentation

CLIR Calling Line Identification Restriction

CN Corporate telecommunications Network

COLP Connected Line identification Presentation

COLR Connected Line identification Restriction

CPE Customer Premises Equipment

CPN Customer Premises Network

PINX Private Integrated services Network eXchange

PISN Private Integrated Services Network

PSS1 Private Signalling System No. 1

SUB Sub-addressing

VPN Virtual Private Network

M.2.2 References

– ISO/IEC 11572:1997, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Circuit mode bearer services – Inter-exchange signalling procedures and protocol.*

– ISO/IEC 11571:1994, *Information technology – Telecommunications and information exchange between systems – Numbering and sub-addressing in private integrated services networks*.

– ISO/IEC 15056:1997, *Information technology – Telecommunications and information exchange between systems – Private Integrated Services Network – Inter-exchange signalling protocol – Transit counter additional network feature.*

M.2.3 Definitions

M.2.3.1 Virtual Private Network (VPN): Is that part of a CN that provides corporate networking using shared switched network infrastructures. This is split into VPN architecture and VPN services.

The VPN architecture is that part of a CN that provides corporate networking between customer equipment where:

– the shared switch network infrastructure takes the place of the traditional analogue or digital leased lines and the function of the transit node irrespective of the network type, e.g. the Public Switched Telephone Network (PSTN), ISDN or a separate network;

– the customer premises may be served in terms of end node functionality with any combination of PBX, Centrex, Local Area Network (LAN) router, or multiplexer;

– the CN user may also be served by terminal equipment connected to end node functionality residing on customer premises, or provided by public network equipment; and

– the VPN architecture in one network, or multiple networks, comprise a part of the total national or international CN.

VPN services offered by the switched network infrastructure provide:

– VPN end-user services to CN users;

– VPN networking services to support the interconnection of PINXs;

– service interworking functionality;

– inter-VPN services to provide co-operation between the VPN services of two networks; and

– VPN management services to enable service subscribers to control and manage their VPN resources and capabilities.

M.2.3.2 Private Integrated services Network eXchange (PINX): A PISN nodal entity that provides automatic switching and call handling functions used for the provision of telecommunication services. The nodal entity can be implemented by one or more pieces of equipment located on the premises of the private network administrator or by equipment co-located with, or physically part of, a public network.

NOTE – If applicable, a PINX provides to users of the same and/or other private integrated services network exchanges:

– telecommunication services within its own area; and/or

– telecommunication services from the public ISDN; and/or

– telecommunication services from other public or private networks; and/or

– within the context of a PISN, telecommunication services from other PINXs.

M.2.3.3 end PINX functionality: Within the context of a call, the functionality of a PINX required to provide attachment and servicing of terminals.

M.2.3.4 originating PINX functionality: End PINX functionality providing support of the calling user.

M.2.3.5 terminating PINX functionality: End PINX functionality providing support of the called user.

M.2.3.6 transit PINX functionality: Within the context of a call, the functionality of a PINX, emulated in the public network, required to interconnect a pair of PINXs.

M.2.3.7 gateway PINX functionality: Within the context of a call, the functionality of a PINX required to interconnect End PINXs or Transit PINXs with nodes of other public or private networks, or with nodes supporting different signalling capabilities.

M.2.3.8 outgoing gateway PINX functionality: Gateway PINX functionality providing support of calls from the Corporate Network to other networks.

M.2.3.9 incoming gateway PINX functionality: Gateway PINX functionality providing support of calls incoming to the Corporate Network.

M.2.3.10 relay node functionality: Within the context of a call, the functionality that identifies calls in a VPN context, and relays such calls to designated PINX functionality emulated by public network equipment, or to a designated physical PINX. This may be via other Relay Nodes. Relay Node functionality includes transparent handling of private networking information (e.g. transit counter).

M.2.3.11 preceding PINX: In the context of a call, an entity with PINX functionality located in the direction towards the calling user.

M.2.3.12 subsequent PINX: In the context of a call, an entity with PINX functionality located in the direction towards the called user.

M.2.3.13 Corporate telecommunications Network (CN): Consists of sets of equipment [Customer Premises Equipment (CPE) and/or Customer Premises Network (CPN) and/or public network providing VPN services] which are located at geographically dispersed locations and are interconnected to provide networking services to a defined group of users.

NOTE 1 – The ownership of the equipment is not relevant to this definition.

NOTE 2 – Even equipment which is not geographically dispersed (e.g. a single PINX or a Centrex providing service to users at a single location) may form a CN.

M.3 Basic call states

The call states apply unchanged, as defined in 2.1/Q.931 and 2.4/Q.931.

M.4 Additional messages and content

No additional messages are defined. However, the content of some messages has additional requirements.

M.4.1 SETUP message

The Called party number information element is mandatory in both the user-network and the network-user directions.

The Transit counter information element may be included in the SETUP message, for use in both user-to-network and network-to-user directions.

The inclusion of the VPN indicator information element is mandatory in both the user-network and the network-user directions.

M.4.2 CONNECT message

The Connected number information element and the Connected subaddress information element may be included in the CONNECT message for use in both user-to-network and network-to-user directions.

M.5 Additional information elements and coding

M.5.1 Called party number

Subclause 4.5.8/Q.931 shall apply with the exception that Table 4-9/Q.931 is replaced by Table M.1:

Table M.1/Q.931

Numbering plan identification (octet 3)

Bits 4 3 2 1	
0 0 0 0	Unknown (Note 1)
0 0 0 1	ISDN/telephony numbering plan (Recommendation E.164)
1 0 0 1	Private numbering plan (ISO/IEC 11571)

All other values are reserved.

NOTE 1 – The numbering plan "unknown" is used when the user or network has no knowledge of the numbering plan. In this case, the number digits field is organized according to the network dialling plan, e.g. prefix or escape digits might be present.

Type of number (octet 3) when Numbering Plan identification is ISDN/telephony numbering plan (Recommendation E.164) (Note 2)

Bits 7 6 5	
0 0 0	Unknown (Note 3)
0 0 1	International number (Note 4)
0 1 0	National number (Note 4)
1 0 0	Subscriber number (Note 4)

All others values are reserved.

NOTE 2 – For the definition of international, national and subscriber number, see Recommendation I.330.

NOTE 3 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

NOTE 4 – Prefix or escape digits shall not be included.

Table M.1/Q.931 *(concluded)*

Type of number (octet 3) when Numbering Plan identification is Unknown

Bits
7 6 5
0 0 0 Unknown (Note 5)

All others values are reserved.

NOTE 5 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

Type of number (octet 3) when Numbering Plan identification is Private numbering plan (Note 6)

Bits
7 6 5
0 0 0 Unknown (Note 7)
0 0 1 Level 2 Regional Number
0 1 0 Level 1 Regional Number
0 1 1 PISN specific number
1 0 0 Level 0 Regional Number

All others values are reserved.

NOTE 6 – For the definition of Level 2 Regional Number, Level 1 Regional Number, Level 0 Regional Number and PISN specific number, see ISO/IEC 11571.

NOTE 7 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. Level 2, Level 1, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

Number digits (octets 4, etc.)

This field is coded with IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

M.5.2 Calling party number

Subclause 4.5.10/Q.931 shall apply with the exception that Table 4.11/Q.931 is replaced by Table M.2:

Table M.2/Q.931

Numbering plan identification (octet 3)

Bits
4 3 2 1
0 0 0 0 Unknown (Note 1)
0 0 0 1 ISDN/telephony numbering plan (Recommendation E.164)
1 0 0 1 Private numbering plan (ISO/IEC 11571)

All other values are reserved.

NOTE 1 – The numbering plan "unknown" is used when the user or network has no knowledge of the numbering plan. In this case, the number digits field is organized according to the network dialling plan, e.g. prefix or escape digits might be present.

Table M.2/Q.931 *(continued)*

Type of number (octet 3) when Numbering Plan identification is ISDN/telephony numbering plan (Recommendation E.164) (Note 2)

Bits 7 6 5	
0 0 0	Unknown (Note 3)
0 0 1	International number (Note 4)
0 1 0	National number (Note 4)
1 0 0	Subscriber number (Note 4)

All others values are reserved.

NOTE 2 – For the definition of international, national and subscriber number, see Recommendation I.330.

NOTE 3 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

NOTE 4 – Prefix or escape digits shall not be included.

Type of number (octet 3) when Numbering Plan identification is Unknown

Bits 7 6 5	
0 0 0	Unknown (Note 5)

All others values are reserved.

NOTE 5 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. international number, national number, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

Type of number (octet 3) when Numbering Plan identification is Private numbering plan (Note 6)

Bits 7 6 5	
0 0 0	Unknown (Note 7)
0 0 1	Level 2 Regional Number
0 1 0	Level 1 Regional Number
0 1 1	PISN specific number
1 0 0	Level 0 Regional Number

All others values are reserved.

NOTE 6 – For the definition of Level 2 Regional Number, Level 1 Regional Number, Level 0 Regional Number and PISN specific number, see ISO/IEC 11571.

NOTE 7 – The type of number "unknown" is used when the user or the network has no knowledge of the type of number, e.g. Level 2, Level 1, etc. In this case, the number digits field is organized according to the network dialling plan; e.g. prefix or escape digits might be present.

Presentation indicator (octet 3a)

Bits 7 6	
0 0	Presentation allowed
0 1	Presentation restricted
1 0	Number not available due to interworking
1 1	Reserved

Table M.2/Q.931 *(concluded)*

Screening indicator (octet 3a)

Bits

2 1

0 0 User-provided, not screened
0 1 User-provided, verified and passed
1 0 Reserved
1 1 Network provided

Number digits (octets 4, etc.)

This field is coded with IA5 characters, according to the formats specified in the appropriate numbering/dialling plan.

M.5.3 Connected number

The coding of the Connected number information element is defined in 5.4/Q.951, with the exception that the content of this information element is coded as defined in M.5.2.

M.5.4 Connected subaddress

The coding of the Connected subaddress information element is defined in 5.4/Q.951.

M.5.5 Progress indicator

The following additional progress description values are defined in the ISO/IEC coding standard:

Bits 7 6 5 4 3 2 1	No.	
0 0 1 0 0 0 0	16	Interworking with a public network.
0 0 1 0 0 0 1	17	Interworking with a network unable to supply a release signal.
0 0 1 0 0 1 0	18	Interworking with a network unable to supply a release signal before answer.
0 0 1 0 0 1 1	19	Interworking with a network unable to supply a release signal after answer.

M.5.6 Transit counter

The Transit counter information element may, optionally, be included in the SETUP message to indicate the number of private network transit exchanges which intervene in the requested connection. The Transit counter information element has a maximum length of 3 octets.

The Transit counter information element is defined in codeset 4.

8	7	6	5	4	3	2	1	Octet
Transit counter information element identifier								
0	0	1	1	0	0	0	1	1
Length of Transit counter								2
ext.	Reserved		Transit count (binary value)					3
1	0	0						

M.5.7 VPN indicator

The VPN indicator information element shall be included in the SETUP message to indicate that the call is in VPN context. The VPN indicator information element may, optionally, include a CN identifier to distinguish between CNs in the VPN. The VPN indicator information element has a maximum length of 15 octets.

The VPN indicator information element is defined in codeset 0.

8	7	6	5	4	3	2	1	Octet
VPN indicator information element identifier								
0	0	0	0	0	1	0	1	1
Length of VPN indicator								2
1	Spare				CN indicator			3
CN identifier								3.1* ... 3.12*

CN indicator (octet 3)

Bits

3 2 1

0 0 0 No indication (Note 1)

0 0 1 Network specific (Note 2)

0 1 0 Global (Note 3)

All others values are reserved.

NOTE 1 – When the CN indicator "no indication" is used, the call belongs to the assigned default CN.

NOTE 2 – When the CN indicator "network specific" is used, the CN identifier is contained in the following octets.

NOTE 3 – When the CN indicator "global" is used, the CN identifier in the following octets contains a globally unique value.

CN identifier (octets 3.1 to 3.12)

The CN identifier indicated "Network specific" is a network provider matter.

When the CN indicator is set to "global", the CN identifier contains the binary representation of the CN identifier. The CN identifier starts with the BCD (Binary Coded Decimal) representation of the E.164 country code digits of the country where the CN was initially assigned. The remainder of the CN identifier is country specific.

M.6 Additional basic call control procedures

M.6.1 Distinction between public network and VPN context

If an entity sends a message that establishes a call reference in a VPN context, that entity shall include a VPN indicator information element in this message.

NOTE – As a network option, the Network-specific facilities information element may be used instead of the VPN indicator information element (see Appendix I to Annex M).

If an entity receives a message that establishes a call reference, and this message does not contain a VPN indicator information element, then the procedures for signalling in a public network context for all messages that use this call reference shall apply.

If an entity receives a message that establishes a call reference, and this message contains a VPN indicator information element, then the procedures for signalling in a VPN context for all messages that use this call reference shall apply.

M.6.2 Procedures applicable for signalling in a public network

For a call which is not identified as a call in a VPN context (see M.4.1), clause 5 shall apply.

M.6.3 Procedures applicable for signalling in a VPN context

For a call which is identified as a call in a VPN context (see M.4.1), clause 5 shall apply with the additions described in M.6.3.1 and M.6.3.2.

M.6.3.1 Call establishment from a physical PINX

M.6.3.1.1 Call Request

The physical PINX at the originating interface shall include the VPN indicator information element in the SETUP message.

If the VPN indicator information element does not contain a CN identifier and a CN identifier is registered as a default for the access, then the default CN identifier shall be used.

If the VPN indicator information element does not contain a CN identifier and there is no CN identifier registered as a default for the access, then the call shall be rejected with cause No. 50, *Requested facility not subscribed*.

If the VPN indicator information element contains a CN indicator value and/or a CN identifier which is not associated with the access, then the call shall be rejected with cause No. 50, *Requested facility not subscribed*.

The physical PINX at the originating interface shall include the Called party number information element in the SETUP message.

If received from the physical PINX at the originating interface, the Calling party number information element and the Calling party subaddress information element shall be handled as follows:

– a Transit PINX shall transfer the information elements to the subsequent PINX regardless of any supplementary service subscription information;

– a Relay Node shall transfer the information elements to the subsequent PINX regardless of any supplementary service subscription information;

– an Outgoing Gateway PINX may transfer the information to the other network;

NOTE – The handling of numbers, e.g. translations, presentation indications, is outside the scope of this Annex.

– a Terminating PINX may transfer the information to the called user, depending on any restrictions (e.g. interface type or service).

The physical PINX at the originating interface may include the Transit counter information element in the SETUP message. Whilst the handling by the public network is outside the scope of this Recommendation, it shall be transferred as follows:

– a Transit PINX shall transfer the information element to the subsequent PINX;

– a Relay Node shall transfer the information element to the subsequent PINX;

– an Outgoing Gateway PINX may transfer the information to the other network.

M.6.3.1.2 Notification of interworking at the interface between a physical PINX and the public network

When the public network receives a specific private network Progress description value from the subsequent PINX, it shall transfer it to the physical PINX at the originating interface, without acting upon it.

Outgoing Gateway PINX functionality shall provide Progress indicator information elements as specified below and this information shall be transferred to the physical PINX. A Progress indicator information element shall be transmitted in a PROGRESS message, an ALERTING message or a

CONNECT message as soon as the information becomes available, subject to a SETUP ACKNOWLEDGE or CALL PROCEEDING message having already been sent. A PROGRESS message shall be used unless an ALERTING or CONNECT message is to be sent at the time. All appropriate interworking indications shall be transmitted by the Outgoing Gateway PINX.

If one of the following progress descriptions has been received on a call exiting the CN, that information shall be passed on:

– No. 1 – *Call is not end-to-end ISDN; further call progress information may be available in-band*.

– No. 2 – *Destination address is non-ISDN*.

– No. 4 – *Call has returned to the ISDN*.

– No. 8 – *In-band information or appropriate pattern is now available*.

If the call is to enter another network (public or private) which is not ISDN, a Progress indicator information element may be sent containing progress description No. 1, *Call is not end-to-end ISDN; further call progress information may be available in-band*.

The physical PINX at the originating interface may, optionally, include any of the specific private network Progress description values in the SETUP message, to enable indication of particular situations at the originating side to the subsequent PINX. The public network shall transfer it to the subsequent PINX.

Up to three Progress indicator information elements may be included in a SETUP, ALERTING, PROGRESS and CONNECT message.

M.6.3.1.3 In-band information provided to the physical PINX at the originating interface

Any progress indications shall be conveyed towards the physical PINX at the originating interface.

On receipt of the Progress description No. 1 or No. 8 in the PROGRESS or ALERTING message, the physical PINX at the originating interface shall switch through in the backward direction to the allocated B-channel in order to enable transfer of in-band tones/information, and stop timer T310, if running.

M.6.3.1.4 Call confirmation

The public network shall include the Connected number information element and the Connected subaddress information element in the CONNECT message as follows:

– if received from a subsequent PINX, a Transit PINX shall transfer the information elements to the physical PINX at the originating interface regardless of any supplementary service subscription information;

– if received from a subsequent PINX, a Relay Node shall transfer the information elements to the physical PINX at the originating interface regardless of any supplementary service subscription information;

– an Outgoing Gateway PINX may transfer the information from the other network;

  NOTE – The handling of numbers, e.g. translations, presentation indications, is outside the scope of this Annex.

– a Terminating PINX shall provide the Connected number information element to the physical PINX regardless of any possible supplementary service subscription information. Furthermore, a Terminating PINX shall transfer the Connected subaddress information element if received from the connected user regardless of any possible service subscription information.

M.6.3.2 Call establishment towards a physical PINX

M.6.3.2.1 Incoming call

For calls in a VPN context, the public network shall include the VPN indicator information element in the SETUP message.

Incoming Gateway PINX functionality and Originating PINX functionality shall identify the call as a call in a VPN context.

The public network shall include the Calling party number information element and the Calling party subaddress information element in the SETUP message as follows:

– if received from a preceding PINX, a Transit PINX shall transfer the information elements to the physical PINX at the destination interface regardless of any supplementary service subscription information;

– if received from a preceding PINX, a Relay Node shall transfer the information elements to the physical PINX at the destination interface regardless of any possible supplementary service subscription information;

– an Incoming Gateway PINX may transfer the information from the other network;

NOTE – The handling of numbers, e.g. translations, presentation indications, is outside the scope of this Annex.

– an Originating PINX shall provide the Calling party number information element to the physical PINX at the destination interface regardless of any possible supplementary service subscription information. Furthermore, an Originating PINX shall transfer the Calling party subaddress information element if received from the calling user regardless of any possible supplementary service subscription information.

The public network shall include the Transit counter information element in the SETUP message if received from the preceding PINX.

M.6.3.2.2 Notification of interworking at the interface between a physical PINX and the public network

The PINX at the destination interface may, optionally, include any of the specific private network Progress description values in the ALERTING, PROGRESS or CONNECT message returned to the public network, to enable notification of particular situations at the destination side. The public network shall then transfer the information as follows:

– a Transit PINX shall transfer the information elements to the preceding PINX;

– a Relay Node shall transfer the information elements to the preceding PINX;

– an Incoming Gateway PINX may transfer the information to the other network;

– an Originating PINX may convey the information to the calling user.

Incoming Gateway PINX functionality shall provide Progress indicator information elements in the SETUP message as specified below, and this information shall be transferred to the physical PINX. If none of the specified conditions apply, no Progress indicator information element shall be included.

If one of the following progress descriptions has been received on a call entering the CN, that information shall be passed on:

– No. 1 – *Call is not end-to-end ISDN, further call progress information may be available in-band*.

– No. 3 – *Origination address is non-ISDN*.

If the call has entered the Corporate Network from a network (public or private) which is not ISDN, a Progress indicator information element may be sent containing progress description No. 1, *Call is not end-to-end ISDN; further call progress information may be available in-band*.

When the public network receives a specific private network Progress description value from the preceding PINX, it shall transfer it to the physical PINX at the destination interface, without acting upon it.

Up to three Progress indicator information elements may be included in a SETUP, ALERTING, PROGRESS and CONNECT message.

M.6.3.2.3 In-band information provided by the physical PINX at the destination interface

During call establishment, after the first message received in response to the SETUP message, on receipt of a Progress indicator information element with a Progress description No. 1 or No. 8 in the PROGRESS or ALERTING message, the public network shall switch through in the backward direction to the allocated B channel in order to enable in-band tones/information provided from the physical PINX at the destination interface to the calling user, stop timer T310, if running, and if progress description No. 1 or No. 8 was received in the PROGRESS message while T310 was running, restart timer T310.

The public network shall transfer the Progress indicator information element towards the preceding PINX.

M.6.3.2.4 Call confirmation

The physical PINX at the destination interface may include the Connected number and the Connected subaddress information elements in the CONNECT message.

The Connected number information element and the Connected subaddress information element, when received from the physical PINX at the destination interface in the CONNECT message, shall be transferred by the public network as follows:

– a Transit PINX shall transfer the information elements towards the preceding PINX, regardless of any supplementary service subscription information;

– a Relay Node shall transfer the information elements to the preceding PINX, regardless of any possible supplementary service subscription information;

– an Incoming Gateway PINX may transfer the information to the other network;

NOTE – The handling of numbers, e.g. translations, presentation indications, is outside the scope of this Annex.

– an Originating PINX may transfer the information to the calling user, depending on any restrictions.

M.7 System parameters

T310: the value of this timer when started or restarted upon receipt of progress description No. 1 or No. 8 has a standard default value of 2 minutes (can have different values in a range from 1 to 7 minutes).

APPENDIX M.I

(to Annex M)

Discrimination of calls in a VPN context by means of the Network-specific facilities information element

As a network option, subject to user and network service provider bilateral agreement, the Network-specific facilities information element may be used to discriminate calls in a VPN context. The coding of this information element is shown in Figure 4-27.

Some networks are known to have assigned the following network-specific coding for the field "Network-specific facility specification".

8	7	6	5	4	3	2	1	Octet
ext. 1	Facility coding value							4
Spare 0	Service parameters (IA5 characters)							5, etc.

Figure M.I.1/Q.931 – Example of Network-specific facilities information element coding to discriminate calls in a VPN context

- *Facility coding value (octet 4)*

  Bits
  7 6 5 4 3 2 1

  1 1 1 1 0 0 1 VPN service selection

- *Service parameters (octet 5)*

  Service parameters (e.g. CN identifier) may be encoded in octet(s) 5 according to the network service provider specifications.

ANNEX N

Flexible channel selection

It is a network option to support the procedures in this Annex.

When a preferred bearer requires a larger bandwidth than an allowed alternate fallback, e.g. 6×64 kbit/s to 64 kbit/s, multiple channel identifications may be present in the SETUP message. In this case, one Channel identification information element will be included for each Bearer capability information element in the SETUP message. If the bandwidth required for two of the bearers is the same, then the same channel shall be indicated for the two bearers. Channel selection procedures for each bearer will follow 5.1.2 and 5.2.3 independently, except that the selection of the same channel for more than one of the bearer alternatives will be allowed and will not constitute a conflict. The bearer capability selected for the call will determine the final selection of the channel for the call.

Procedures for early cut through, the use of the Repeat indicator information element for the channel identifications and the confirmed release of unused channels are for further study.

APPENDIX I

Definition of causes values

Table I.2 indicates the usage of cause values within this Recommendation. Other usage may be provided within other Recommendations, e.g. Q.700-series and Q.699. Other causes may also be used by Q.931 entities where this is not precluded by the procedures defined elsewhere in this Recommendation.

Table I.1 defines the key for the location of generation in Table I.2. For more precise usage of the location codes in the cause information element, see Recommendation Q.850.

Table I.1/Q.931 – Key to the location in Table I.2

LU	Local User
LN	Local Network
TN	Transit Network
RN	Remote Network
RU	Remote User
LPE	Local Peer Entity (for symmetrical operation, see Annex D)
The following abbreviations to message types are used in Table I.2	
CON CON	CONGESTION CONTROL
DISC	DISCONNECT
REL	RELEASE
REL COM	RELEASE COMPLETE
RES REJ	RESUME REJECT
STAT	STATUS
SUSP REJ	SUSPEND REJECT

Table I.2/Q.931 – Usage of cause values

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
1	000	0001	Unassigned (unallocated) number	Condition	5.1.4	LN		REL COM DISC
					5.2.4	RU	REL COM DISC	
2	000	0010	No route to specified transit network	Transit network identity/network specific facilities info. Elements	C.2	TN		DISC
					E.3	LN		REL COM
3	000	0011	No route to destination	Condition	5.1.4	LN		DISC REL COM
					5.2.4	RU	REL COM DISC	DISC
6	000	0110	Channel unacceptable	–	5.2.3.1 c) 5.3.2 d) 6.2.2.3.1	LN		REL
7	000	0111	Call awarded and being delivered in an established channel	–	6.2.2.3.1	LN		REL
16	001	0000	Normal call clearing	Condition		RU	DISC	DISC

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
17	001	0001	User busy	–	5.2.5.1 5.2.5.4 b)	RU	REL COM	DISC
					No procedure	RN		DISC
18	001	0010	No user responding	–	5.2.5.3	RN		DISC
19	001	0011	User alerting, no answer	–	5.2.5.3	RN		DISC
21	001	0101	Call rejected	Condition: user supplied diagnostic	5.2.5.1 5.2.5.4 b)	RU	REL COM	DISC
22	001	0110	Number changed	New destination number	5.1.4	LN		DISC REL COM
					5.2.4	RU	REL COM DISC	DISC
26	001	1010	Non-selected user clearing	–	5.3.2 b) 6.2.2.3.1	LN		REL
27	001	1011	Destination out of order	–	5.8.9	RN		DISC
28	001	1100	Invalid number format (incomplete number)	–		LN		DISC REL COM
					5.2.4	RU	DISC REL COM	DISC

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
					5.1.5.2	LN		DISC
					5.2.4	RN		DISC
					5.1.4	LN		DISC REL COM
29	011	1101	Facility rejected	Facility identification	No procedure in Q.931	LN		REL COM DISC
						RN		DISC
						RU	REL COM DISC	
30	001	1110	Response to STATUS ENQUIRY	–	5.8.10	LU, LN		STAT
31	001	1111	Normal, unspecified	–	5.8.4	RN		REL COM DISC
34	010	0010	No circuit/channel available	–	5.1.1 5.1.2 5.1.5.1 5.1.5.2	LN		REL COM
					5.2.3.1 b) 5.2.3.1 e) 5.2.3.2 6.2.2.3.1	RU	REL COM	DISC
					C.2	LN	REL COM DISC	REL COM DISC

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
					C.2	TN		DISC
					D.1.1 e) D.3 b)	LPE		REL COM
38	010	0110	Network out of order	–	No procedure			
41	010	1001	Temporary failure	–	5.8.8	LU, LN		DISC
					5.8.10	LN, RU, RN	DISC	DISC
42	010	1010	Switching equipment congestion	–	No procedure			REL REL COM
43	010	1011	Access information discarded	Discarded into element identifier(s)	7.1.5.7	RU, LN, RU		CON CON
					5.8.7.2	LN, LU		STAT
44	010	1100	Requested circuit/channel not available	–	5.1.2 5.1.5.1 5.1.5.2	LN		REL COM
					5.2.3.1 e) 5.2.3.2 6.2.2.3.1	RU	REL COM	DISC
					D.1.1 e)			REL COM
47	010	1111	Resource unavailable, unspecified	–	No procedure			

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
57	011	1001	Bearer capability not authorized	Attributes of bearer capability	5.1.5.2	LN		DISC REL COM
					7.2	LN		REL REL COM
58	011	1010	Bearer capability not presently available	Attributes of bearer capability	5.1.5.2	LN		DISC REL COM
					7.2	LN		REL REL COM
63	011	1111	Service or option not available, unspecified	–	5.1.5.2	LN		DISC REL COM
65	100	0001	Bearer capability not implemented	Attributes of bearer capability	5.1.5.2	LN		DISC REL COM
					6.1	LN		REL COM
66	100	0010	Channel type not implemented	Channel type	No procedure			
69	100	0101	Requested facility not implemented	Facility identification	7.1.3.6	RU	DISC REL COM	DISC
					7.1.4.3 7.1.5.3	RN		REL DISC
					7.3	LN		REL REL COM

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
70	100	0110	Only restricted digital information bearer capability is available	–	No procedure (network dependent option)			
79	100	1111	Service or option not implemented, unspecified					
81	101	0001	Invalid call reference value	–	5.8.3.2 a)	LU, LN		REL REL COM
					5.8.3.2 b)	LU, LN		REL COM
					5.8.3.2 f)	LU, LN		STAT
82	101	0010	Identified channel does not exist	Channel identity	5.1.4	LN		DISC REL COM
83	101	0011	A suspended call exists, but this call identity does not	–	5.6.5	LN		RES REJ
84	101	0100	Call identity in use	–	5.6.3	LN		SUSP REJ
85	101	0101	No call suspended	–	5.6.5	LN		RES REJ
86	101	0110	Call having the requested call identity has been cleared		5.6.5	LN		RES REJ

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
88	101	1000	Incompatible destination	Incompatible parameter	5.2.2 5.2.5.1 5.2.5.3 a) B.3.2 B.3.3	RU	REL COM	DISC
91	101	1011	Invalid transit network selection	–	C.2	TN		DISC
						LN		DISC REL REL COM
95	101	1111	Invalid message, unspecified	Message type	5.8	LN		REL COM STAT
96	110	0000	Mandatory information element is missing	Information element identifier(s)	5.8.6.1	LN, LU		REL REL COM STAT
					5.8.11	LN, LU		STAT
97	110	0001	Message type non-existent or not implemented	Message type	5.8.4 5.8.10 5.8.11	LU, LN		STAT
98	110	0010	Message not compatible with call state or message type non-existent or not implemented	Message type	5.8.4	LU, LN		STAT

Table I.2/Q.931 – Usage of cause values *(continued)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
99	110	0011	Information element non-existent or not implemented	Information element identifier(s)	5.8.7.1 5.8.11	LU, LN		STAT
					5.8.7.1	LN		REL REL COM
100	110	0100	Invalid information element contents	Information element identifier(s)	5.8.6.2	LU, LN		STAT REL REL COM
					5.8.7.2 5.8.11	LU, LN		STAT
101	110	0101	Message not compatible with call state	Message type	5.8.4	LN, LU		STAT
					5.8.11	LN, LU		DISC REL REL COM
102	110	0110	Recovery on time expiry	Timer number	5.2.4 5.2.5.3 5.6.5	LN		DISC
					5.3.3 5.3.4	LN		REL
					5.3.2 f) 5.3.3 5.6.5	LU		REL

Table I.2/Q.931 – Usage of cause values *(concluded)*

Cause No.	Class	Value	Cause name	Diagnostics	Reference	Typical location of generation	Typical carrying message as identified by receiving side	
							At remote interface	At local interface
111	110	1111	Protocol error, unspecified		5.8.4	RN		DISC
127	111	1111	Interworking, unspecified		No explicit procedure			

APPENDIX II

Example message flow diagrams and example conditions for cause mapping

II.1 Example message flow diagrams

Examples of the procedures for the use of the B- and D-channel network connection types and the selection of the appropriate channel types are summarized in Figures II.1 to II.7. These figures are intended to complement the description in the preceding text and do not illustrate all possible situations.

NOTE – Not all frames that may be sent across the TA interface may be represented in the following figures.

II.1.1 Key to the figures

Q.931 messages

[ ]	Layer 3
C	CONNECT
CA	CONNECT ACKNOWLEDGE
CP	CALL PROCEEDING
D	DISCONNECT
R	RELEASE
RC	RELEASE COMPLETE
S	SETUP

X.25 layer 3 messages

Any layer 3 message preceded by X.25 indicates an X.25 layer 3 packet (e.g. X.25 CR means X.25 *call request*).

CA	Call accepted
CC	Call connected
CLC	Clear confirmation
CLI	Clear indication
CLR	Clear request
CR	Call request
IC	Incoming call
RSR	Restart request
RSC	Restart confirmation

Layer 2 frames

( )	Layer 2
GTEI	Group TEI (127)
A.B	X.25 layer 2 addresses (includes command and response)
SABM	Set Asynchronous Balanced Mode
SABME	Set Asynchronous Balanced Mode Extended

UA Unnumbered acknowledgement frame

UI Unnumbered information frame (i.e. using unacknowledged information transfer at layer 2)

I Information frame

DISC Disconnect frame

Layer 2 addresses marked (x, p) indicate that the SAPI element of the frame address is coded for packet type (SAPI = 16) information as described in Recommendation Q.921. Layer 2 addresses marked (x, s) refer to signalling type (SAPI = 0) information.

II.2 Example conditions for cause mapping

Figures II.8 through II.16 show example conditions when cause mappings would be utilized between Q.931 and X.25 messages and utilize the specific mappings of Tables 6-5 and 6-6 as shown below.

Q.931 failures during call establishment

Figure II.8 Table 6-5

Figure II.9 Table 6-5

Figure II.10 Table 6-5

Figure II.11 Table 6-5

Figure II.12 Table 6-5

User side failures during X.25 data transfer phase

Figure II.13 Table 6-5 (Note 1)

Figure II.14 Table 6-5 (Note 2)

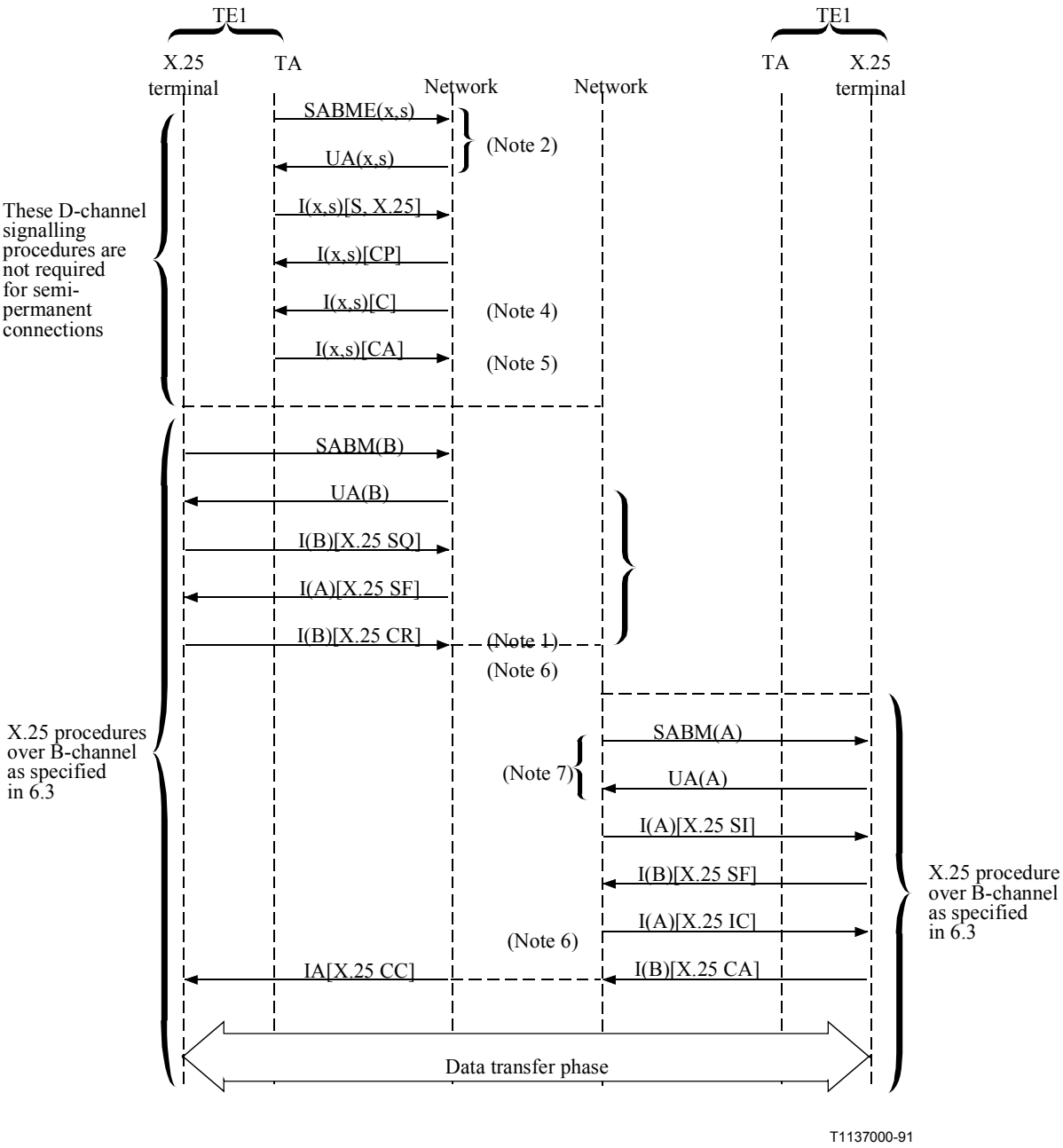
Network side premature clearing

Figure II.15 Table 6-6

Figure II.16 Table 6-6

NOTE 1 – This mapping is only needed in the case of the Q.931 message arriving prior to the clearing of the last virtual call.

NOTE 2 – This situation always results in either an X.25 *clear indication* packet with cause No. 9, *out of order*, for switched virtual calls, or an X.25 *reset* packet with cause No. 9, *out of order*, for permanent virtual circuits.



NOTE 1 – When the called side establishes the call using D-channel access, the message sequence will continue as from point <3> in Figure II.3.

NOTE 2 – If signalling link is not already established.

NOTE 3 – For packet call offering, the incoming call may be offered to the TA and a B-channel established using the procedure shown in Figures II.5 and II.7.

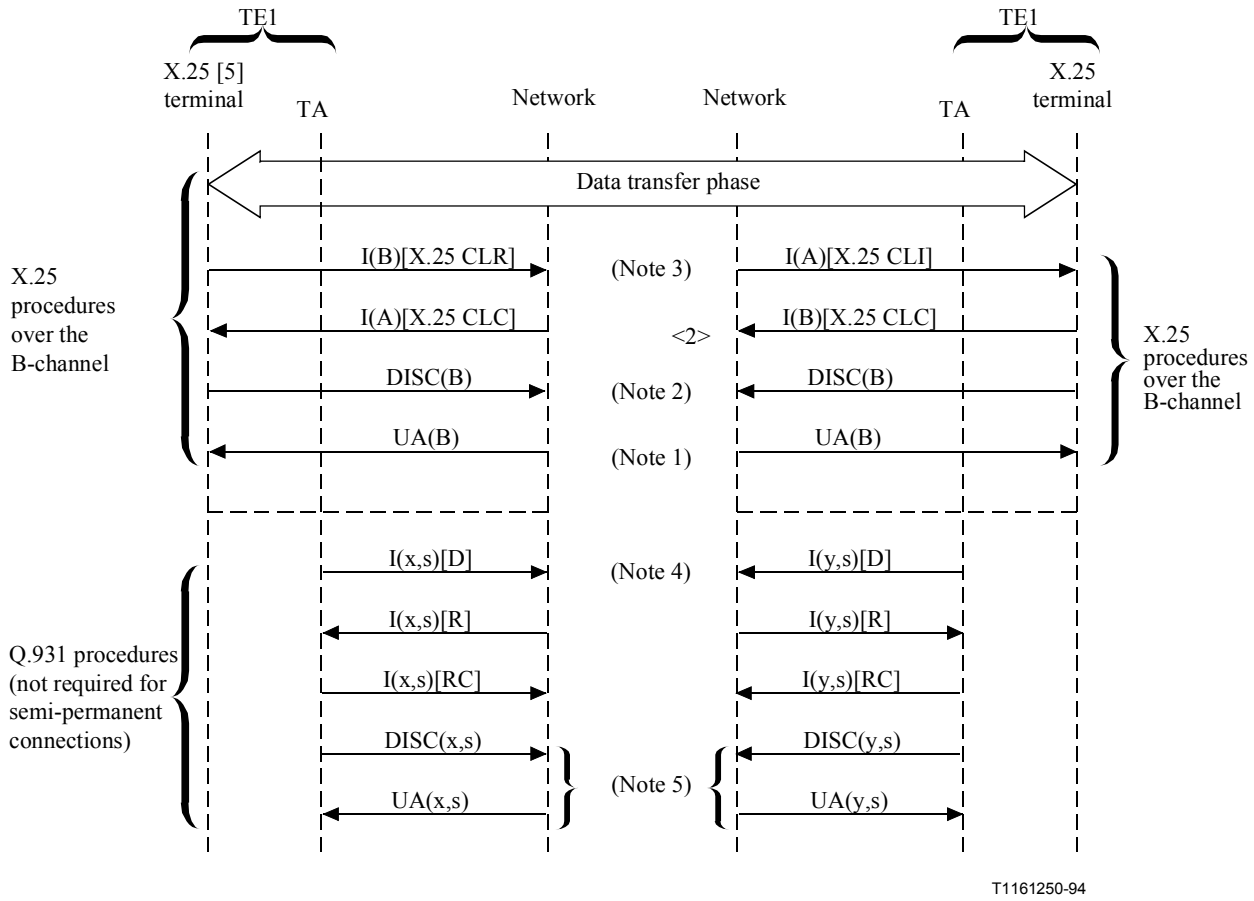
NOTE 4 – The network starts timer T320, if implemented.

NOTE 5 – This message is optional.

NOTE 6 – The network cancels timer T320, if implemented and running.

NOTE 7 – The network establishes the Link Layer on the B-channel, if it is not already established as specified in 6.3.

Figure II.1/Q.931 – Example message sequence for the ISDN virtual circuit service B-channel access – First virtual call set-up in this channel



NOTE 1 – When the cleared side has set up the call using D-channel access, the message sequence at the cleared side will be as from point <4> in Figure II.4.

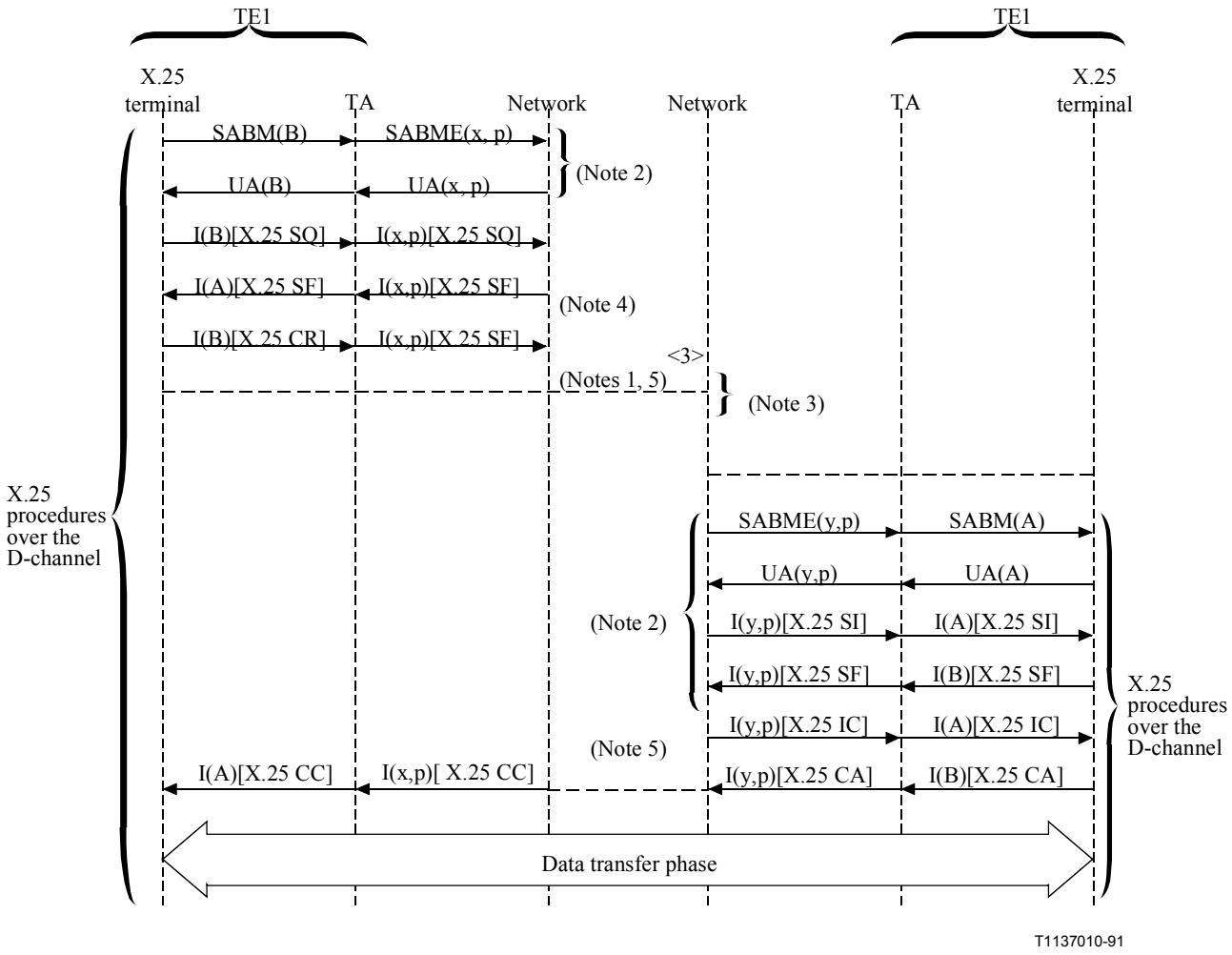
NOTE 2 – Clearing of the B-channel may be initiated by the network upon expiry of Timer T320, if implemented (see 6.4).

NOTE 3 – The network starts Timer T320, if implemented.

NOTE 4 – The network cancels Timer T320, if implemented and running.

NOTE 5 – This sequence is only required if the terminal does not wish to continue with further communication.

Figure II.2/Q.931 – Example message sequence for the ISDN virtual circuit service B-channel access – Last virtual call cleared in this channel



NOTE 1 – When the called side establishes the call using B-channel access, the message sequence will continue as from point <1> in Figure II.1.

NOTE 2 – If SAPI 16 link is not already established.

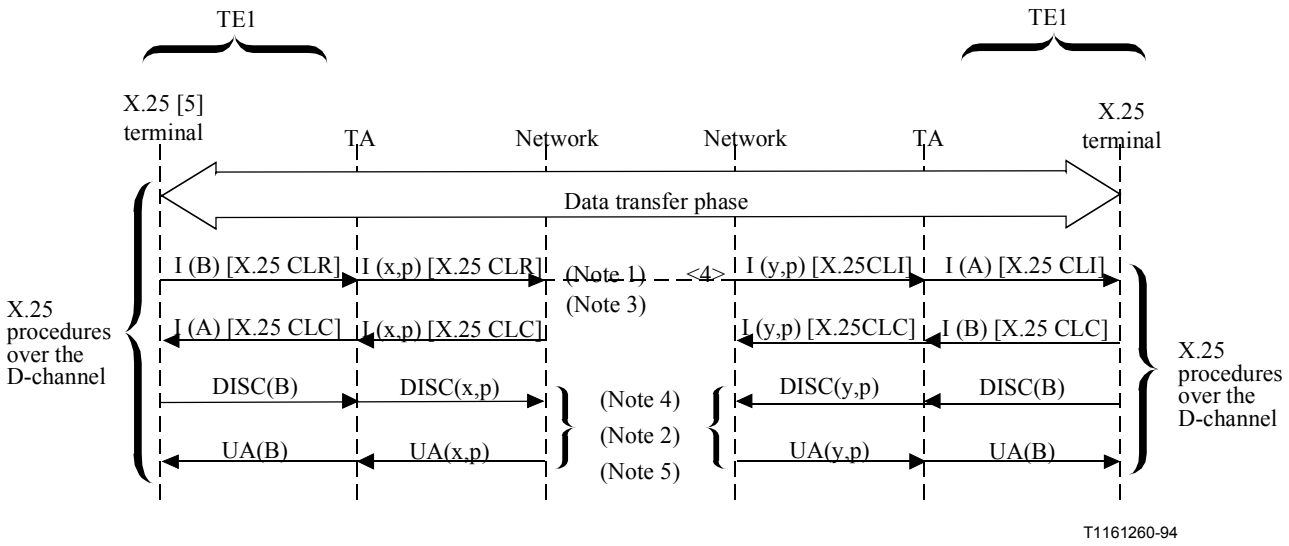
NOTE 3 – The incoming call may be offered to the TA using the procedures shown in Figures II.5 and II.7.

NOTE 4 – The network starts timer T320, if implemented.

NOTE 5 – The network cancels timer T320, if implemented and running.

NOTE 6 – Not shown in the diagram; it is a possible X.25 restart procedure performed after link set-up.

Figure II.3/Q.931 – Example message sequence for the ISDN virtual circuit service D-channel access – First virtual call set-up in this SAPI = 16 link



NOTE 1 – When the cleared side has set up the call using B-channel access, the message sequence at the cleared side will be as from point <2> in Figure II.2.

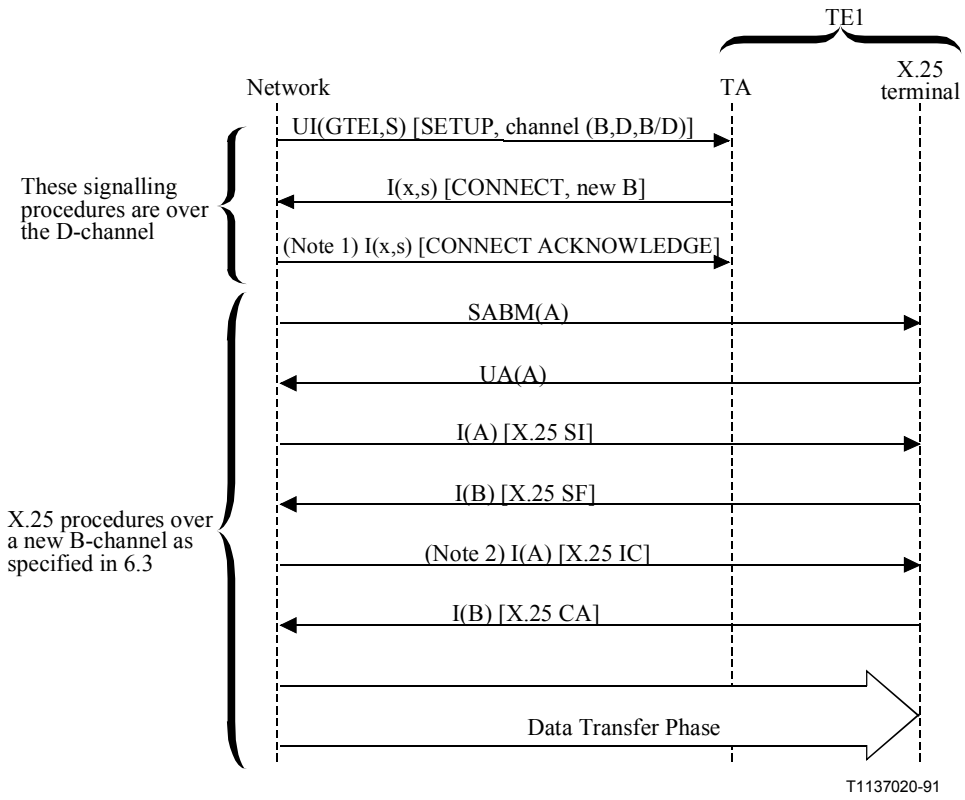
NOTE 2 – This sequence is only required if the X.25 DTE does not wish to continue with further communications.

NOTE 3 – The network starts timer T320, if implemented.

NOTE 4 – The network cancels timer T320, if implemented and running.

NOTE 5 – Link layer release may be initiated by the network upon expiry of Timer T320, if implemented (see 6.4).

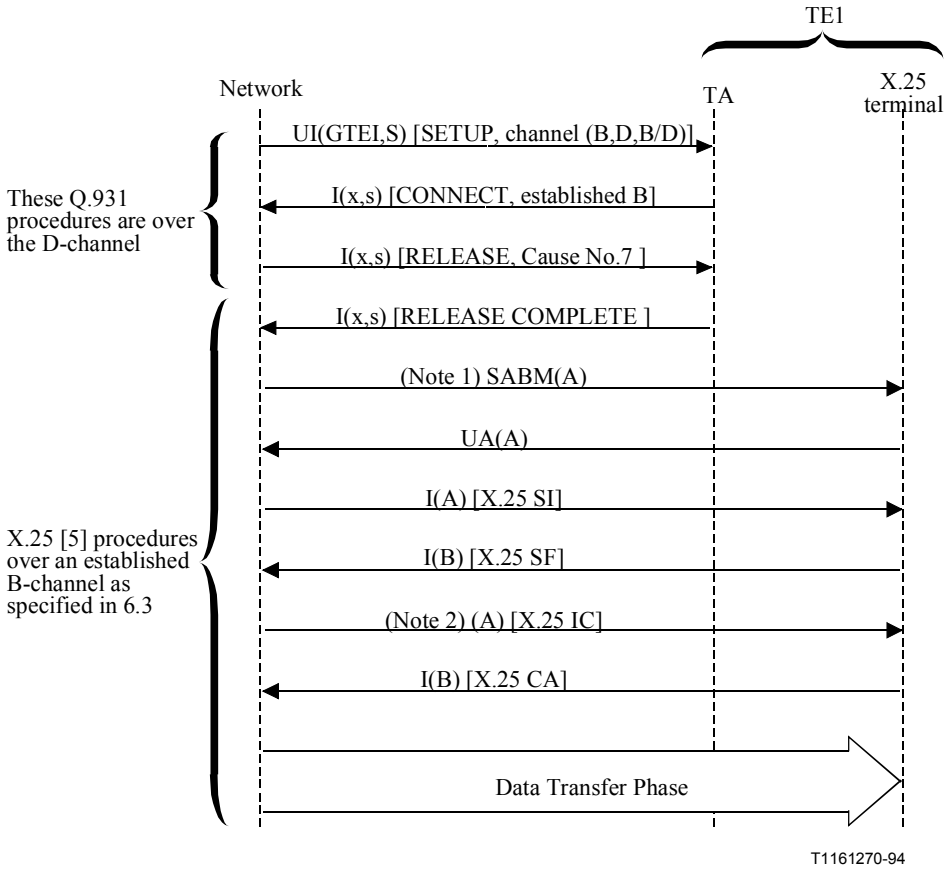
Figure II.4/Q.931 – Example message sequence for the ISDN virtual circuit service D-channel access – Last virtual call cleared in this SAPI = 16 link



NOTE 1 – The network starts Timer T320, if implemented.

NOTE 2 – The network cancels Timer T320, if implemented and running.

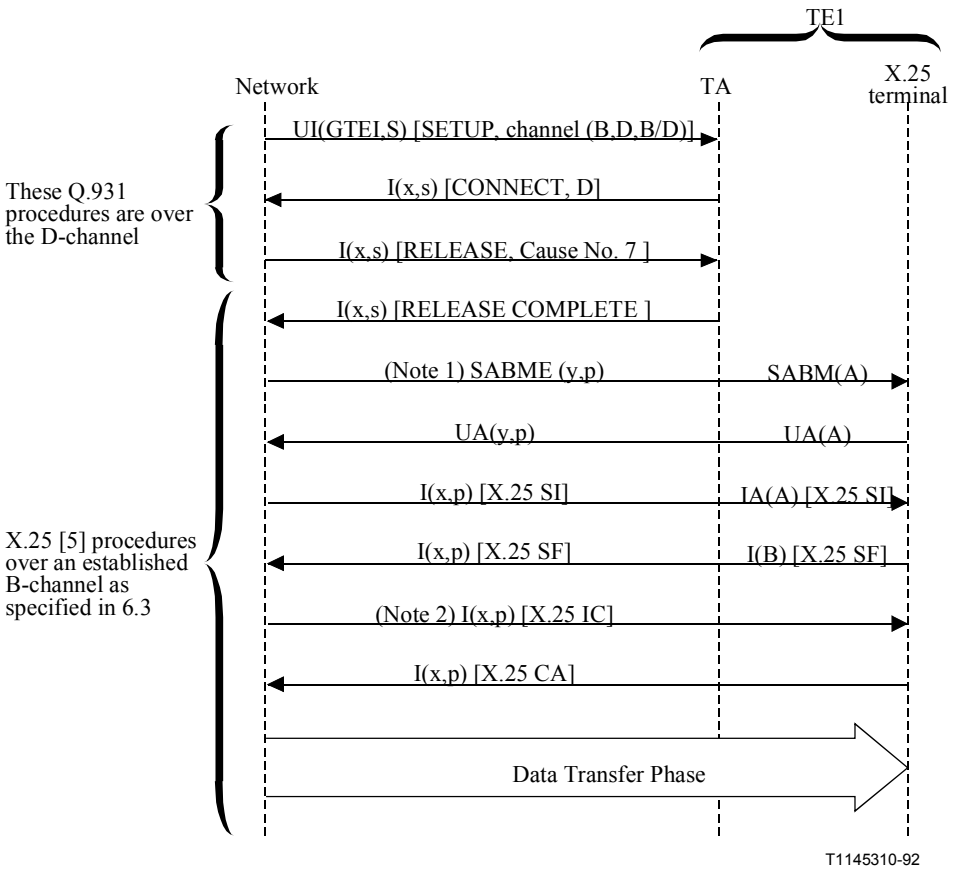
Figure II.5/Q.931 – Example of incoming call offering procedures using signalling on SAPI = 0 link – Terminal accepts call on a new B-channel



NOTE 1 – The network establishes the link layer in the B-channel if it is not already established (see 6.3).

NOTE 2 – The network cancels Timer T320, if implemented and running.

Figure II.6/Q.931 – Example of incoming call offering procedures using signalling on SAPI = 0 link – Terminal accepts call on a established B-channel



NOTE 1 – The network establishes the link layer in the B-channel if it is not already established (see 6.3). The network starts Timer T320, if implemented.

NOTE 2 – The network cancels Timer T320, if implemented and running.

Figure II.7/Q.931 – Example of incoming call offering procedures using signalling on SAPI = 0 link – Terminal accepts call on the D-channel

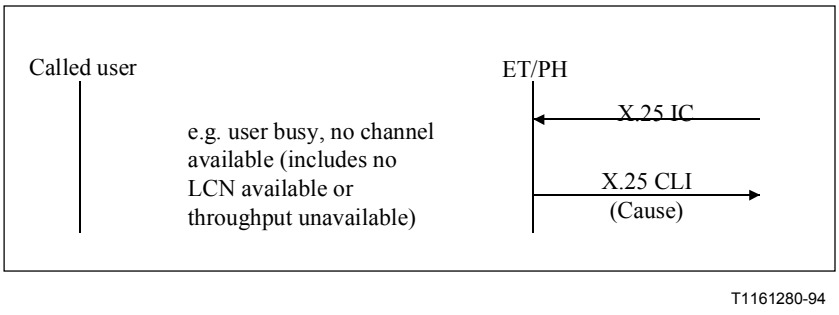
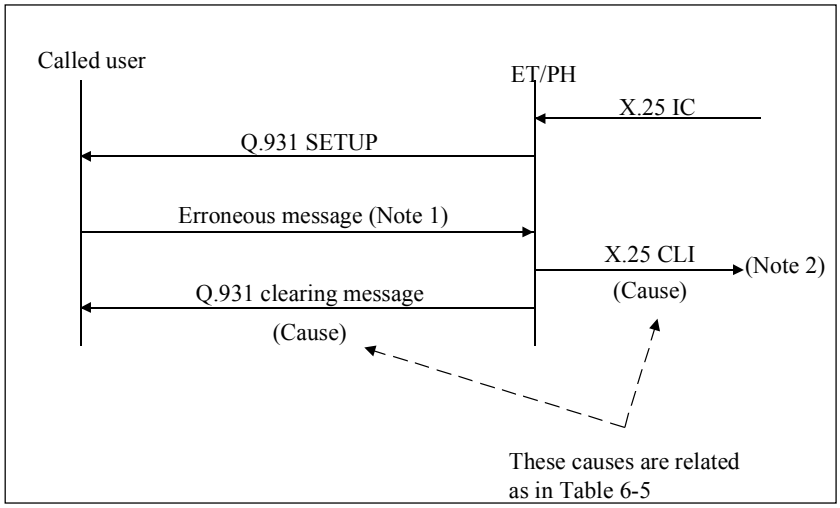


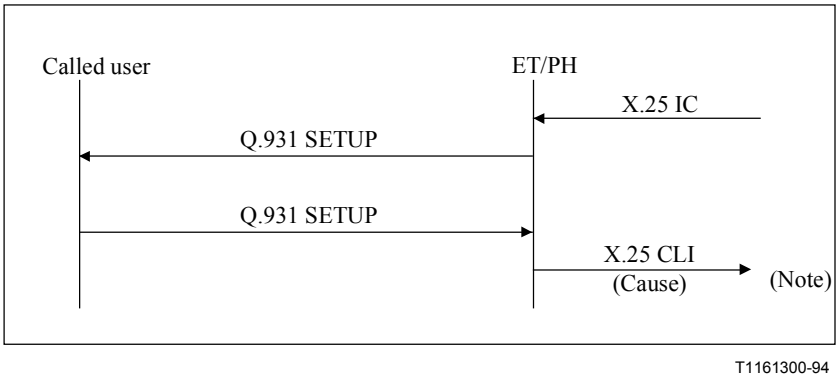
Figure II.8/Q.931 – Undeliverable call



NOTE 1 – This figure only applies to the case where the erroneous message results in a Q.931 clearing message. See 6.4.3 for more information.

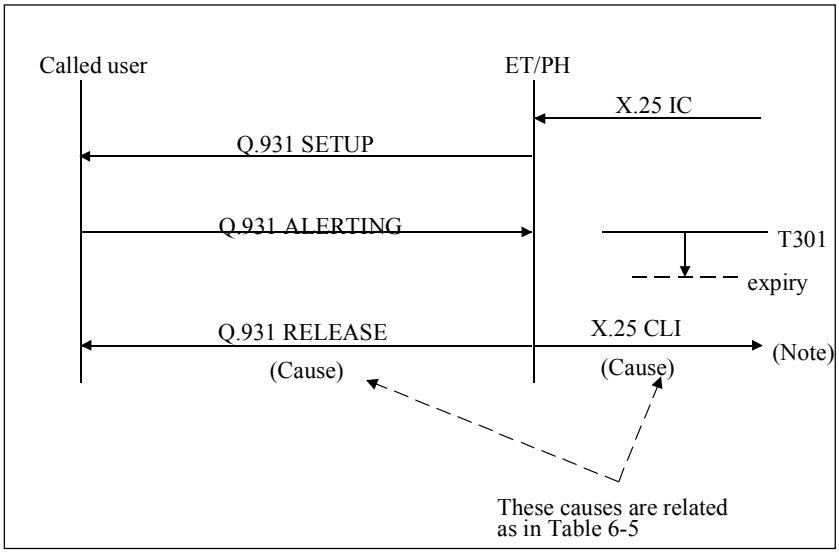
NOTE 2 – This message would be sent after the expiry of timer T303 on a multipoint interface.

Figure II.9/Q.931 – Erroneous message (e.g. format error)



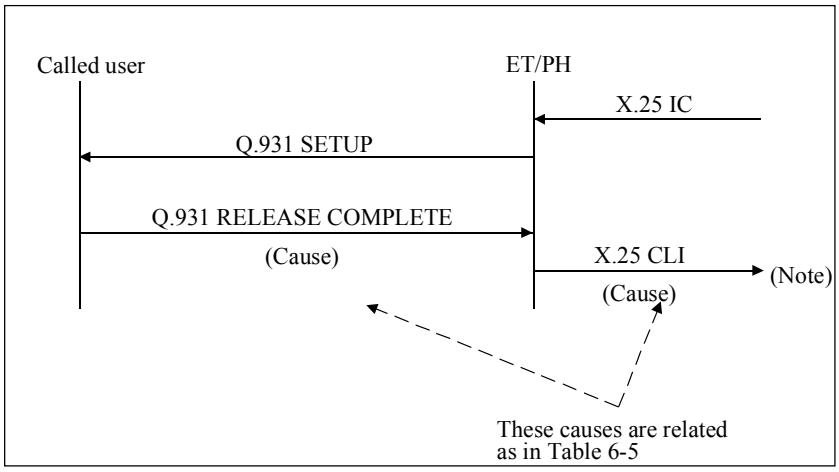
NOTE – This message is sent after the second expiry of timer T303.

Figure II.10/Q.931 – No responding user



NOTE – This message is sent after the expiry of timer T301.

Figure II.11/Q.931 – Expiry of timer T301



NOTE – This message would be sent after the expiry of T303 when on a multipoint interface.

Figure II.12/Q.931 – Call rejection by called party

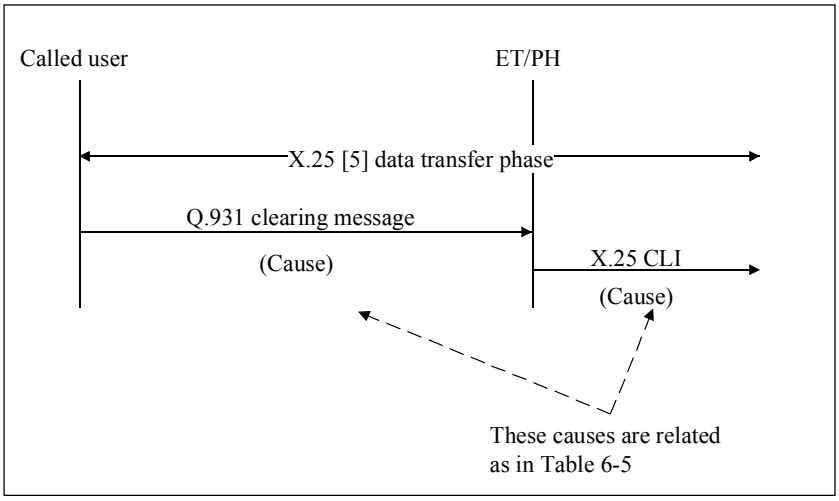
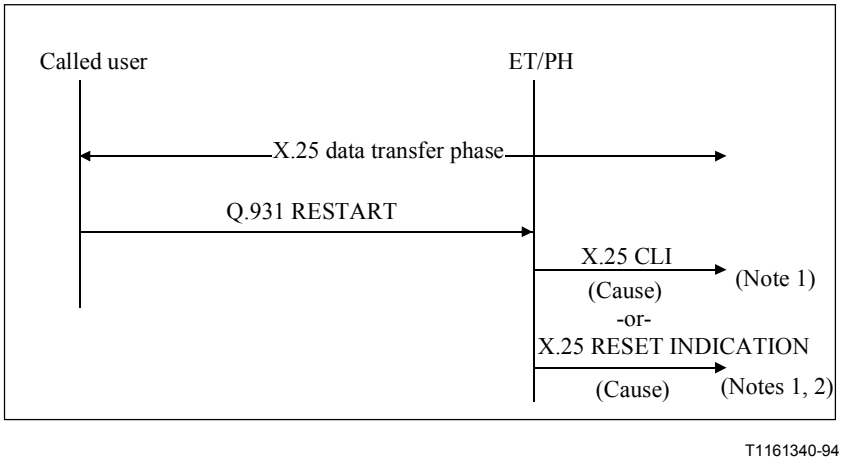


Figure II.13/Q.931 – Q.931 clearing during X.25 data transfer phase



NOTE 1 – This cause parameter in the X.25 packet will indicate "out of order" with diagnostic value 0.

NOTE 2 – For permanent virtual circuits only.

Figure II.14/Q.931 – Q.931 RESTART during X.25 data transfer phase

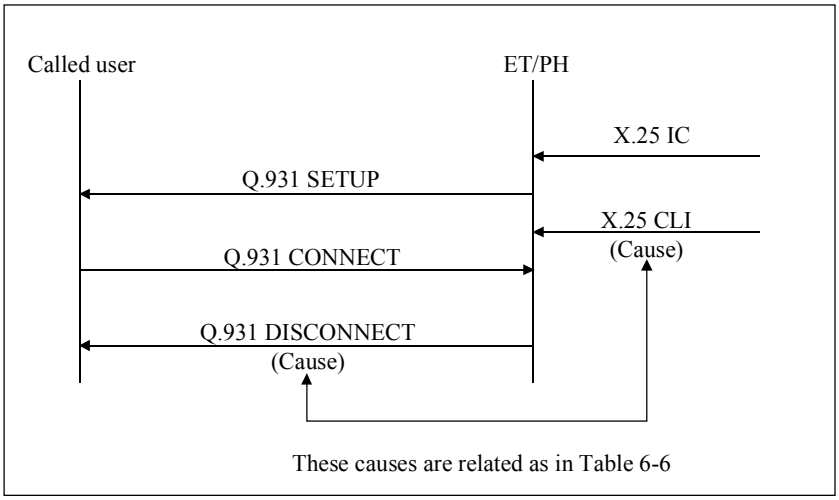
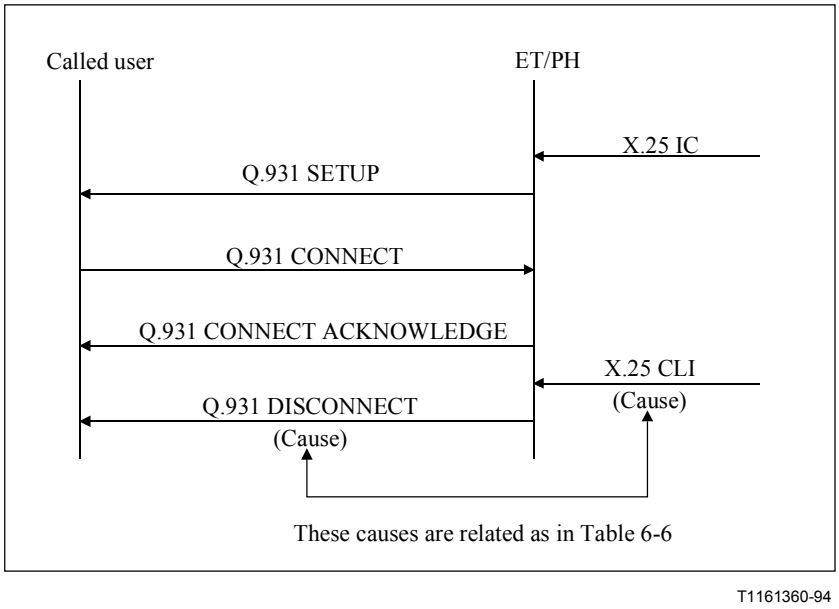


Figure II.15/Q.931 – Premature clearing of the virtual call (e.g. expiry of X.25 timer T21)



NOTE – This is the case when an X.25 incoming call packet has NOT been delivered.

Figure II.16/Q.931 – Premature clearing of the virtual call

APPENDIX III

Summary of assigned information element identifier and message type code points for the Q.93x-series and Q.95x-series of Recommendations

Table III.1/Q.931 – Information element codepoints

Bits 8	7	6	5	4	3	2	1		Recommendation reference
1	:	:	:	-	-	-	-	*Single octet information elements:*	
	0	0	0	-	-	-	-	Reserved	Q.931
	0	0	1	-	-	-	-	Shift	Q.931
	0	1	0	0	0	0	0	More data	Q.931
	0	1	0	0	0	0	1	Sending complete	Q.931
	0	1	1	-	-	-	-	Congestion level	Q.931
	1	0	1	-	-	-	-	Repeat indicator	Q.931
0	:	:	:	:	:	:	:	*Variable length information elements:*	
	0	0	0	0	0	0	0	Segmented message	Q.931
	0	0	0	0	1	0	0	Bearer capability	Q.931
	0	0	0	1	0	0	0	Cause	Q.931
	0	0	0	1	1	0	0	Connected address	(Note 1)
	0	0	0	1	1	0	1	Extended facility	Q.932
	0	0	1	0	0	0	0	Call identity	Q.931
	0	0	1	0	1	0	0	Call state	Q.931
	0	0	1	1	0	0	0	Channel identification	Q.931
	0	0	1	1	0	0	1	Data link connection identifier	Q.933
	0	0	1	1	1	0	0	Facility	Q.932
	0	0	1	1	1	1	0	Progress indicator	Q.931
	0	1	0	0	0	0	0	Network-specific facilities	Q.931
	0	1	0	0	1	0	0	Terminal capabilities	(Note 1)
	0	1	0	0	1	1	1	Notification indicator	Q.931
	0	1	0	1	0	0	0	Display	Q.931
	0	1	0	1	0	0	1	Date/time	Q.931
	0	1	0	1	1	0	0	Keypad facility	Q.931
	0	1	1	0	0	0	0	Keypad echo	(Note 1)
	0	1	1	0	0	1	0	Information request	Q.932 [4]
	0	1	1	0	1	0	0	Signal	Q.931
	0	1	1	0	1	1	0	Switchhook	(Note 1)
	0	1	1	1	0	0	0	Feature activation	Q.932
	0	1	1	1	0	0	1	Feature indication	Q.932
	0	1	1	1	0	1	0	Service profile identification	Q.932
	0	1	1	1	0	1	1	Endpoint identifier	Q.932
	1	0	0	0	0	0	0	Information rate	Q.931

Table III.1/Q.931 – Information element codepoints *(concluded)*

Bits 8	7	6	5	4	3	2	1		Recommendation reference
	1	0	0	0	0	0	1	Precedence level	Q.955 (clause 3)
	1	0	0	0	0	1	0	End-to-end transit delay	Q.931
	1	0	0	0	0	1	1	Transit delay selection and indication	Q.931
	1	0	0	0	1	0	0	Packet layer binary parameters	Q.931
	1	0	0	0	1	0	1	Packet layer window size	Q.931
0	:	:	:	:	:	:	:	*Variable length information elements:*	
	1	0	0	0	1	1	0	Packet size	Q.931
	1	0	0	0	1	1	1	Closed user group	Q.931
	1	0	0	1	0	0	0	Link layer core parameters	Q.933
	1	0	0	1	0	0	1	Link layer protocol parameters	Q.933
	1	0	0	1	0	1	0	Reverse charging indication	Q.931
	1	0	0	1	1	0	0	Connected number	Q.951-series [85]
	1	0	0	1	1	0	1	Connected subaddress	Q.951
	1	0	1	0	0	0	0	X.213 priority	Q.933
	1	0	1	0	0	0	1	Report type	Q.933
	1	0	1	0	0	1	1	Link integrity verification	Q.933
	1	0	1	0	1	1	1	PVC status	Q.933
	1	1	0	1	1	0	0	Calling party number	Q.931
	1	1	0	1	1	0	1	Calling party subaddress	Q.931
	1	1	1	0	0	0	0	Called party number	Q.931
	1	1	1	0	0	0	1	Called party subaddress	Q.931
	1	1	1	0	1	0	0	Redirecting number	Q.931, Q.952 [86]
	1	1	1	0	1	1	0	Redirection number	Q.952
	1	1	1	1	0	0	0	Transit network selection	Q.931
	1	1	1	1	0	0	1	Restart indicator	Q.931
	1	1	1	1	1	0	0	Low layer compatibility	Q.931
	1	1	1	1	1	0	1	High layer compatibility	Q.931
	1	1	1	1	1	1	0	User-user	Q.931
	1	1	1	1	1	1	1	Escape for extension	Q.931

NOTE 1 – These codepoints are reserved to ensure backward compatibility with earlier versions of this Recommendation.

NOTE 2 – All reserved values with bits 5-8 coded "0000" are for future information elements for which comprehension by the user is required (see 5.8.7.1).

Table III.2/Q.931 – Message type codepoints

Bits 8	7	6	5	4	3	2	1		Recommendation reference
0	0	0	0	0	0	0	0	Escape to nationally specific message types	Q.931
0	0	0	-	-	-	-	-	*Call establishment messages:*	
			0	0	0	0	1	ALERTING	Q.931
			0	0	0	1	0	CALL PROCEEDING	Q.931
			0	0	0	1	1	PROGRESS	Q.931
			0	0	1	0	1	SETUP	Q.931
			0	0	1	1	1	CONNECT	Q.931
			0	1	1	0	1	SETUP ACKNOWLEDGE	Q.931
			0	1	1	1	1	CONNECT ACKNOWLEDGE	Q.931
0	0	1	-	-	-	-	-	*Call information phase messages:*	
			0	0	0	0	0	USER INFORMATION	Q.931
			0	0	0	0	1	SUSPEND REJECT	Q.931
			0	0	0	1	0	RESUME REJECT	Q.931
			0	0	1	0	0	HOLD	Q.932 [4]
			0	0	1	0	1	SUSPEND	Q.931
			0	0	1	1	0	RESUME	Q.931
			0	1	0	0	0	HOLD ACKNOWLEDGE	Q.932
			0	1	1	0	1	SUSPEND ACKNOWLEDGE	Q.931
			0	1	1	1	0	RESUME ACKNOWLEDGE	Q.931
			1	0	0	0	0	HOLD REJECT	Q.932
			1	0	0	0	1	RETRIEVE	Q.932
			1	0	0	1	1	RETRIEVE ACKNOWLEDGE	Q.932
			1	0	1	1	1	RETRIEVE REJECT	Q.932
0	1	0	-	-	-	-	-	*Call clearing messages:*	
			0	0	0	0	0	DETACH	(Note)
			0	0	1	0	1	DISCONNECT	Q.931
			0	0	1	1	0	RESTART	Q.931
			0	1	0	0	0	DETACH ACKNOWLEDGE	(Note)
			0	1	1	0	1	RELEASE	Q.931
			0	1	1	1	0	RESTART ACKNOWLEDGE	Q.931
			1	1	0	1	0	RELEASE COMPLETE	Q.931
0	1	1	-	-	-	-	-	*Miscellaneous messages:*	
			0	0	0	0	0	SEGMENT	Q.931
			0	0	0	1	0	FACILITY	Q.932 [4]
			0	0	1	0	0	REGISTER	Q.932

Table III.2/Q.931 – Message type codepoints *(concluded)*

Bits 8 7 6 5 4 3 2 1		Recommendation reference
0 1 0 0 0	CANCEL ACKNOWLEDGE	(Note)
0 1 0 1 0	FACILITY ACKNOWLEDGE	(Note)
0 1 1 0 0	REGISTER ACKNOWLEDGE	(Note)
0 1 1 1 0	NOTIFY	Q.931
1 0 0 0 0	CANCEL REJECT	(Note)
1 0 0 1 0	FACILITY REJECT	(Note)
1 0 1 0 0	REGISTER REJECT	(Note)
1 0 1 0 1	STATUS ENQUIRY	Q.931
1 1 0 0 1	CONGESTION CONTROL	Q.931
1 1 0 1 1	INFORMATION	Q.931
1 1 1 0 1	STATUS	Q.931

NOTE – These codepoints are reserved to ensure backward compatibility with earlier versions of this Recommendation.

III.1 Acronyms used in this Recommendation

ABM	Asynchronous Balanced Mode (of HDLC)
ACK	Acknowledgement
ADPCM	Adaptive Differential Pulse Code Modulation
AFI	Authority and Format Identifier
ARM	Asynchronous Response Mode (of HDLC)
AU	Access Unit
BC	Bearer Capability
BCD	Binary Coded Decimal
Bi	Indicated B-channel
Bi`	An idle B-channel Bi
Bj	A B-Channel in use
CEI	Connection Endpoint Identifier
CES	Connection Endpoint Suffix
CSPDN	Circuit Switched Public Data Network
D	The D-channel
DDI	Direct-Dialling-In
DLCI	Data Link Connection Identifier (see Recommendations Q.920 and Q.921)
DSP	Domain Specific Part

DTE	Data Terminal Equipment
HDLC	High Level Data Link Control (procedures)
HLC	High Layer Compatibility
I	Information (frame)
IA5	International Alphabet No. 5 (defined by ITU-T)
IDI	Initial Domain Identifier
IE	Information Element
IEC	International Electrotechnical Commission
ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
IWF	Interworking Function
IWU	Interworking Unit
LAN	Local Area Network
LAPB	Link Access Protocol-Balanced
LAPD	Link Access Protocol on the D-channel
LLC	Low Layer Compatibility
LLI	Logical Link Identifier (see Recommendation Q.921)
NACK	Negative Acknowledgement
NIC	Network Independent Clock
NRM	Normal Response Mode (of HDLC)
NSAP	Network Service Access Point
NT2	Network Termination of type two
OSI	Open Systems Interconnection
PABX	Private Automatic Branch Exchange
PCM	Pulse Code Modulation
PH	Packet Handler
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
PVC	Permanent Virtual Circuit
RDTD	Restricted Differential Time Delay
RSC	Restart confirmation
RSI	Restart indication
RSR	Restart request
SABME	Set Asynchronous Balanced Mode Extended (frame)
SAPI	Service Access Point Identifier (see Recommendation Q.921)
SDL	Specification and Description Language

TA	Terminal Adaptor (see Recommendation I.411)
TE1	Terminal Equipment of type 1 (see Recommendation I.411)
TE2	Terminal Equipment of type 2 (see Recommendation I.411)
TEI	Terminal Endpoint Identifier (see Recommendations Q.920 and Q.921)
TID	Terminal identifier
UDI	Unrestricted Digital Information
UDI-TA	Unrestricted Digital Information with Tones/Announcements
UI	Unnumbered Information (frame)
USID	User Service Identificator
VC	(Switched) Virtual Circuit

III.2 References

[1] ITU-T Recommendation Q.930/I.450 (1993), *ISDN user-network interface layer 3 – General aspects*.

[2] CCITT Recommendation I.412 (1988), *ISDN user-network interfaces – Interface structures and access capabilities*.

[3] ITU-T Recommendation Q.921/I.441 (1997), *ISDN user-network interface – Data link layer specification*.

[4] ITU-T Recommendation Q.932 (1998), *Digital subscriber signalling system No. 1 – Generic procedures for the control of ISDN supplementary services*.

[5] ITU-T Recommendation X.25 (1996), *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*.

[6] ITU-T Recommendations I.231 series, *Circuit mode bearer service categories*.

[7] ITU-T Recommendation V.110/I.463 (1996), *Support of data terminal equipments with V-series type interfaces by an integrated services digital network.*

[8] ITU-T Recommendation X.30/I.461 (1993), *Support of X.21, X.21 bis and X.20 bis based Data Terminal Equipments (DTEs) by an Integrated Services Digital Network (ISDN)*.

[9] ITU-T Recommendation V.120/I.465 (1996), *Support by an ISDN of data terminal equipment with V-series type interfaces with provision for statistical multiplexing*.

[10] CCITT Recommendation G.711 (1988), *Pulse Code Modulation (PCM) of voice frequencies*.

[11] CCITT Recommendation G.721 (1988 – Withdrawn, replaced by G.726), *32 kbit/s adaptive differential pulse code modulation (ADPCM)*.

[12] CCITT Recommendation G.722 (1988), *7 kHz audio-coding within 64 kbit/s*.

[13] ITU-T Recommendation H.261 (1993), *Video codec for audiovisual services at p × 64 kbit/s*.

[14] ITU-T Recommendation X.31/I.462 (1995), *Support of packet mode terminal equipment by an ISDN*.

[15] CCITT Recommendation I.460 (1988), *Multiplexing, rate adaption and support of existing interfaces*.

[16] CCITT Recommendation V.6 (1988 – Withdrawn), *Standardization of data signalling rates for synchronous data transmission on leased telephone-type circuits*.

[17] ITU-T Recommendation X.1 (1996), *International user classes of service in, and categories of access to public data networks and Integrated Services Digital Networks (ISDNs)*.

[18] CCITT Recommendation I.330 (1988), *ISDN numbering and addressing principles*.

[19] ITU-T Recommendation E.164 (1997), *The international public telecommunication numbering plan*.

[20] CCITT Recommendation E.163 (1988 – Now superseded by E.164), *Numbering plan for the international telephone service*.

[21] ITU-T Recommendation X.121 (1996), *International numbering plan for public data networks*.

[22] ITU-T Recommendation F.69 (1994), *The international telex service – Service and operational provisions of telex destination codes and telex network identification codes*.

[23] ITU-T Recommendation X.213 (1995) | ISO/IEC 8348:1996, *Information technology – Open Systems Interconnection – Network service definition*.

[24] ISO/IEC 8348:1988, *Information processing systems – Data communications – Network service definition – Addendum 2: Network layer addressing*.

[25] CCITT Recommendation I.334 (1988), *Principles relating ISDN numbers/sub-addresses to the OSI reference model network layer addresses*.

[26] CCITT Recommendation X.21 (1992), *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for synchronous operation on public data networks*.

[27] ITU-T Recommendation I.431 (1993), *Primary rate user-network interface – Layer 1 specification*.

[28] ITU-T Recommendation T.62 (1993), *Control procedures for teletex and Group 4 facsimile services*.

[29] CCITT Recommendation T.503 (1991), *A document application profile for the interchange of Group 4 facsimile documents*.

[30] ITU-T Recommendation T.501 (1993), *Document application profile MM for the interchange of formatted mixed mode documents*.

[31] ITU-T Recommendation T.502 (1994), *Document application profile PM-11 for the interchange of simple structure, character content documents in processable and formatted forms*.

[32] ITU-T Recommendation T.70 (1993), *Network-independent basic transport service for the telematic services*.

[33] ITU-T Recommendation T.504 (1993), *Document application profile for videotex interworking*.

[34] ITU-T Recommendations I.241-series, *Teleservices supported by an ISDN*.

[35] CCITT Recommendation G.725 (1988), *System aspects for the use of the 7 kHz audio codec within 64 kbit/s*.

[36] ISO 1745:1975, *Information processing – Basic mode control procedures for data communication systems*.

[37] CCITT Recommendation T.71 (1988), *Link Access Protocol Balanced (LAPB) extended for half-duplex physical level facility*.

[38] ISO/IEC 4335:1993, *Information technology – Telecommunications and information exchange between systems – High-level Data Link Control (HDLC) procedures – Elements of procedures*.

[39] ISO/IEC 8802-2:1998, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control*.

[40] ITU-T Recommendation X.75 (1996), *Packet-switched signalling system between public networks providing data transmission services*.

[41] ISO/IEC 8208:1995, *Information technology – Data communications – X.25 Packet Layer Protocol for Data Terminal Equipment*.

[42] ISO/IEC 8348:1987, *Information processing systems – Data communications – Network service definition*.

[43] ISO/IEC 8473:1988, *Information processing systems – Data communications – Protocol for providing the connectionless-mode network service*.

[44] CCITT Recommendation X.244 (1988 – Withdrawn), *Procedure for the exchange of protocol identification during virtual call establishment on packet switched public data networks*.

[45] ITU-T Recommendation Q.920/I.440 (1993), *ISDN user-network interface data link layer – General aspects*.

[46] ITU-T Recommendation I.430 (1995), *Basic user-network interface – Layer 1 specification*.

[47] CCITT Recommendation I.230 (1988), *Definition of bearer service categories*.

[48] CCITT Recommendation I.240 (1988), *Definition of teleservices*.

[49] CCITT Recommendation T.50 (1992), *International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5) – Information technology – 7-bit coded character set for information exchange*.

[50] ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*.

[51] ITU-T Recommendations *on Integrated services digital network (ISDN)*.

[52] ITU-T Recommendation V.110/I.463 (1996), *Support of data terminal equipments with V-series type interfaces by an integrated services digital network (ISDN)*.

[53] CCITT Recommendation Q.931 (1988), *ISDN user-network interface layer 3 specification for basic call control*.

[54] ITU-T Recommendation Q.957, *Stage 3 description for additional information transfer supplementary services using DSS1*.

[55] CCITT Recommendation V.21 (1984), *300 bits per second duplex modem standardized for use in the general switched telephone network*.

[56] CCITT Recommendation V.22 (1988), *1200 bits per second duplex modem standardized for use in the general switched telephone network and on point-to-point 2-wire leased telephone-type circuits*.

[57] CCITT Recommendation V.22 *bis* (1988), *2400 bits per second duplex modem using frequency division technique standardized for use in the general switched telephone network and on point-to-point 2-wire leased telephone-type circuits*.

[58] CCITT Recommendation V.23 (1988), *600/1200-baud modem standardized for use in the general switched telephone network.*

[59] CCITT Recommendation V.26 (1984), *2400 bits per second modem standardized for use on 4-wire leased telephone-type circuits*.

[60] CCITT Recommendation V.26 *bis* (1984), *2400/1200 bits per second modem standardized for use in the general switched telephone network.*

[61] CCITT Recommendation V.26 *ter* (1988), *2400 bits per second duplex modem using the echo cancellation technique standardized for use on the general switched telephone network and on point-to-point 2-wire leased telephone type circuits*.

[62] CCITT Recommendation V.27 (1984), *4800 bits per second modem with manual equalizer standardized for use on leased telephone-type circuits*.

[63] CCITT Recommendation V.27 *bis* (1984), 4800/*2400 bits per second modem with automatic equalizer standardized for use on leased telephone-type circuits*.

[64] CCITT Recommendation V.27 *ter* (1984), *4800/2400 bits per second modem standardized for use in the general switched telephone network.*

[65] CCITT Recommendation V.29 (1988), *9600 bits per second modem standardized for use on point-to-point 4-wire leased telephone-type circuits*.

[66] ITU-T Recommendation V.32 (1993), *A family of 2-wire, duplex modems operating at data signalling rates of up to 9600 bit/s for use on the general switched telephone network and on leased telephone-type circuits*.

[67] ITU-T Recommendation Q.850 (1998), *Usage of cause and location in the digital subscriber signalling system No. 1 and the Signalling System No. 7 ISDN user part*.

[68] ITU-T Recommendation F.182 (1996), *Operational provisions for the international public facsimile service between subscribers with Group 3 facsimile machines (Telefax 3)*.

[69] ITU-T Recommendation F.184 (1996), *Operational provisions for the international public facsimile service between subscribers stations with Group 4 facsimile machines (Telefax 4)*.

[70] CCITT Recommendation F.230 (1988 – Withdrawn), *Service requirements unique to Mixed Mode (MM) used within the teletex service*.

[71] ITU-T Recommendation F.220 (1993 – Withdrawn), *Service requirements unique to the processable mode number eleven (PM11) used within the teletex service*.

[72] CCITT Recommendation F.200 (1992 – Withdrawn), *Teletex service*.

[73] ITU-T Recommendation F.300 (1993), *Videotex service*.

[74] ITU-T Recommendation T.102 (1993), *Syntax-based videotex end-to-end protocols for the circuit mode ISDN*.

[75] ITU-T Recommendation T.101 (1994), *International interworking for videotex services*.

[76] CCITT Recommendation F.60 (1992), *Operational provisions for the international telex service*.

[77] ITU-T Recommendations X.400-series, *Message Handling Systems (MHS)*.

[78] ITU-T Recommendations X.200-series, *Open Systems Interconnection (OSI): Model and notation*.

[79] CCITT Recommendation F.721 (1992), *Videotelephony teleservice for ISDN*.

[80] ITU-T Recommendations F.700-series, *Audiovisual services*.

[81] ISO/IEC 8878:1992, *Information technology – Telecommunications and information exchange between systems – Use of X.25 to provide the OSI Connection-mode Network Service*.

[82] ISO/IEC TR 9577:1996, *Information technology – Protocol identification in the network layer*.

[83] ITU-T Recommendation Q.95x-series, *Stage 3 description for number identification supplementary services using DSS1*.

[84] ITU-T Recommendations Q.953-series, *Stage 3 description for call completion supplementary services using DSS1*.

[85] CCITT Recommendation Q.951-series, *Stage 3 description for number identification supplementary services using DSS1*.

[86] ITU-T Recommendation Q.952 (1993), *Stage 3 service description for call offering supplementary services using DSS1*.

[87] ITU-T Recommendation V.110 (1996), *Support of data terminal equipments with V-series type interfaces by an integrated services digital network* (in relation to 38.4 kbit/s).

[88] ITU-T Recommendation V.14 (1993), *Transmission of start-stop characters over synchronous bearer channels*.

[89] ITU-T Recommendation V.110 (1996), *Support of data terminal equipments with V-series type interfaces by an integrated services digital network* (in relation to 28.8 kbit/s and 24 kbit/s).

[90] ITU-T Recommendation V.34 (1998), *A modem operating at data signalling rates of up to 33 600 bit/s for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits*.

[91] CCITT Recommendation F.720 (1992), *Videotelephony services – General*.

[92] ITU-T Recommendation H.223 (1996), *Multiplexing protocol for low bit rate multimedia communication*.

[93] ITU-T Recommendation H.245 (1998), *Control protocol for multimedia communication*.

[94] ITU-T Recommendation F.702 (1996), *Multimedia conference services*.

[95] ITU-T Recommendation F.700 (1996), *Framework Recommendation for audiovisual multimedia services*.

[96] ITU-T Recommendation X.223 (1993), *Use of X.25 to provide the OSI connection-mode network service for ITU-T applications.*.

[97] ITU-T Recommendation F.731 (1997), *Multimedia conference services in the ISDN*.

ITU-T RECOMMENDATIONS SERIES

Series A	Organization of the work of the ITU-T
Series B	Means of expression: definitions, symbols, classification
Series C	General telecommunication statistics
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	TMN and network maintenance: international transmission systems, telephone circuits, telegraphy, facsimile and leased circuits
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	**Switching and signalling**
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks and open system communications
Series Y	Global information infrastructure
Series Z	Programming languages

Telcordia™ Technologies

Performance from Experience

Telcordia Notes on the Networks

Telcordia Technologies Special Report
SR-2275
Issue 4
October 2000

An SAIC Company

Telcordia Notes on the Networks

SR-2275 replaces SR-2275, *Bellcore Notes on the Networks*, Issue 3, December 1997.

Related documents:

SR-NOTES-SERIES-01, *Telcordia Notes on the Synchronous Optical Network (SONET)*
SR-NOTES-SERIES-02, *Telcordia Notes on Dense Wavelength-Division Multiplexing (DWDM) and Optical Networking*
SR-NOTES-SERIES-03, *Telcordia Notes on Number Portability and Number Pooling*
SR-NOTES-SERIES-04, *Telcordia Notes on the Evolution of Enhanced Emergency Services.*

To obtain copies of this document, contact your company's document coordinator or your Telcordia account manager, or call +1 800.521.2673 (from the USA and Canada) or +1 732.699.5800 (all others), or visit our Web site at www.telcordia.com. Telcordia employees should call +1 732.699.5802.

Trademark Acknowledgments

Telcordia is a trademark of Telcordia Technologies, Inc.
CLCI, CLEI, CLFI, CLLI, ISCP, NMA, and SEAS are trademarks of Telcordia Technologies, Inc.
COMMON LANGUAGE, SPACE, TELEGATE, AIRBOSS, and TIRKS are registered trademarks of Telcordia Technologies, Inc.
CLASS is a service mark of Telcordia Technologies, Inc.
Appletalk is a registered trademark of Apple Computer, Inc.
DECNet is a trademark of Digital Equipment Corporation.
1/1AESS, 4ESS, 5ESS, Dataphone, and SLC are registered trademarks of Lucent Technologies, Inc.
DMS-10, DMS-100F, DATAPATH, and TOPS are trademarks of Nortel.
DMS-100 is a registered trademark of Nortel.
NEAX-61E is a trademark of NEC America, Inc.
EWSD is a registered trademark of Siemens AG.

Any other companies and products not specifically mentioned herein are trademarks or service marks of their respective trademark and service mark owners.

For example, a “Termination Attempt” DN trigger can be placed at this TDP to support services such as Carrier Access Restriction (CAR) and Personal Communications Services (PCS). In response to the above triggers, AIN service logic in an SCP can request an AIN 0.1 SSP to perform actions such as rerouting the call or playing a terminating or interactive announcement to the caller.

Evolving AIN SSP capabilities support four new call processing triggers: O_Called_Party_Busy, O_No_Answer, T_Busy, and T_No_Answer. These triggers allow AIN SSPs to detect a busy condition from the originating or terminating end of a call, and to detect when the called party does not answer from the originating or terminating end of a call. These new triggers provide AIN with the capability to redirect calls on busy/no answer. Other TDPs defined subsequent to AIN 0.1 include: O_Term_Seized, O_Answer, and Term_Resource_Available, as shown in Figures 14-15 and Figure 14-16[10]. The Off_Hook_Delay trigger has been extended to apply to ISDN PRI interfaces. The 3/6/10-digit trigger has been extended to trigger on any number of three to ten digits and has been renamed “Specific_Digit_String” trigger to reflect this extension.

In addition, AIN provides the non-call related functions such as:

- *Monitor* — Allows an SSP to notify an SCP when a designated facility, such as a line, changes status.
- *Update* — Allows an SCP to change the status of triggers in an SSP, for example, from inactive to active.
- *Non-Call Associated Signalling* –– Allows the exchange of data between an IP and an SCP.

14.7.5.2 Event Detection Points (EDPs)

Events are detected as a result of processing a call. AIN enables an SCP to send a *list* of subsequent events that may occur during a call handled by an AIN SSP such that when one of the events on the list occurs, the SSP may be required to suspend call processing and launch a query to the SCP. This list of events is known as a Next Event List (NEL). The NEL allows an SCP to request information regarding the status of a call (e.g., network busy conditions, called party busy conditions). When a NEL request is made, the TCAP transaction remains open between the SCP and SSP, and the SCP awaits notification of the event from the SSP.

Like TDPs, EDPs are associated with PICs. However, requested events are not administered at the SSP. The SCP activates EDPs dynamically (during an already open transaction) in the form of a returned NEL. The SCP activates EDPs (during an already open transaction) by sending the SSP a NEL. The SSP detects the need for additional AIN control when an event included in the NEL is encountered at an EDP. There are two types of requested events: EDP-Requests and EDP-

10. For more detailed descriptions of trigger detection points and associated triggers, refer to GR-1298-CORE, Issue 3.

Notifications. When the SSP recognizes an event as an EDP-Request, the SSP stops call processing, sends an EDP-Request message to the SCP, and awaits instruction from the SCP for further call processing. When the SSP recognizes an event as an EDP-Notification, the SSP does not stop call processing, but sends an EDP-Notification message to the SCP. Upon receiving an EDP-Notification message, the SCP does not respond to the SSP, but may record the occurrence of the event for subsequent processing.

Some EDPs supported by AIN include the following:

A.EDP-R

- *Origination_Attempt* – tells the calling party's service that an off-hook indication or SETUP message is received by the SSP.
- *Network Busy* – tells the calling party's service that the network beyond the AIN switch cannot complete the call due to no available routes.
- *O_Called_Party_Busy* – tells the calling party's service that the called party is busy.
- *O_No_Answer* – tells the calling party's service that the called party has not answered the call before a timer expired.
- *O_Suspended* – tells the calling party's service that the called party has released the call.
- *O_Disconnect* – tells the calling party's service that the called party has released the call and disconnect timing has completed.
- *O_Mid_Call* – tells the calling party's service that a switch-hook flash (analog) or a feature activator indication (ISDN) has been received.
- *T_Busy* – tells the called party's service that the subscriber's line is not idle or is unable to receive calls.
- *T_Mid_Call* – tells the called party's service that a switch-hook flash (analog) or a feature activator indication (ISDN) has been received.
- *T_No_Answer* – tells the called party's service that the subscriber's line has not answered the call before a timer expired.
- *T_Disconnect* – tells the called party's service that the called party has released the call and disconnect timing has completed.

B.EDP-N

- *Origination_Attempt* – tells the calling party's service that an off-hook indication or SETUP is received by the SSP.
- *O_Term_Seized* – tells the calling party's service that the called party's access has been successfully seized.
- *O_Answer* – tells the calling party's service that the called party has answered the call.

Telcordia™ Technologies

Performance from Experience

Telcordia Notes on the Networks

Telcordia Technologies Special Report
SR-2275
Issue 4
October 2000

An SAIC Company

Telcordia Notes on the Networks

SR-2275 replaces SR-2275, *Bellcore Notes on the Networks*, Issue 3, December 1997.

Related documents:

SR-NOTES-SERIES-01, *Telcordia Notes on the Synchronous Optical Network (SONET)*
SR-NOTES-SERIES-02, *Telcordia Notes on Dense Wavelength-Division Multiplexing (DWDM) and Optical Networking*
SR-NOTES-SERIES-03, *Telcordia Notes on Number Portability and Number Pooling*
SR-NOTES-SERIES-04, *Telcordia Notes on the Evolution of Enhanced Emergency Services.*

To obtain copies of this document, contact your company's document coordinator or your Telcordia account manager, or call +1 800.521.2673 (from the USA and Canada) or +1 732.699.5800 (all others), or visit our Web site at www.telcordia.com. Telcordia employees should call +1 732.699.5802.

Trademark Acknowledgments

Telcordia is a trademark of Telcordia Technologies, Inc.
CLCI, CLEI, CLFI, CLLI, ISCP, NMA, and SEAS are trademarks of Telcordia Technologies, Inc.
COMMON LANGUAGE, SPACE, TELEGATE, AIRBOSS, and TIRKS are registered trademarks of Telcordia Technologies, Inc.
CLASS is a service mark of Telcordia Technologies, Inc.
Appletalk is a registered trademark of Apple Computer, Inc.
DECNet is a trademark of Digital Equipment Corporation.
1/1AESS, 4ESS, 5ESS, Dataphone, and SLC are registered trademarks of Lucent Technologies, Inc.
DMS-10, DMS-100F, DATAPATH, and TOPS are trademarks of Nortel.
DMS-100 is a registered trademark of Nortel.
NEAX-61E is a trademark of NEC America, Inc.
EWSD is a registered trademark of Siemens AG.

Any other companies and products not specifically mentioned herein are trademarks or service marks of their respective trademark and service mark owners.

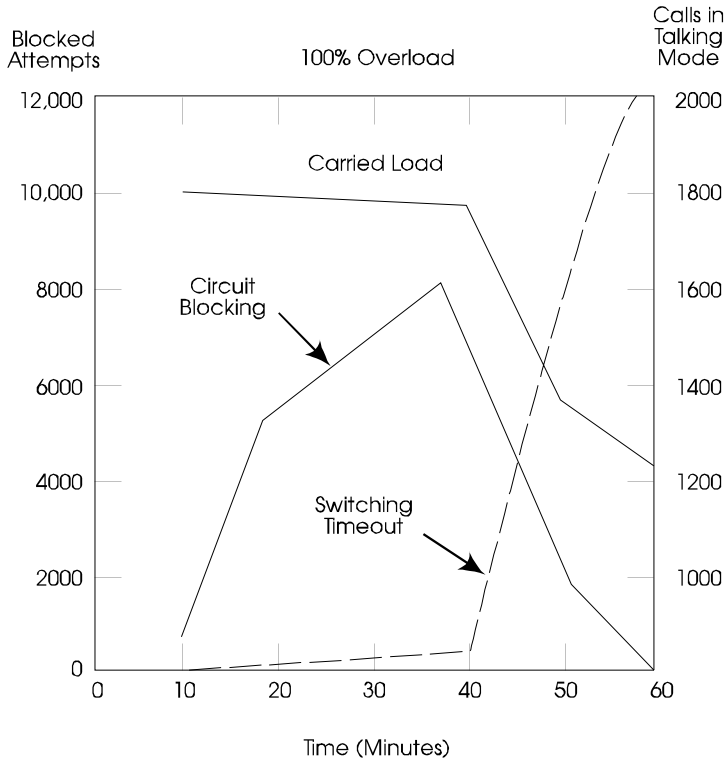


Figure 10-2. Network Congestion

At the onset of the overload, also known as circuit shortage, the dominant cause for customer blockage is the failure to find an idle circuit. Circuit blocking alone limits the number of extra calls that can be completed but does not cause a significant loss in the call-carrying capacity of the network below its maximum. As the overload persists and the network enters a congested state, regenerated-calling pressure changes customer blockage from circuit shortage to switching delays.

Switching delays cause timeout conditions during call setup and occur when switching systems become severely overloaded. Timeouts are designed into switching systems to release common-control components after excessively long delay periods and provide the customer with a signal indicating call-attempt failure. Switching-congestion timeouts with short holding-time attempts on circuit groups replace normal holding-time calls. Switching delays spread quickly throughout the network.

- *A trunk-group overload* usually occurs during general or focused overloads and/or atypical busy hours. Some of the overload causes not discussed above are facility outages, inadequate trunk provisioning, and routing errors. The results of a trunk-group overload can be essentially the same as those previously discussed for general overloads. However, the adverse effects are usually confined to the particular trunk group or the apex area formed by the trunk group and those groups' alternate-routing to the overloaded trunk group.