

of messages, assembly and disassembly of messages is necessary. Further, assembly or disassembly of messages would be performed in accordance with how such messages are defined as noted in element **[B]**.

Regarding a comparison being performed, EMV '96 discloses that an error can be detected during communication involving the system or the terminal, which would require some form of comparison to a stored definition of how an acceptable message should be formatted.

The First Data '879 Patent teaches the use of a distinct communication software module, referred to as a “communication processor” [the claimed “virtual message processor”] handling messaging functions with an “execution control processor” software module [the claimed “virtual function processor”]. The First Data '879 Patent explicitly teaches how distinct software modules can be used to handle communication processing separately from other processing.

OMNI 300 discloses that a service call invoked by an application can initiate a data transfer. Regarding a message comparison, OMNI 300 performs a check for corrupt data received in messages by comparing a received cyclic redundancy check (CRC) value with calculated CRC value to determine if a match is present. Two bytes in the header of messages exchanged via a network are used for the comparison of CRC values, as indicated in the first table of element **[B]**.

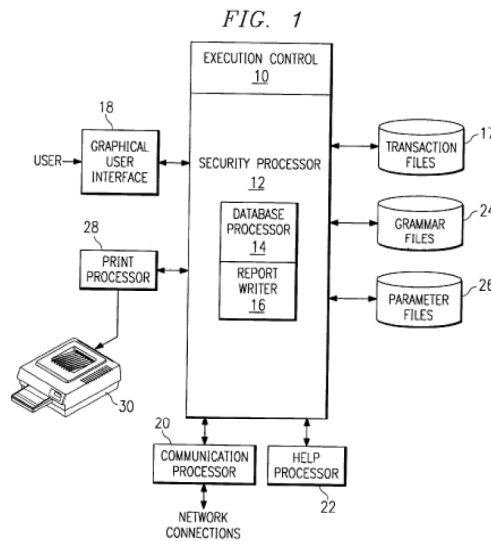
[C] a virtual message processor, which is arranged to be called by the function processor and which is arranged to carry out the message handling tasks of assembling the messages, disassembling messages and comparing the messages under the direction of the message instruction means that is arranged to provide directions for operation of the virtual message processor,

EMV '96

“An authorisation message shall be used when transactions are batch data captured. A financial transaction message shall be used when online data capture is performed by the acquirer.” *EMV '96*, at §2.1, p. III-6. “The terminal shall be able to support at least one or more Issuer Scripts in each authorization or financial transaction response it receives” *Id.*, at §2.2.9, p. I-10. Further, “The terminal shall transmit the Issuer Script Results in the batch data capture message” *Id.*

“ ‘0F’ - PROCESSING ERROR - Displayed to the cardholder or attendant when the card is removed before the processing of a transaction is complete or when the transaction is aborted because of a power failure, or the system or terminal has malfunctioned, such as communication errors or time-outs.” *Id.*, at pp. III-3, III-4.

The First Data '879 Patent



First Data, Fig. 1. “It should be understood that the term processor used herein refers to a software module operating to perform a particular task or group of tasks. A single such module may

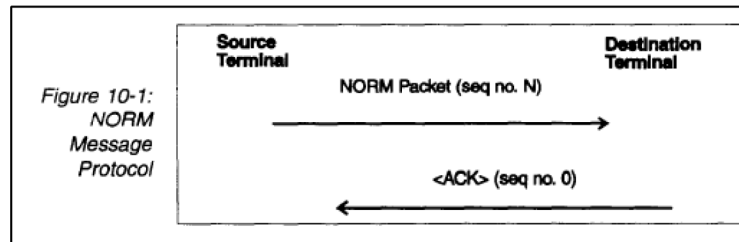
actually be running on a variety of hardware architectures which could include single or multiple hardware processors.” *First Data '879 Patent*, at col. 2, ll. 58-63.

“The execution control processor 10 is also coupled to a communications processor 20.” *Id.*, at col. 4, ll. 5, 6.

OMNI 300

An application causes communication to occur: “Terminal applications may be programmed to initiated ZONTALK 2000 downloads (full or partial) by using the following service call: result – SVC_ZONTALK(x)” *OMNI 300*, p. 2-4.

Messages are both assembled for transmission and disassembled following being received:



Id., at p. 10-8. Such messages are constructed

and deconstructed in accordance with “struct packet_header” as detailed in element [B].

“Corrupt Data – If the destination terminal receives a data packet whose CRC-16 entry does not match the terminal’s CRC calculation, it sends a NAK to the sending terminal.” *Id.*, at p. 10-10.

Element D. This element of the ‘945 Patent is directed to calling a message processor to handle a message. Each of EMV ’96 and the First Data ‘879 patent disclose a software module being called to handle a message. Further, OMNI 300

specifies how an application can call on a communication software module to handle a message exchange such that a download can occur. In OMNI 300, an application (which could be executing as a first software module) can make a call to start a download (which is handled by a separate software module).

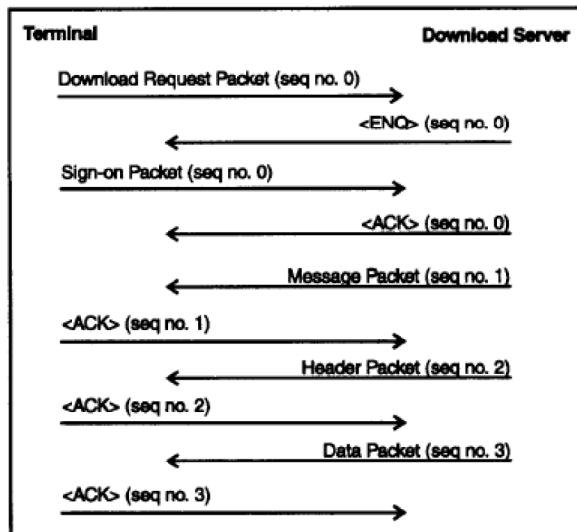
<p>[D] whereby when a message is required to be handled by the communications device the message processor is called to carry out the message handling task,</p>	<p><u>EMV '96</u> “If the card indicates to process online, the terminal shall transmit an authorization or financial transaction request message, if capable.” <i>EMV '96</i>, at §2.2.7, p. I-10.</p> <p>“The terminal shall be able to recognize the tag for the Issuer Script transmitted in the response message. If the tag is ‘71’, the terminal shall process the script before issuing the second GENERATE AC command.” <i>EMV '96</i>, at §2.2.9, p. I-11.</p> <p><u>The First Data '879 Patent</u> “The Execution control processor 10 is also coupled to a communications processor 20. The communications processor 20 allows the integrated system to communicate with other system through network connections. According to one embodiment of the present invention, the communications processor 20 allows for communication with an integrated communications platform system to allow for session-based communications with a host computer.” <i>First Data</i>, col. 4, ll. 5-8. “The communications processor 20 acts as an interface between the integrated systems and whatever communication facilities are available via network connections.” <i>Id.</i>, at col. 4, ll. 32-35.</p> <p>“The system of the present invention then uses this serial number to access the data files of the host computer through the communications processor 20</p>
---	--

and network connections as described in step 116.”
Id., at col. 14, ll. 4-7.

OMNI 300

The application may initiate a LAN download:
“OMNI 300 Series LAN terminals support application download via the LAN port. Prior to beginning the download, each terminal must have a particular set of parameters present in its CONFIG.SYS file.” *OMNI 300*, p. 2-4. “Terminal applications may be programmed to initiate ZONTALK 200 downloads (full or partial) by using the following service call: result=SVC_ZONTALK(x); . . .” *Id.*

The exchange initiated by such a request may be represented as:



Id., at p.

10-36.

Element E. This element of the ‘945 Patent requires that the implemented virtual machine be emulatable on different computers (e.g., different POS devices)

having different, incompatible hardware platforms or different, incompatible operating systems. EMV '96 and the First Data '879 Patent disclose such an ability to be emulatable on different hardware platforms and/or different operating systems. Further, OMNI 300 indicates how different dialects of the C programming language can be used to allow for compatibility across varying systems. OTA indicates how application programs for OTA terminals can be “completely platform independent.”

<p>[E] wherein the virtual machine means is emulatable in different computers having incompatible hardwares or operating systems.</p>	<p><u>EMV '96</u> “The kernel for each particular CPU type is written to make that processor emulate the virtual machine. The virtual machine concept makes a high degree of standardisation possible across widely varying CPU types and simplifies program portability, testing, and certification issues.” <i>EMV '96</i>, at ¶1.4.4, p. II-5.</p> <p>Also, see element [A] re: <i>EMV '96</i>, §1.4.1, pp. II-3–II-4.</p> <p><u>The First Data '879 Patent</u> “The financial instrument processing system of the present invention comprises an object-oriented software system that is highly portable between various hardware platforms. The architecture of the integrated software system is constructed such that the system can be easily and conveniently ported to a variety of operating system such as MS DOS, Windows, OS2, or UNIX.” <i>First Data '879 Patent</i>, col. 2, ll. 43-49.</p> <p><u>OMNI 300</u> “VeriFone supports the Standard ANSI C and a</p>
--	---

	<p>UNIX-V7 compatible dialect of the C language (non-ANSI) for TXO application development.” <i>OMNI 300</i>, p. 3-1.</p> <p>“VeriFone has maintained programming compatibility to enable you to port application source code from one terminal platform to another.” p. F-1, Table F-1, p. F-2 and p. F-7</p> <p><u>OTA</u> “Using the ‘Open Terminal Architecture’ (OTA) it will be possible for credit card issuers and acquirers to write application programs that will be completely platform independent, and run on all OTA-compliant kernels.” <i>OTA</i>, p. 73, ¶1.</p>
--	--

Claim 2. Claim 2 of the ‘945 Patent discloses that a virtual protocol processor is used to organize communications. The protocol processor has instructions which direct the operation of the protocol processor. The ‘945 Patent describes the “protocol processor” as follows: “The protocol processor means is preferably a program module the specific function of which is to control and select the sequence of message processor operations in relation to messages received and transmitted.” Col. 4, ll. 27-30. “The protocol instructions are divided into ‘sections’ 130, ‘lines’ 131 and ‘protocol commands’ 132, as illustrated in FIG. 12A. FIG. 12B illustrates how an instruction is displayed on a development tool for protocol instructions. Protocol instructions describe message flow both from

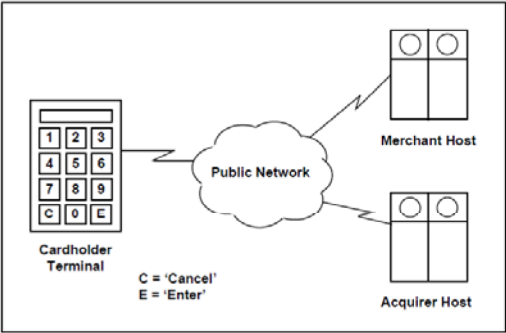
and to the device. The top line specifies outgoing messages and the other lines display possible incoming results.” Col. 15, l 63 – Col. 16, l. 2.

The control and sequencing of messages is shown in Omni 300, such as the example chart in Claim 1, element **[D]** above. OMNI 300’s exchange of messages in claim 1 element **[D]** indicates how the terminal responds to a sequence of messages, the messages and acknowledgments having distinct sequence numbers. As detailed below, control is exercised over packets by OMNI 300 indicating that transmitted messages are sequenced and the sequencing of received packets is monitored to ensure that packets are not processed if an invalid sequence number is identified. Whether the instructions for controlling and sequencing messages (“protocol processor”) and the instructions for assembling and disassembling messages (“message processor”) are in the same module (virtual processor) or not is a simple matter of design choice. The ‘945 Patent describes the advantage of being in native code, and that is shown in the claim 1 chart above.

EMV ’96 involves a terminal communicating via a network, thereby necessarily requiring the network’s protocol (sequence of messages) to be handled by the connected terminal. It is inherent that protocol handling by the terminal rely on some form of instructions in order for communications to be organized in accordance with the protocol. OMNI 300 clarifies that sequencing and control of messages (e.g., ignoring messages based on a sequence number) is handled by the

terminal. Each message is assigned a sequence number, which identifies the proper order of data packets. Such ordering constitutes organization of communications.

Further, the First Data '879 Patent discloses that the communication processor, which is a software module, communicates with other systems via a network connection. Such network-based communication requires that the appropriate network protocols be enforced by the device. Also, OTA indicates that the handling of a protocol occurs at the target system (the POS device). This handling of the protocol is performed by instructions stored in a ROM of the terminal. The loading of code into the terminal's ROM allows for a protocol of the interactive development link to be handled.

<p>The '945 Patent</p>	
<p>2. A device in accordance with claim 1, further comprising a virtual protocol processor arranged to organize communications to and from the device, and protocol processor instruction means arranged to provide directions for operation of the protocol processor</p>	<p><u>EMV '96</u></p>  <p><i>EMV '96, p. I-5.</i></p> <p><u>The First Data '879 patent</u> “The communications processor 20 allows the integrated system to communicate with other systems through network connections. According to one</p>

means.

embodiment of the present invention, the communications processor 20 allows for communication with an integrated communications platform system to allow for session-based communications with a host computer.” *The First Data ‘879 patent*, p. 4, ll. 6-8.

OMNI 300

Sequence Number—Bytes #9–12

Identifies order of data packets from sending terminal.

Defined as unsigned long tx_seq;

Range of values: 0 - 0xFFFFFFFF

See *Sequence Numbers* later in this section.

The application may write to the tx_seq field if the type field is DOWNLOAD or dest_addr is BROADCAST. Otherwise, the operating system controls the sequence field and overwrites any value set by the application. If an application-generated BROADCAST message is the first message after LAN initialization, the application must manually set the sequence number to 1.

OMNI 300, p.

10-5, 10-6.

“Generally, if a terminal receives more than one packet with the same sequence number, the second packet and all subsequent packets with the same sequence number are ACKed but are thrown away.” *Id.*, at p. 10-11.

“The firmware checks sequence numbers of all packets in the download protocol. Any packet with an invalid sequence number is ignored by the firmware, and is not passed up to the application. In all protocols, the sequence numbers of packets prior to and including the ENQ packet is zero. After the ENQ packet, the sequence numbers are incremented after every ACK is sent or received. The sequence numbers of all ACKs must match the sequence numbers of the packets they are ACKing.” *Id.*, at p. 10-48.

OTA

“An OTA development environment for a terminal small program to support the terminal end of the Interactive Development Link (IDL) protocol is usually placed in the [terminal’s ROM]”. *OTA*, p. 73, Fig. 1.

Claim 3. Claim 3 of the Ogilvy '945 patent requires that the protocol processor run in the microprocessor's native code. EMV '96 indicates that virtual processor instructions can be implemented as the actual code for the target CPU. Such recitations would be obvious to combine with OMNI 300, which discloses sequencing and control of messages (e.g., ignoring messages based on a sequence number) is handled by the terminal. Therefore, handling of protocols could be performed in the native code of the processor on which the "protocol processor" is implemented. OTA discloses that instructions to handle a protocol may be loaded to the terminal's ROM and OTA further indicates that a virtual machine may be implemented as native code.

<p>The '945 Patent</p>	
<p>3. A device in accordance with claim 2, wherein the device includes a microprocessor which runs in accordance with native software code and the protocol processor is implemented as a native software code of the microprocessor.</p>	<p><u>EMV '96</u> "Virtual machine emulation may be accomplished by one of three methods: interpreting virtual machine instructions, translating the virtual machine language into a directly executable 'threaded code' form, or translating it into actual code for the target CPU." <i>EMV '96</i>, at §1.4.4, p. II-5.</p> <p><u>Omni 300</u> See claim 2, re: <i>OMNI 300</i>.</p> <p><u>OTA</u> See claim 1, element [A] and claim 2, re: <i>OTA</i></p>

Claim 4. This claim of the ‘945 Patent requires that the protocol instruction means does not require translation to the native code of the microprocessor. EMV ‘96 discloses several options for implementation of a virtual machine, including implementing as the actual code discussed with respect to claim 3, and that virtual machine emulation may involve interpreting virtual machine instructions. Such recitations would be obvious to combine with OMNI 300, which discloses control and sequencing of messages being managed by the terminal. Implementing the protocol processing of OMNI 300 using a virtual machine implementation of EMV ‘96 that does not require translation would have been obvious. Further, OTA discloses that instructions to handle a protocol may be loaded to the terminal’s ROM and OTA further indicates that implementation of a virtual machine may not require translation in a processor’s native code.

<p>The ‘945 Patent</p>	
<p>4. A device in accordance with claim 2, wherein the device includes a microprocessor which runs in accordance with native software code and wherein the protocol instruction means are implemented in software defined by the protocol processor means, and do not require translation to the native code of the microprocessor.</p>	<p><u>EMV ‘96</u> “Virtual machine emulation may be accomplished by one of three methods: interpreting virtual machine instructions, translating the virtual machine language into a directly executable ‘threaded code’ form, or translating it into actual code for the target CPU.” <i>EMV ‘96</i>, at §1.4.4, p. II-5.</p> <p><u>Omni 300</u> See claim 2, re: <i>OMNI 300</i>.</p>

	<p><u>OTA</u> See claim 1, element [A] and claim 2, re: <i>OTA</i></p>
--	---

Claim 5. This claim of the '945 Patent requires that the message processor be implemented in the native code of the microprocessor. EMV '96 and OTA disclose that virtual machine emulation may be based on implementation in the actual code for CPU. Further, the communication processor of the First Data '879 Patent, is implemented as a software module that is executed by an underlying hardware processor, and therefore is in the native code of the microprocessor.

The '945 Patent	
<p>5. A device in accordance with claim 1, wherein the device includes a microprocessor which runs in accordance with native software code, and the message processor is implemented as the native software code of the microprocessor.</p>	<p><u>EMV '96</u> See claim 3 re: <i>EMV '96</i>, at §1.4.4, p. II-5.</p> <p>See claim 1, element [C].</p> <p><u>OTA</u> See claim 1, element [A].</p>

Claim 6. Claim 6 of the '945 Patent requires that the software functioning as the function processor be in the native code of the microprocessor. EMV '96 and OTA disclose that a kernel that allows for the virtual machine function on various CPU types be written for the type of CPU on which it will be implemented.

Further, EMV '96 and OTA explicitly state that the virtual machine instructions can be translated into instructions for the processor's native code.

The '945 Patent	
<p>6. A device in accordance with claim 5, wherein the function processor is implemented as native code of the microprocessor.</p>	<p><u>EMV '96</u> "Virtual machine emulation may be accomplished by one of three methods: interpreting virtual machine instructions, translating the virtual machine language into a directly executable 'threaded code' form, or translating it into actual code for the target CPU." <i>EMV '96</i>, at §1.4.4, p. II-5.</p> <p>See claim 1, element [E], <i>EMV '96</i>.</p> <p><u>OTA</u> "A kernel contains all functions whose implementation depends upon a particular platform (CPU and OS)." <i>OTA</i>, p. 74.</p> <p>See claim 1, element [A] re: OTA.</p>

Claim 7. Claim 7 of the '945 Patent requires that the message processor need not be translated into the native software code of the microprocessor. EMV '96 explicitly states that the virtual machine instructions do not need to be translated into native instructions for the microprocessor. When combined with the First Data '879 Patent, which discloses a separate communication processor module being used, it is obvious that a message processor could be implemented in the native software code of a microprocessor.

The '945 Patent	
------------------------	--

<p>7. A device in accordance with claim 1, wherein the message processor instruction means is implemented in software defined by the message processor, wherein the device includes a microprocessor, and wherein the message instruction means do not require translation to the native software code of the microprocessor.</p>	<p><u>EMV '96</u> “Virtual machine emulation may be accomplished by one of three methods: interpreting virtual machine instructions, translating the virtual machine language into a directly executable ‘threaded code’ form, or translating it into actual code for the target CPU.” <i>EMV '96</i>, at §1.4.4, p. II-5.</p> <p><u>First Data '879 Patent</u> See claim 1, element [C], re: the <i>First Data '879 Patent</i>.</p>
---	--

Claim 8. Claim 8 of the ‘945 Patent requires that the function processor need not be translated into the native software code of the microprocessor. EMV ‘96 explicitly states that the virtual machine instructions does not need to be translated into native instructions for the microprocessor. The virtual machines of EMV ‘96 and OTA allow for code that is interpreted by a virtual machine (i.e., written in code for the virtual machine rather than the processor itself). The virtual machine, which executes on the CPU, performs the instructions.

<p>The ‘945 Patent</p>	
<p>8. A device in accordance with claim 1, wherein the device includes a microprocessor which runs in accordance with native software code, and wherein the function processor instruction means are implemented in software defined by the function</p>	<p><u>EMV '96</u> “Virtual machine emulation may be accomplished by one of three methods: interpreting virtual machine instructions, translating the virtual machine language into a directly executable ‘threaded code’ form, or</p>

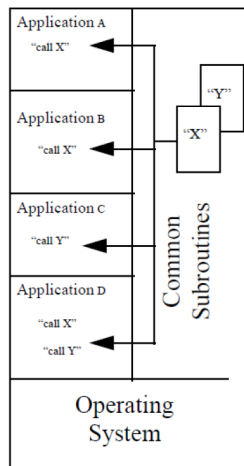
<p>processor means and do not require translation to the native code of the microprocessor.</p>	<p>translating it into actual code for the target CPU.” <i>EMV '96</i>, at §1.4.4, p. II-5.</p> <p>See claim 1, element [A], <i>EMV '96</i>.</p> <p><u>OTA</u> See claim 1, element [A] re: OTA.</p>
---	---

Claim 9. Claim 9 of the '945 Patent requires that a hardware abstraction layer be present that includes a series of routines which provide an API to exercise an operating system, BIOS, or hardware drivers of the device. *EMV '96* specifies that device drivers are included as part of the kernel to allow a virtual machine to be implemented on a specific real machine. *EMV '96* also discloses that an operating system that interacts with a variety of routines that can be called by applications. *OMNI 300*, such as in relation to claim 2, discloses the use of various abstraction layers in relation to hardware as part of the OSI model. Also, *OTA* discloses the concept of a kernel which serves as the interface between the virtual machine and the specific operating system and hardware of the terminal.

<p>The '945 Patent</p>	
-------------------------------	--

9. A device in accordance with claim 1, including a hardware abstraction layer comprising a series of routines which provide an application program interface to exercise an operating system, BIOS or hardware drivers of the device.

EMV '96



“The kernel contains device drivers, interface routines, security and control functions, and the software for translating from the virtual machine language to the language used by the real machine. In other words, the kernel is the implementation of the virtual machine on a specific real machine.” *Id.*, at §3, p. X.

Figure II-1 - Terminal Software

EMV '96, §1.2, p. II-2.

OMNI 300

See claim 1, element [D] and claim 2.

OTA

“A kernel contains all functions whose implementation depends upon a particular platform (CPU and OS). It includes a selected subset of ANS Forth words, plus a number of specialized OTA functions such as terminal I/O support, token loader/interpreter support, and operations designed to support the particular needs of payment programs.” *OTA*, p. 74, ¶7.

Claims 10 and 11. Claims 10 and 11 of Ogilvy focus on the type of device claimed. Claim 10 focuses on a device that is a network access device and claim 11 focuses on a remote payment terminal. A point of service device, as presented

in relation EMV '96, is a network access device that communicates with a host to provide financial services, such as purchase transactions. In addition, both OMNI 300 and The First Data '879 Patent disclose network access devices and, more specifically, remote payment terminals that communicate with a host system. OTA discloses the use of credit card terminals, which require communication via a network with a host system in order to perform credit card transactions, such as 8051-based POS terminals.

<p>The '945 Patent</p>	
<p>10. A device in accordance with claim 1, wherein the device is a specialized network access device arranged for communication over a network.</p>	<p><u>EMV '96</u> See claim 1, Preamble, <i>EMV '96</i>.</p> <p><u>OMNI 300</u> “The OMNI™ 300 Series is a family of dial-type and LAN-type transaction automation systems These terminals are ideal for a multitude of applications, including: Point of Sale/Service (POS). <i>OMNI 300</i>, p. 1-1.</p> <p><u>First Data '879 Patent</u> “According to another embodiment of the present invention, a communications processor is included and is operable to communicate with the data base processing system to provide information stored in the data base processing system to other systems such as host accounting systems.” <i>First Data '879 Patent</i>, col. 1, l. 66 – col. 2, l. 4.</p> <p><u>OTA</u> “The purpose of an OTA system</p>

	is to provide software to run in terminals used in payment applications.” <i>OTA</i> , p. 73, ¶3. “This will require new software in all credit card terminals (ranging from 8051-based POS terminals to high-end ATMs). <i>OTA</i> , p. 73.
11. A device in accordance with claim 10, the device being a remote payment terminal and the messages being messages relating to remote payment transactions.	See claim 10.

Claim 12. Independent claim 12 of Ogilvy is a method claim that is substantially similar to claim 1 in scope.

The ‘945 Patent	
12. A method of programming a device for processing communications, comprising the steps of	See claim 1, Preamble.
loading a processing means of the device with a virtual machine which includes a virtual function processor and function processor instructions for controlling operation of the device,	See claim 1, element [A] .
and a virtual message processor which is arranged to be called by the functions processor and which is arranged to carry out the task of assembling, disassembling and comparing messages, under the direction of the message instruction means that is arranged to provide directions for operation of the virtual message processor,	See claim 1, elements [B] and [C] .
whereby when a message is required to be handled by the	See claim 1, element [D] .

communications device the message processor is called to carry out the message handling task,	
wherein the virtual machine means is emulatable in different computers having incompatible hardwares or operating systems.	See claim 1, element [E].

Claim 13. This claim of the '945 Patent requires that processor of claim 12 be loaded with a virtual protocol processor to handle organization of communications to and from the device, with protocol processor instructions controlling the protocol processor. Both EMV '96 and OMNI 300 discloses that software can be loaded to their respective devices. Further, by definition, software that is implemented by a processor must at some point be loaded onto the processor; otherwise the processor would have no way of having the software available. Moreover, OTA discloses that programs (instructions) can be downloaded to a terminal.

The '945 Patent	
13. A method in accordance with claim 12, comprising the further step of loading the processor means of the device with a virtual protocol processor arranged to organize communications to and from the device, and protocol processor instructions arranged to provide directions for operation of the protocol processor.	<p><u>EMV '96</u></p> <p>“A means of software upgrade shall be supported wherever this is not in conflict with national legal restrictions. The software upgrade may be facilitated from a remote site over a network or locally.” <i>EMV '96</i>, p. II-7.</p> <p>“A means for updating data elements specific to payment system</p>

	<p>applications shall be supported wherever this is not in conflict with national legal restrictions. Data update may be facilitated from a remote site over a network or locally.” <i>Id.</i>, at p. II-11.</p> <p style="text-align: center;"><u>OMNI 300</u> “OMNI 300 Series firmware support several methods of downloading application, data and configuration files to the terminal.” <i>OMNI 300</i>, p. 2-3.</p> <p style="text-align: center;">See claim 1, element [D], and claim 2 re: <i>OMNI 300</i>.</p> <p style="text-align: center;"><u>OTA</u> “For final testing using the terminal’s own kernel and I/O, these programs would be tokenized . . . and downloaded to the target terminal.” <i>OTA</i>, p. 74, ¶1.</p>
--	--

Claim 14. Independent claim 14 is substantially similar to claim 1 in scope. While claim 1 is directed to a device, the instructions of EMV ’96, The First Data ‘879 Patent, and OMNI 300 are implemented as software, thus a computer memory is used to store such instructions for execution as indicated in claim 14.

The ‘945 Patent	
14. A computer memory storing instructions for controlling a computing device to implement a virtual machine means	See claim 1, element [A].
which includes a virtual function processor and function processor	See claim 1, element [A].

instructions for controlling operation of the device,	
and a virtual message processor which is arranged to be called by the function processor and which is arranged to carry out the task of assembling, disassembling and comparing messages, under the direction of the message instruction means that is arranged to provide directions for operation of the virtual message processor,	See claim 1, elements [B] and [C] .
whereby when a message is required to be handled by the communications device the message processor is called to carry out the message handling task,	See claim 1, element [D] .
wherein the virtual machine means is emulatable in different computers having incompatible hardwares or operating systems.	See claim 1, element [E] .

Claim 15. Claim 15 of the '945 Patent is substantially similar to recitations previously discussed in relation to claim 1, element **[B]**.

The '945 Patent	
15. A computer readable memory in accordance with claim 14, further storing instructions for implementing message processor instruction means arranged to provide directions for operation of the message processor.	See claim 1, element [B] .

Claims 16 and 17. Claim 16 and 17 of the ‘945 Patent are directed to implementation of a virtual protocol processor. Similar to as detailed in relation to claim 2, OMNI 300 discloses that the control and sequencing of messages is shown in Omni 300, such as the example chart in Claim 1, element [D]. Whether the instructions for controlling and sequencing messages (“protocol processor”) and the instructions for assembling and disassembling messages (“message processor”) are in the same module (virtual processor) or not is a simple matter of design choice. The example in claim 1 element [D] indicates how the terminal responds to a sequence of messages, the messages and acknowledgments having distinct sequence numbers. Further, control is exercised over packets by OMNI 300 indicating that transmitted messages are sequenced and the sequencing of received packets is monitored to ensure that packets are not processed if an invalid sequence number is identified.

The ‘945 Patent	
16. A computer readable memory in accordance with claim 14, further storing instructions for implementing a virtual protocol processor arranged to organize communications to and from the computing device.	See claim 2.
17. A computer readable memory in accordance with claim 16, further storing instructions for implementing protocol processor instructions arranged to provide directions for operation of the protocol processor means.	See claim 16.

66242276V.1