

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

1. STATUS OF THIS MEMO	1
2. INTRODUCTION	3
2.1. Overview	3
2.2. Common configurations	4
2.3. Conventions	7
2.3.1. Preferred name syntax	7
2.3.2. Data Transmission Order	8
2.3.3. Character Case	9
2.3.4. Size limits	10
3. DOMAIN NAME SPACE AND RR DEFINITIONS	10
3.1. Name space definitions	10
3.2. RR definitions	11
3.2.1. Format	11
3.2.2. TYPE values	12
3.2.3. QTYPE values	12
3.2.4. CLASS values	13

Mockapetris

[Page 1]

RFC 1035

Domain Implementation and Specification

November 1987

3.2.5. QCLASS values

13

3.3. Standard RRs	13
3.3.1. CNAME RDATA format	14
3.3.2. HINFO RDATA format	14
3.3.3. MB RDATA format (EXPERIMENTAL)	14
3.3.4. MD RDATA format (Obsolete)	15
3.3.5. MF RDATA format (Obsolete)	15
3.3.6. MG RDATA format (EXPERIMENTAL)	16
3.3.7. MINFO RDATA format (EXPERIMENTAL)	16
3.3.8. MR RDATA format (EXPERIMENTAL)	17
3.3.9. MX RDATA format	17
3.3.10. NULL RDATA format (EXPERIMENTAL)	17
3.3.11. NS RDATA format	18
3.3.12. PTR RDATA format	18
3.3.13. SOA RDATA format	19
3.3.14. TXT RDATA format	20
3.4. ARPA Internet specific RRs	20
3.4.1. A RDATA format	20
3.4.2. WKS RDATA format	21
3.5. IN-ADDR.ARPA domain	22
3.6. Defining new types, classes, and special namespaces	24
4. MESSAGES	25
4.1. Format	25
4.1.1. Header section format	26
4.1.2. Question section format	28
4.1.3. Resource record format	29
4.1.4. Message compression	30
4.2. Transport	32
4.2.1. UDP usage	32
4.2.2. TCP usage	32
5. MASTER FILES	33
5.1. Format	33
5.2. Use of master files to define zones	35
5.3. Master file example	36
6. NAME SERVER IMPLEMENTATION	37
6.1. Architecture	37
6.1.1. Control	37
6.1.2. Database	37
6.1.3. Time	39
6.2. Standard query processing	39
6.3. Zone refresh and reload processing	39
6.4. Inverse queries (Optional)	40
6.4.1. The contents of inverse queries and responses	40
6.4.2. Inverse query and response example	41
6.4.3. Inverse query processing	42

7.2. Sending the queries	44
7.3. Processing responses	46
7.4. Using the cache	47
8. MAIL SUPPORT	47
8.1. Mail exchange binding	48
8.2. Mailbox binding (Experimental)	48
9. REFERENCES and BIBLIOGRAPHY	50
Index	54

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

Mockapetris

[Page 3]

☐

RFC 1035

Domain Implementation and Specification

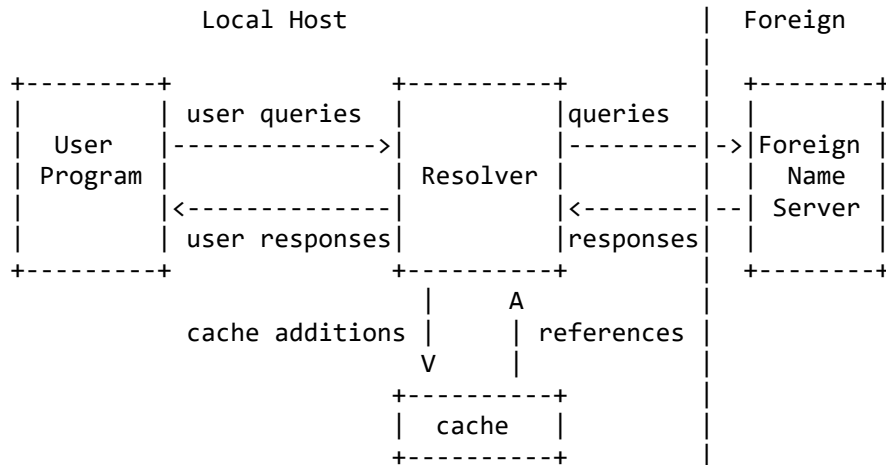
November 1987

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.

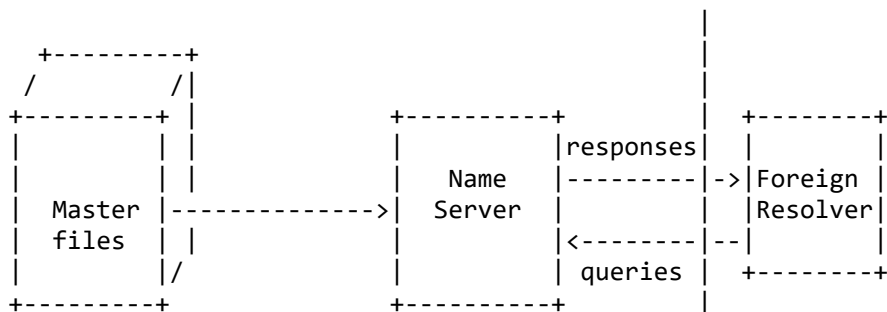
Note that the resolver may have to make several queries to several different foreign name servers to answer a particular user query, and hence the resolution of a user query may involve several network accesses and an arbitrary amount of time. The queries to foreign name servers and the corresponding 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:

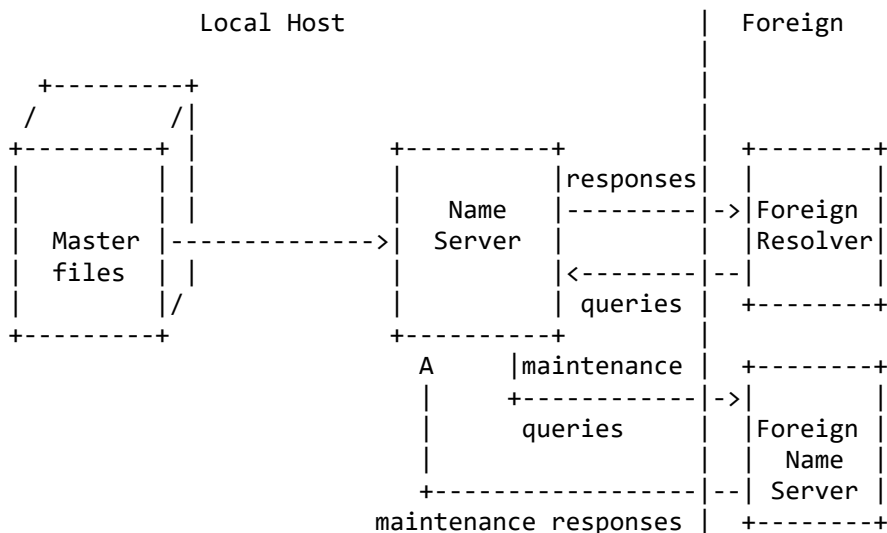
Local Host

| Foreign



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.

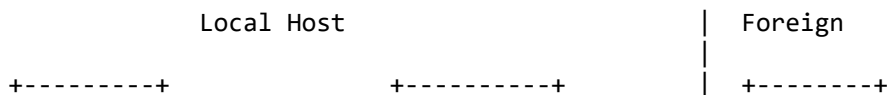
The DNS requires that all zones be redundantly supported by more than one name server. Designated secondary servers can acquire zones and check for updates from the primary server using the zone transfer protocol of the DNS. This configuration is shown below:



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:



Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.