Network and Services
Integration Forum

NSIF-033-1999

(NSIF Document #SIF-AR-9806-088R11)

NSIF APPROVED DOCUMENT

WORK GROUP: **ARCHITECTURE**

TITLE: **Requirements for the TCP/IP Protocol Suite on the SONET Access DCN**

DATE: **December 19, 1999**

EDITOR: **Name:** **Brian Crowe**
Voice: **(972) 495-1282**
email: **brian_crowe@i-worx.com**

ABSTRACT: This document presents a set of high level specifications which are intended to form the basis for interoperable SONET Management Communications Networks which contain both IP-based Data Communications Networks (DCN) and CLNP-based DCNs. The purpose of the specifications is to promote the development of interoperable OSs, SONET Gateway Network Elements (GNE), and SONET Network Elements (NE) which use the IP and CLNP-based DCNs for management communications. In addition, the document contains generic specifications for the required protocols, tools, and procedures needed to structure and to administer the IP-based Network.

This document has received the approval of the Network Services and Integration Forum (NSIF).

Table Of Contents

List of Figures

1 Introduction

1.1 Background

Beginning in February 1998, SIF members identified a business need for using the TCP/IP protocol suite to support SONET management communications. This need was documented in Contributions SIF-AR-9802-005 and SIF-AR-9804-055 and discussed at a SIF conference call on May 13, 1998. During the regularly scheduled June 1998 SIF meeting, contributions on this topic were reviewed. The contributions reviewed included SIF-AR-9806-086 and SIF-AR-9806-088R1. At the conclusion of the meeting it was decided that there was sufficient member interest and a sufficient base of contributions to begin work on a SIF Draft Document on Specifications for TCP/IP protocol suite on the SONET access DCN.

1.2 Objective/Purpose

The objectives of this document are to:

1. Present specifications which can form the basis for interoperable SONET Management Communications Networks that contain both IP-based Data Communications Networks (DCN) and CLNP-based DCNs. These specifications are intended to promote the development of interoperable OSs, SONET Gateway Network Elements (GNE), and SONET Network Elements (NE) which use the IP and CLNP-based DCNs for management communications.
2. Identify well defined reference points at which the specifications apply.
3. Insure multi-supplier interoperability.
4. Present generic specifications for the required protocols, tools, and procedures needed to structure and to administer the IP-based Network in order to encourage innovative and cost effective implementations.

1.3 Scope

The following issues are included in this document. In some cases, only the initial set of specifications is provided.

1. Requirements for use of TL1 based interactive services and the use of FTP for file transfer services on the IP-based network. These specifications do not address the functionality and protocols used by the SONET Section Data Communication Channel (DCC), nor do they require changes in that functionality or in those protocols. These specifications are intended to address a significant near term service provider need and are intended to be consistent with other specifications contained in this document.

2. The scope of this document is limited to implementations based on Version 4 of IP. Use of future versions of IP is left for future study.

2 References

1. GR-253-CORE, *Synchronous Optical Network (SONET) Transport Systems: Common Generic Criteria*, Bellcore, Issue 2, Revision 2, January, 1999.
2. GR-815-CORE, *Generic Requirements for Network Element/Network System (NE/NS) Security, Bellcore*, Issue 1, November 1997.
3. GR-831-CORE, *Operations Application Messages – Language for Operations Application Messages*, Issue 1, November 1996.
4. GR-833-CORE, *Network Maintenance: Network Element and Transport Surveillance Messages*, Bellcore, Issue 2, November 1996.
5. TR-NWT-000835, *Operations Application Messages - Network Element and Network System Security Administration Messages*", Telcordia Operations Technology Generic Requirements, Bellcore, Issue 3, January 1993.
6. GR-1250-CORE, *Generic Requirements For Synchronous Optical Network (SONET) File Transfer*, Bellcore, Issue 1, June 1995.
7. GR-1332-CORE, *Generic Requirements for Data Communications Network Security*, Bellcore, Issue 2, April 1996.
8. M.3010, *Principles For A Telecommunications Management Network,* ITU-T Recommendation, 5/96.
9. Q.811, *Lower Layer Protocol Profiles For The Q3 And X Interfaces*, ITU-T, 6/97.
10. Q.812, *Upper Layer Protocol Profiles For The Q3 And X Interfaces*, ITU-T, 6/97.
11. ISO/IEC 8571-4 (1988), *Information processing systems - Open Systems Interconnection - File Transfer, Access and Management - Part 4: File Protocol Specification.*
12. 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*.
13. ISO/IEC 8802-3 (1996), *Information technology--Telecommunications and information exchange between systems--Local and metropolitan area networks--Specific requirements--Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*.
14. X.207, *Application Layer Structure*, ITU-T, 11/93.
15. RFC 768, *User Datagram Protocol*, J. Postel (ISI), August 1980.

16. RFC-791, *Internet Protocol*, September 1981.

17. RFC-792, *Internet Control Message Protocol*, J. Postel (ISI) Editor, September 1981.

18. RFC-793, *Transmission Control Protocol*, September 1981.

19. RFC-854, *TELNET Protocol Specification*, J. Postel and J. Reynolds (ISI) Editors, May 1983.

20. RFC-855, *TELNET Options Specification*, J. Postel and J. Reynolds (ISI) Editors, May 1983.

21. RFC-894, A Standard for the Transmission of IP Datagrams over Ethernet Networks, C. Hornig (Symbolics CRC) Editor, April, 1984.

22. RFC-919, *Broadcasting Internet Datagrams*, Jeffrey Mogul (Stanford) Editor, October 1984.

23. RFC-922, *Broadcasting Internet Datagrams in the Presence of Subnets*, Jeffrey Mogul (Stanford) Editor, October 1984.

24. RFC-950, *Internet Standard Subnetting Procedure*, J. Mogul (Stanford) and J. Postel (ISI) Editors, August 1985.

25. RFC-959, File Transfer Protocol (FTP), J. Postel and J. Reynolds (ISI) Editors, October 1985.

26. RFC-1034, *Domain Names - Concepts and Facilities*, P. Mockapetris (ISI) Editor, November 1987.

27. RFC-1035, *Domain Names - Implementation and Specification*, P. Mockapetris (ISI) Editor, November 1987.

28. RFC-1042, *A Standard for the Transmission of IP Datagrams over IEEE 802 Networks*, J. Postel and J. Reynolds Editors, February 1988.

29. RFC-1112, *Host Extensions for IP Multicasting*, S. Deering (Stanford) Editor, August 1989.

30. RFC-1122, *Requirements for Internet Hosts -- Communication Layers*. R. Braden, Editor, October 1989.

31. RFC-1123, *Requirements for Internet Hosts -- Application and Support*, R. Braden, Editor, October 1989.

32. RFC-1155, *Structure and Identification of Management Information for TCP/IP-based Internets*, M. Rose (PSI) and K. McCloghrie (Hughes) Editors, May, 1990.

33. RFC-1157, *A Simple Network Management Protocol (SNMP)*, J. Case (SNMP Research), M. Fedor (PSI), M. Schoffstall (PSI), and J. Davin (MIT) Editors, May, 1990.

34. RFC-1212, *Concise MIB Definitions*, M. Rose (PSI) and K. McCloghrie (Hughes) Editors, March, 1991.

35. RFC 1213, *Management Information Base For Network Management Of TCP/IP-Based Internets: MIB-II*, M. Rose (PSI) and K. McCloghrie (Hughes) Editors, March, 1991.

This document has received the approval of the Network Services and Integration Forum (NSIF).

36. RFC-1661, *The Point-to-Point Protocol (PPP),* W. Simpson (Daydreamer) Editor, July, 1994.

37. RCF-1662, *PPP in HDLC-like Framing*, W. Simpson (Daydreamer) Editor, July, 1994.

38. RFC-1738, *Uniform Resource Locators (URL),* T. Berners-Lee (CERN), L. Masinter (XEROX), M. McCahill (U. of Minnesota) Editors, December, 1994.

39. RFC-2500, *Internet Official Protocol Standards*, J. Reynolds and R. Braden Editors, June 1999.

40. .SIF-011-1997, *FTAM Profile for SONET Operations Communications, 1997.*

41. SIF-018-1998, Guidelines for SONET DCN Architecture Engineering, 1998.

42. SIF-022-1999, *SIF SONET TMN Architecture Requirements, 1999.*

43. Stevens, W. Richard, "*UNIX Network Programming, 2nd edition.*", Prentice Hall, 1998.

3 Definitions and Acronyms

3.1 Definitions

This section contains definitions for terms that have specialized meanings in this document. As a convenience to the reader, it also contains definitions of selected technical terms defined in other documents. In the latter case, the source of the definition is also provided.

CLNP-Based Data Communication Network[1] **(DCN)**: The CLNP-Based DCN is a communication network within a TMN that supports the Data Communication Function (DCF). The CLNP-Based DCN represents an implementation of the OSI layers 1 to 3, which include any relevant ITU-T or ISO standards for layers 1 to 3. The CLNP-Based DCN provides no functionality at layers 4 to 7.

Connection is used to refer to a TCP connection or OSI presentation association.

Gateway Network Element (GNE): A Gateway Network Element is the NE in a collection of interconnected NEs (such as a UPSR or BLSR rings) that connects the collection of NEs to the management systems.

Half Open Connection[2]**:** An established TCP connection on which one of the sides has closed or aborted the connection without the knowledge of the other, or on which the two ends of the connection have become desynchronized owing to a crash that resulted in a loss of memory.

[1] ITU-T Recommendation M.3010, p 26.

[2] RFC-793.

Inactive TCP Connection: An established TCP connection over which no data has been transmitted in either direction for a configurable amount of time.

IP-Based Data Communication Network: The IP-Based DCN is a communications network which supports the Data Communication Function. The IP-Based DCN represents an implementation of the Internet physical, data link and network layers, and associated protocols. The IP-Based DCN provides no functionality above the Internet network layer. For convenience, the IP-based DCN is limited to mean the access component of the SONET DCN, and does not include the DCC.

Management Communications: The communications required to allow the exchange of management information to assist telecommunications service providers in conducting their business efficiently.

OS: OS refers to a generic management system and includes CITs.

TL1 Initiator: The TL1 initiator sets up outgoing TL1 connections, sends TL1 input messages (i.e. commands) and receives TL1 output messages (i.e. acknowledgements, responses, and notifications). The TL1 initiator corresponds to the client role in a client-server model. OSs and CITs are typically TL1 initiators.

TL1 Responder: The TL1 responder accepts incoming TL1 connections, sends TL1 output messages (i.e. acknowledgements, responses, and notifications) and receives TL1 input messages (i.e. commands). The TL1 responder corresponds to the server role in a client-server model. NEs are typically TL1 responders.

TL1 Translation Device (T-TD): T-TD refers to a device that translates TL1 over TCP to TL1 over OSI presentation. This T-TD may be an NE (which is generally referred to as the GNE) or it may be a separate device. Even if the device is a separate device it need not be dedicated to providing the TL1 over TCP to TL1 over OSI translation function; it may also provide other functions. The T-TD encompasses both the TL1 initiator and TL1 responder aspects.

Translation Function: A generic term for interworking between networks using different lower layer protocols and possible different upper layer protocols. A Translation Function may include application gateway, transport gateway, network layer relay, and/or transport service bridge functionality. See SIF-AR-9701-001R10, *Guidelines For SONET DCN Architecture Engineering*, for definitions of application gateway, transport gateway, network layer relay, and transport service bridge.

3.2 Acronyms

BLSR	Bidirectional Line Switched Ring
CIT	Craft Interface Terminal
CLNP	Connectionless-mode Network Layer Protocol
CMISE	Common Management Information Service Element

DCC	Data Communication Channel
DCF	Data Communication Function
DCN	Data Communications Network
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name Service
FTAM	File Transfer, Access and Management
FTP	File Transfer Protocol
FT-TD	File Transfer Translation Device
GNE	Gateway Network Element
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union - Telecommunications Standardization Sector
LAN	Local Area Network
MD	Mediation Device
NE	Network Element
NSAP	Network Service Access Point
NSIF	Network Services and Integration Forum
NTP	Network Time Protocol
OFS	Operations File Server
OS	Operations System
OSI	Open Systems Interconnection
PSAP	Presentation Service Access Point
RFC	Request For Comments
SIF	SONET Interoperability Forum
SNMP	Simple Network Management Protocol
SNTP	Simple Network Time Protocol
SONET	Synchronous Optical Network
TARP	TID Address Resolution Protocol

TCP	Transmission Control Protocol
TD	Translation Device
TID	Target Identifier
TL1	Transaction Language 1
TMN	Telecommunications Management Network
T-TD	TL1 Translation Device
UDP	User Datagram Protocol
UPSR	Unidirectional Path Switched Ring
WS	Workstation

4 Types Of Specifications

Three types of specifications are used in this document. These are:

1. **Requirements** imply that essentially all service providers would require this capability. Requirements are indicated by **R**.
2. **Conditional Requirements** are requirements which are conditional on other objectives referenced in the conditional requirements. If a referenced objective is met by the implementation, a conditional requirement must also be met. Conditional Requirements are indicated by **CR**.
3. **Objectives** imply that some but not all service providers would require this capability now, and/or that some service providers expect future upgrades will require this capability. Objectives are indicated by **O**.

5 The Reference Architectures

This document uses the following reference architectures specified in SIF approved document SIF-022-1999*, SIF SONET TMN Architecture Requirements*:

1. The TL1/FTP Over TCP/IP Reference Architecture,
2. The CMISE/FTAM Over TCP/IP Reference Architecture,
3. The IP End-To-End Reference Architecture

CMISE/FTAM over TCP/IP and IP End-to-End reference architectures are for further study.

6 TL1/FTP Over TCP/IP

This section is an overview section and as such contains no requirements. Implementation requirements are provided in Section 7.

6.1 The Reference Architecture

As stated in the *SIF SONET TMN Architecture Requirements* document, networks based on the TL1/FTP Over TCP/IP Architecture may not contain all of the elements depicted in the definition. For example, a management network may contain only translation MDs or only translation GNEs, or it may contain both types of devices. For the requirements contained in Section 7.3, the IP-based Network is considered to include the IP-based DCN, the OSs, the MDs, and the GNEs. Similarly, the OSI-based network includes the CLNP-based DCN, MDs, NE, and GNE End Systems, and/or Intermediate Systems. Workstations are outside of the scope of Section 7.

In the remainder of Section 7, the term "translation device" will be used when the reference applies to both the GNE and MD. Figure 6-1 illustrates the translation function required between data communication networks using IP-based routing protocols and CLNP-based routing protocols. This figure illustrates that the management applications may be based on TL1 or FTP/FTAM protocols.

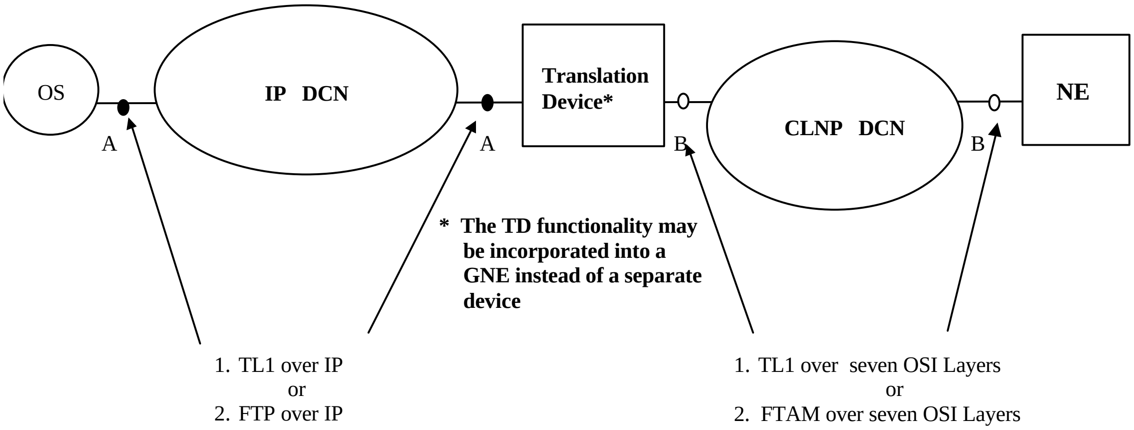

OS
IP DCN
Translation Device*
CLNP DCN
NE
A
A
B
B
* The TD functionality may be incorporated into a GNE instead of a separate device
1. TL1 over IP
or
2. FTP over IP
1. TL1 over seven OSI Layers
or
2. FTAM over seven OSI Layers

Figure 6-1: Protocols at the A & B Reference Points

6.2 Management Context

This section describes the characteristics of SONET NEs and OSs and their associated management support systems from the perspective of the TL1/FTP Reference Architecture. The characteristics described are only those considered relevant to managing the operational aspects of SONET transport systems. These characteristics are intended to represent the inherent properties of these systems and, consequently, are considered to be the basic environment into which proposed solutions to the specifications contained in this document shall apply.

The OSs located in the IP-based network manage the SONET NEs located in the OSI-based network. That is, they perform functions associated with configuration, fault, performance, accounting, and security management. The management interaction between the OSs and the SONET NEs may be as follows:

1. OSs initiate TL1 interactive class communication requests/commands to NEs. These exchanges may be confirmed or unconfirmed.
2. NEs autonomously initiate interactive class notifications to OSs. These exchanges may be confirmed or unconfirmed.
3. OSs initiate and support file transfers to and from NEs. The NEs comply with the file transfer requests and provide indications of the success or failure of these transfers.

File transfer functionality is required to support the management of software and/or data stored on the NEs and to support a remote memory back-up capability for NEs. The file transfer occurs between the IP-based OSs and the CLNP-based NEs. Both the send and the receive functions are supported. It is desirable that these allow a store and forward method of file distribution when a TD has storage space. An interoperable method for transferring files to an NE when the file size exceeds the storage capacity of the TD is also necessary. Furthermore, the use of “store and forward” or “on the fly” file transfer modes must be transparent to the destination (or origination) NE. This implies that the TD must implement a process to determine which file transfer process is used. The NE should not have any knowledge of which file transfer process is being used. Some additional functional requirements for the file transfer process are as follows:

1. The solution shall have a single TL1 command that contains all necessary information to perform a file transfer from an OFS in the IP access DCN to a target network element in the OSI embedded DCN via an FT-TD.
2. The file transfer function shall be independent of the buffer size on the FT-TD.
3. The file transfer function shall be independent of the type of media used for the buffer on the FT-TD.
4. The FT-TD must support the following management commands :
   - Command to remove file from the file server
   - Command to list files on the file server
   - Command to query filestore space remaining on the file server.

6.2.1 Characteristics Of The IP-Based OSs

The following are characteristics of the IP-based OSs. In general, a service provider will have several OSs distributed over a large geographical area.

1. All IP-based OSs will use protocols from the TCP/IP protocol suite.
2. The interactive protocol used by the IP-based OSs is TL1.
3. The file transfer protocol between the IP-based OSs is FTP.
4. Management of entities in the IP-Based OSs is performed using tools and tool sets based on well known TCP/IP applications and protocols, e.g., SNMP, Telnet, PING, and Traceroute. Specific management requirements are contained in Section 11.
5. The OS does not use or pass NSAP information to the TD for file transfers.

6.2.2 Characteristics Of The IP-Based DCN And The Translation Devices

The following are characteristics of the IP-Based DCN and the translation devices of the reference architecture.

1. The IP-based DCN and the translation devices are under the administrative control of the service provider/network operator.
2. The IP-Based DCN is a protected, secured network.
3. Entities in the IP-Based DCN are aware of, at most, network layer protocols. Consequently, this network is transparent to Upper Layer Protocols used by the IP-based OSs.
4. Management of entities in the IP-Based DCN and of the translation devices is performed using tools and tool sets based on well known TCP/IP applications and protocols, e.g., SNMP, Telnet, PING, and Traceroute. Specific management requirements are contained in Section 11.
5. The IP-Based DCN may use various physical and data link layer protocols.
6. The IP-Based DCN must have at least one DNS server. DNS is required to allow operation through firewalls and also to minimize administrative changes related to IP addresses.

6.2.3 Characteristic Of The Translation Function

The communications between an IP-based OS and an OSI-based NE may be interactive or they may involve file transfers. Examples of interactive communications include an OS requesting the value of parameters of the NE data model such as the current interval bit error ratio. Alternatively an NE may autonomously transmit notifications to an OS such as the notification that a performance parameter threshold has been exceeded. File transfers are used to load software on an NE or to remotely back up data stored in an NE. Both types of OS-NE communication require a translation function.

The following IP-based OS to OSI-based NE interworking issues are addressed in Section 7.3:

1. The translation device must determine the NSAP of the OSI-based NEs for IP-based OS to OSI-based NE messages.
2. The translation device must direct autonomous messages from OSI-based NEs to the appropriate IP-based OS(s).
3. Responsibility for establishing connections between IP-based OSs, translation devices, and OSI-based NEs.
4. In general, any translation devices attached to a CLNP-Based DCN component can forward packets from any NE in that component to any IP-based OS.
5. Translation devices attached to the same DCN component are not necessarily co-located, i.e., are not necessarily located in the same central office.
6. In general, more than one translation device may be attached to any CLNP-based DCN, e.g., the use of dual GNEs.
7. Translation devices which are co-located may or may not be attached to the same LAN.
8. The translation function should not add new types of network broadcast traffic to the IP network.
9. The translation function shall not preclude the use of more than one TD, e.g., dual GNE configuration, for high-availability access to the particular NEs.

6.2.4 Characteristics Of The CLNP-Based NEs

The following are characteristics of the CLNP-based component of the management network shown in Figure 6-1. An NE may have one or more operations applications.

1. An NE may be an end NE or an intermediate NE in the sense of the CLNP routing protocol.
2. An NE may be a GNE.
3. A GNE may contain a translation function.

7 Specifications

These requirements should enable a GNE to be constructed where the distinction between a GNE and a GNE/TD is at most a different software image on a common hardware platform or is a provisioning difference.

7.1 Common Translation Device Specifications

This section contains specifications which are common to all Translation Devices (TDs) as portrayed by Figure 6-1 and Figure 7.1.

7.1.1 Naming and Addressing Specifications

R7-1 An IP-based OS shall have at least one IP address.

R7-2 A TD shall have at least one IP address.

R7-3 Fully qualified DNS names shall be supported for all communications between the TD and the OS. It shall be possible to avoid IP addresses being carried in any application layer PDUs other than DNS or FTP.

R7-5 Each OS shall have access to all IP addresses of each TD which has access to any NE contained within the managed domain of the OS.

R7-6 Each TD shall have a unique TID and shall support the TARP protocol running on the OSI DCN.

R7-7 The service provider IP network shall have a DNS server independent of the TD.

R7-8 The OS shall have the capability to store a relationship between the Target NE TID and a minimum of two host names of TL1 TDs.

7.1.2 Other Specifications

R7-9 The IP-Based DCN shall support IP Version 4.

O7-10 The TD will implement the IP version 6 protocol in addition to the current generation of IP version 4.

R7-11 The TD is required to interwork OS-NE operations communications traffic. It is not required to interwork NE-NE operations communications traffic (i.e., traffic that does not either originate or terminate at an OS).

7.2 Specifications For TL1 over TCP

Unless specifically mentioned otherwise, all requirements apply to both TL1 initiators and TL1 responders.

7.2.1 General Requirements

R7-12 TL1 message formats shall conform to GR-831-CORE.

Note: This is to allow the T-TD to find the Target Identifier (TID) and use the TID to discover the NSAP of the CLNP-based NE. Upon reception of an ACT-USER command, the T-TD will parse the TID from the TL1 message and resolve the TID to an NSAP. Upon successful resolution the NSAP, the T-TD will establish an association between itself and the target NE.

R7-16 The length-value encoding of TL1 over TCP as defined in section 7.2.2 shall be supported. The length-value encoding is the preferred encoding for machine-machine interactions.

O7-17 The raw encoding of TL1 over TCP as defined in section 7.2.3 should be supported. The raw encoding is allowed only for backwards compatibility.

O7-18 The telnet encoding of TL1 over TCP as defined in section 7.2.4 should be supported. The Telnet encoding is provided for human-machine interactions.

The length-value encoding is the preferred encoding for machine-machine interactions; the raw encoding is allowed only for backwards compatibility. The Telnet encoding is provided for human-machine interactions.

R7-19 Either side of a TL1 connection (i.e. the responder side or the initiator side) may disconnect the connection at any time. When the remote side disconnects a connection or when a connection is aborted all resources allocated for that connection shall be de-allocated.

R7-20 Either side of a TL1 connection (i.e. the responder side or the initiator side) may send a keep-alive probe message at any time during the connection. However, keep-alive probes shall always be sent between TL1 messages and never in the middle of a TL1 message.

R7-21 Received keep-alive probes shall be silently discarded.

A keep-alive probe is a message which can be sent across a TL1 connection without being delivered to the TL1 application at the remote end of the connection. The purpose of a keep-alive probe is to detect a half-open TCP connection, i.e. a TCP connection whose remote side is not operational anymore (the TCP protocol will not spontaneously abort half-open connections unless an attempt is made to send some data across it). The format of a keep-alive probe depends on the encoding (see sections 7.2.2, 7.2.3, and 7.2.4).

R7-22 The Nagle algorithm defined in [RFC896] shall be disabled on all TL1 connections. If sockets are used in the implementation, this requirement implies that the TCP_NODELAY socket option should be set (see [Stevens] page 202 for details).

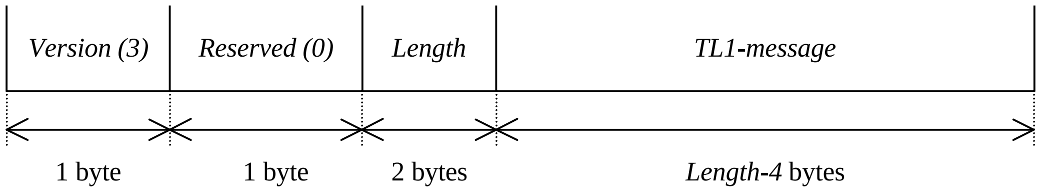
R7-23 TL1 initiators shall use ephemeral TCP port numbers as the source port number for all outgoing TL1 connections.

Ephemeral port numbers are sometimes called dynamic port numbers or private port numbers. See section 2.7 of [Stevens] for details.

7.2.2 Specification of Length-Value Encoding

This section specifies the length-value encoding of TL1 over TCP which TL1 initiators and TL1 responders shall implement according to requirement R7-16.

R7-24 Each TL1 message shall be encapsulated into the TCP data stream as follows:



Version: The sender shall set this field to 3 for the version of length-value encoding described in this contribution. If a non-supported version number is received, the TCP connection shall be closed (at the time of writing, version 3 is the only version but additional version numbers may be defined in future enhancements).

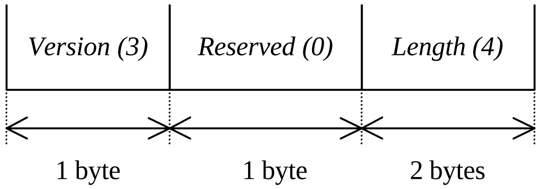
Reserved: The sender shall set this field to 0. The receiver shall ignore this field.

Length: Length of the TL1 message in octets *including* the version, reserved and length header fields. The length shall be encoded in network byte order. For TL1 messages, the *length* field shall contain a value between 5 and 4100. Value 4 is used by keep-alive probes. Values 4101...65535 are reserved for future use (if the *length* field contains a value greater than 4100, the next *length-4* bytes shall be read and discarded).

TL1-message: The TL1 message.

This encoding scheme is almost identical to the packetization described in RFC1006. Two minor differences between the encoding described here and the encoding described in RFC1006 are: (a) this explicitly requires that the *reserved* field shall be set to zero, and (b) in RFC1006 the minimum value for the *length* field is 7 because RFC1006 is used to carry TP0 instead of TL1 (the server TCP port number is used to determine the contents of the packetized stream: TP0 or TL1).

R7-25 Keep-alive probes shall be encoded as an empty TL1 message:



Version: Shall be 3 for the version of length-value encoding described in this requirement.

Reserved: Shall be 0.

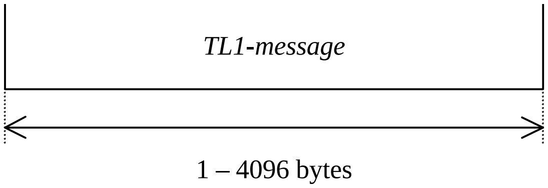
Length: Shall be 4.

R7-26 TL1 responders shall use port number 3081 to listen for incoming TL1 connections using the length-value encoding.

7.2.3 Specification of Raw Encoding

This section specifies the raw encoding of TL1 over TCP which TL1 initiators and TL1 responders should implement according to requirement O7-17

R7-27 Each raw encoded TL1 message shall be encapsulated into the TCP data stream as follows:



When the raw encoding is used, TL1 messages shall be parsed to find the boundary between successive TL1 messages (see [Bellcore GR-831-CORE] for the syntax of TL1 messages). The parser shall locate TL1 terminator strings which denote the end of a TL1 message. The TL1 terminator string depends on the type of TL1 message:

- *Commands*: a semi-colon (;) or a cancel character (ASCII code 0x18).
- *Acknowledgements*: a carriage return (CR) followed by a linefeed (LF) followed by a smaller-than (<) character.
- *Responses* and *notifications*: either a carriage return (CR) followed by a linefeed (LF) followed by a semi-colon (;), or a carriage return (CR) followed by a linefeed (LF) followed by a greater-than character (>).

The TL1 parser shall ignore terminator characters which are embedded in strings, comments, or escape sequences (this does not apply to the cancel character which may occur anywhere in a command).

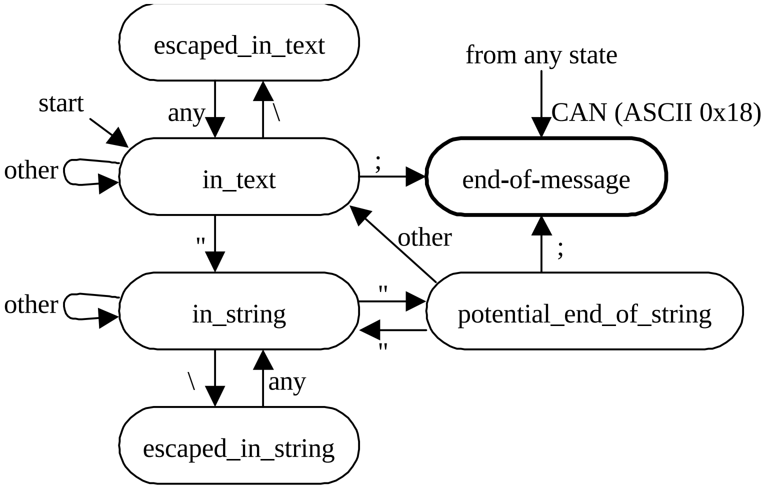
To ensure interoperability, requirements R7-28 and R7-29 provide a detailed specification of a state machine which shall be used to parse TL1 messages. These state machines are only intended to provide an unambiguous description of the required external behavior of the T-TD; they are not intended to impose any restrictions on the internal implementation of the T-TD.

A state machine is characterized by states (ovals) and transitions (arrows). The state machine starts in the state pointed to by the "start" transition. For each character received from the TCP data stream a state transition takes place by following the transition arrow labeled with the received character. The reception of a TL1 message is complete when the "end-of-message" state is reached.

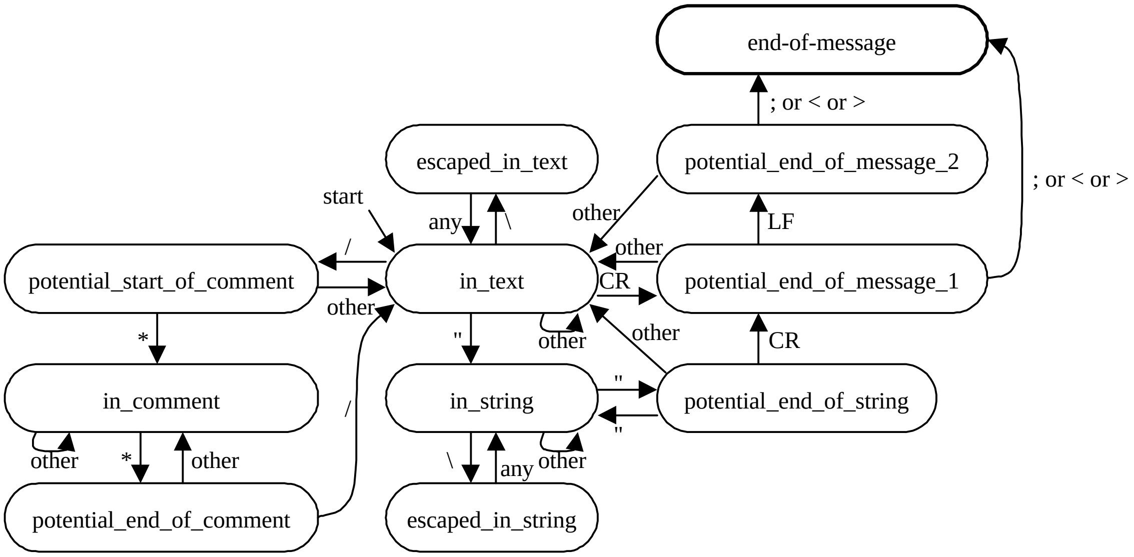
The state machines rely on the fact that TCP connections are reliable; they should not be re-used for other applications involving non-reliable links (e.g. TL1 over RS-232).

Note that parsing is only needed for received TL1 messages; not for sent TL1 messages.

R7-28 The behavior of the following state machine shall be used to locate the end of a received TL1 input message (i.e. TL1 commands).



R7-29 The behavior of the following state machine shall be used to locate the end of a received TL1 output message (i.e. TL1 acknowledgements, TL1 responses, and TL1 notifications).



The state machine for locating the end of a TL1 input message is slightly different from the state machine for locating the end of a TL1 output message. The reasons for this are:

(a) In addition to a semi-colon, a greater than or a less than character may also be used to terminate an output message.

(b) A response or notification might contain a semi-colon, greater than, or less than character outside a string or a comment which is nevertheless not a terminator because it is not preceded by a carriage-return and linefeed.

(c) The cancel character does not apply to output messages.

Note that the state machine allows CR-NUL or just CR at the end of a TL1 output message instead of the CR-LF required by [Bellcore GR-831-CORE]. The reason for this is that the state machine is also used in the Telnet encoding and some Telnet clients generate CR-NUL or just CR instead of CR-LF when the enter key is pressed.

R7-30 Characters which are not in the TL1 character set shall be silently discarded upon reception *prior* to exercising the state machines specified in R7-28 and R7-29. According to [Bellcore GR-831-CORE] the following characters are outside the TL1 character set: ASCII codes 0-8, 14-23, 25-31, and >= 127.

R7-31 Keep-alive probes shall use the character ASCII 255 immediately followed by the character ASCII 241 as the keep-alive probe.

Both of these characters are outside of the TL1 character set. This character pair was chosen because in the Telnet protocol the command IAC NOP is encoded as ASCII 255 241, and it is desirable that the same keep-alive probe encoding is used for the raw encoding and the Telnet encoding.

R7-32 TL1 responders which support the raw encoding shall use port number 3082 to listen for incoming TL1 connections using the raw encoding.

7.2.4 Specification of TELNET Encoding

This section specifies the telnet (see [RFC854]) encoding of TL1 over TCP which TL1 responders should implement according to requirement R7-18.

Possible applications of the telnet encoding include:

(a) Managing network elements using off-the-shelf Telnet clients.

(b) Obtaining access to a network element through an off-the-shelf terminal server which provides reverse Telnet.

The telnet encoding is a variation on the raw encoding, and inherits some requirements from section 7.2.3.

R7-33 The following requirements for the raw encoding also apply to the telnet encoding:

R7-27 Encapsulation of TL1 messages in a TCP stream.
R7-28 State machine for decoding TL1 input messages.
R7-29 State machine for decoding TL1 output messages.
R7-30 Silent discard of characters outside the TL1 character set.
R7-31 Encoding of keep-alive probe.

Like the raw encoding, the Telnet encoding relies on the TL1 delimiter characters to detect the boundaries between TL1 messages. The state machine used by the Telnet encoding is identical to the state machine used by the raw encoding except that the TL1 message may contain Telnet commands and VT (virtual terminal) characters which shall be filtered from the data stream and processed separately as described in additional requirements below.

Keep-alive probes are encoded differently in telnet encoding than in raw encoding because off-the-shelf Telnet clients will not silently discard the ENQ character but they will silently discard the Telnet NOP command:

R7-35 The Telnet ECHO option shall be supported as follows:

(a) A state variable *echo* is associated with each individual telnet encoded TL1 connection.

(b) The initial value of *echo* is FALSE.

(c) When the Telnet command IAC DO ECHO (ASCII 255 253 1) is received, it shall be processed as follows.

- If the state variable *echo* is FALSE, the Telnet command IAC WILL ECHO (ASCII 255 251 1) shall be sent back and the TL1 connection state variable *echo* shall be set to TRUE.
- If the state variable *echo* is TRUE, no action shall be taken.

(d) When the Telnet command IAC DON’T ECHO (ASCII 255 254 1) is received, it shall be processed as follows.

- If the state variable *echo* is TRUE, the Telnet command IAC WON’T ECHO (ASCII 255 252 1) shall be sent back and the TL1 connection state variable *echo* shall be set to FALSE.
- If the state variable *echo* is FALSE, no action shall be taken.

(e) When the Telnet command IAC WILL ECHO (ASCII 255 251 1) is received, the Telnet command IAC DON’T ECHO (ASCII 255 254 1) shall be sent back.

(f) When the Telnet command IAC WON’T ECHO (ASCII 255 252 1) is received, the Telnet command IAC DON’T ECHO (ASCII 255 254 1) shall be sent back.

(g) When the connection is initially established, either the Telnet command IAC WILL ECHO or IAC WON'T ECHO may be sent to indicate the local preference for using or not using echo to the remote side.

(h) If the TL1 connection *echo* state variable is TRUE, then each received character which is in the TL1 character set and which is not part of a Telnet command is echoed. When a carriage-return is received and echo is enabled, then a carriage-return and a line-feed is sent back. Received line-feeds are never echoed.

The reason for this strange processing of carriage returns and linefeeds is that some Telnet clients send CR-LF when the enter key is hit, some Telnet clients send CR-NUL when the enter key is hit, some Telnet clients send just CR when the enter key is hit, and some Telnet clients support all of the above.

R7-36 The Telnet SUPPRESS GO AHEAD option shall be supported as follows:

(a) A state variable *suppress-go-ahead* is associated with each individual telnet encoded TL1 connection.

(b) The initial value of *suppress-go-ahead* is FALSE.

(c) When the Telnet command IAC DO SUPPRESS-GO-AHEAD (ASCII 255 253 3) is received, it shall be processed as follows.

- If the state variable *suppress-go-ahead* is FALSE, the Telnet command IAC WILL SUPPRESS-GO-AHEAD (ASCII 255 251 3) shall be sent back and the TL1 connection state variable *suppress-go-ahead* shall be set to TRUE.
- If the state variable *suppress-go-ahead* is TRUE, no action shall be taken.

(d) When the Telnet command IAC DON’T SUPPRESS-GO-AHEAD (ASCII 255 254 3) is received, it shall be processed as follows.

- If the state variable *suppress-go-ahead* is TRUE, the Telnet command IAC WON’T SUPPRESS-GO-AHEAD (ASCII 255 252 3) shall be sent back and the TL1 connection state variable *suppress-go-ahead* shall be set to FALSE.
- If the state variable *suppress-go-ahead* is FALSE, no action shall be taken.

(e) When the Telnet command IAC WILL SUPPRESS-GO-AHEAD (ASCII 255 251 3) is received, the Telnet command IAC DO SUPPRESS-GO-AHEAD (ASCII 255 253 3) shall be sent back.

(f) When the Telnet command IAC WON’T SUPPRESS-GO-AHEAD (ASCII 255 252 3) is received, the Telnet command IAC DON’T SUPPRESS-GO-AHEAD (ASCII 255 254 3) shall be sent back.

(g) When the connection is initially established, either the Telnet command IAC WILL SUPPRESS-GO-AHEAD or IAC WON'T SUPPRESS-GO-AHEAD may be sent to indicate the local preference for using or not suppressing go-ahead to the remote side.

(h) If the TL1 *suppress-go-ahead* state variable is FALSE, then the Telnet command IAC GA (ASCII 255 249) is appended to each TL1 output message (i.e. each TL1 acknowledgement, TL1 response, or TL1 notification). When a Telnet IAC GA (ASCII 255 249) is received it is always ignored.

R7-37 An implementation may support additional Telnet options.

Only the support for the ECHO and the SUPPRESS-GO-AHEAD options is required. Support for other Telnet options is optional.

R7-38 If an implementation does not implement a given telnet option (say option number *option*) then that implementation shall refuse to negotiate that option as follows:

(a) When the Telnet command IAC DO *option* (ASCII 255 253 *option*) is received, the Telnet command IAC WON'T *option* (ASCII 255 251 *option*) shall be sent back.

(b) When the Telnet command IAC DON'T *option* (ASCII 255 254 *option*) is received, no action shall be taken.

(c) When the Telnet command IAC WILL *option* (ASCII 255 251 *option*) is received, the Telnet command IAC DON'T *option* (ASCII 255 254 *option*) shall sent back.

(d) When the Telnet command IAC WON'T *option* (ASCII 255 252 *option*) is received, the Telnet command IAC DON'T *option* (ASCII 255 254 *option*) shall be sent back.

Note that an implementation which only supports the minimally required ECHO and SUPPRESS-GO-AHEAD options and refuses to negotiate all other options will never enter Telnet subnegotiations (which greatly simplifies implementation).

R7-39 Line editing shall be supported as follows:

(a) When the Telnet command IAC EC (ASCII 255 247) is received or when the character backspace (ASCII 8) is received or when the character delete (ASCII 127) is received, it is processed as follows.
If the receive buffer is empty, or if the last character in the receive buffer is a carriage return or linefeed, then the received IAC EC Telnet command or backspace character or delete character is ignored.
If the receive buffer is not empty and the last character in the receive buffer is not a linefeed, then the last character is removed from the receive buffer and (if echo is enabled) the following characters are sent back: backspace, space, backspace.

(b) When the Telnet command IAC EL (ASCII 255 248) is received it is processed as follows.

Starting at the end of the receive buffer, all characters are removed from the receive buffer until the receive buffer is empty or until a carriage return or linefeed is encountered. If echo is enabled, for each character thus removed from the receive buffer, the following characters are sent: backspace, space, backspace.

Inline editing allows a user which is managing a network element using an off-the-shelf Telnet client to correct typing mistakes using editing keys such as backspace.

R7-40 All other received Telnet commands not specifically mentioned in the above requirements may be ignored.

R7-41 When a TCP connection is in urgent mode (signaling the presence of urgent data on the TCP connection) incoming data must be processed in the same way as when the TCP connection is in normal mode (i.e. not in urgent mode). Specifically, when the TCP connection is in urgent mode, incoming user data (i.e. TL1 messages) must not be discarded.

TCP urgent mode is described in [Stevens] chapter 21, and the usage of TCP urgent mode in Telnet is described in [RFC854] in section "The Telnet SYNCH signal". In Telnet, a SYNCH signal is sent by writing the Data Mark (DM) character into the TCP stream in urgent mode. This causes the remote side of the Telnet connection to enter urgent mode in which it is allowed to discard incoming user data until the DM is encountered (at which point the remote side exits urgent mode). The above requirement R7-41 specifies that user data (i.e. TL1 messages) must not be discarded in urgent mode but must instead be processed in the same way as when the connection is in normal mode (i.e. not in urgent mode).

R7-42 TL1 responders shall support prompts on a Telnet encoded TL1 connection as follows:

(a) Whenever the TL1 responder has finished reading a complete TL1 command (i.e. the state machine has reached state end-of-message) and is ready to begin reading the next TL1 command it should send a prompt string.

(b) The prompt string shall be provisionable and it shall be possible to turn prompts off. The default prompt string shall be carriage-return, line-feed, '[', TID of the system, ']', space (e.g. "\r\n[server] ").

Prompts are intended to make the exchange of messages between network elements and users who use an off-the-shelf Telnet client more user-friendly: without prompts the cursor will appear after the semi-colon of the last response or notification. Also, the presence of the TID in the prompt will allow the user to quickly determine to which system he/she is connected.

R7-43 TL1 responders which support the telnet encoding shall use port number 3083 to listen for incoming TL1 connection using the telnet encoding.

7.3 Specifications for TL1/TCP – TL1/OSI Gateway (T-TD)

Several aspects of the referenced networks are illustrated in the Figure 7-1 below:

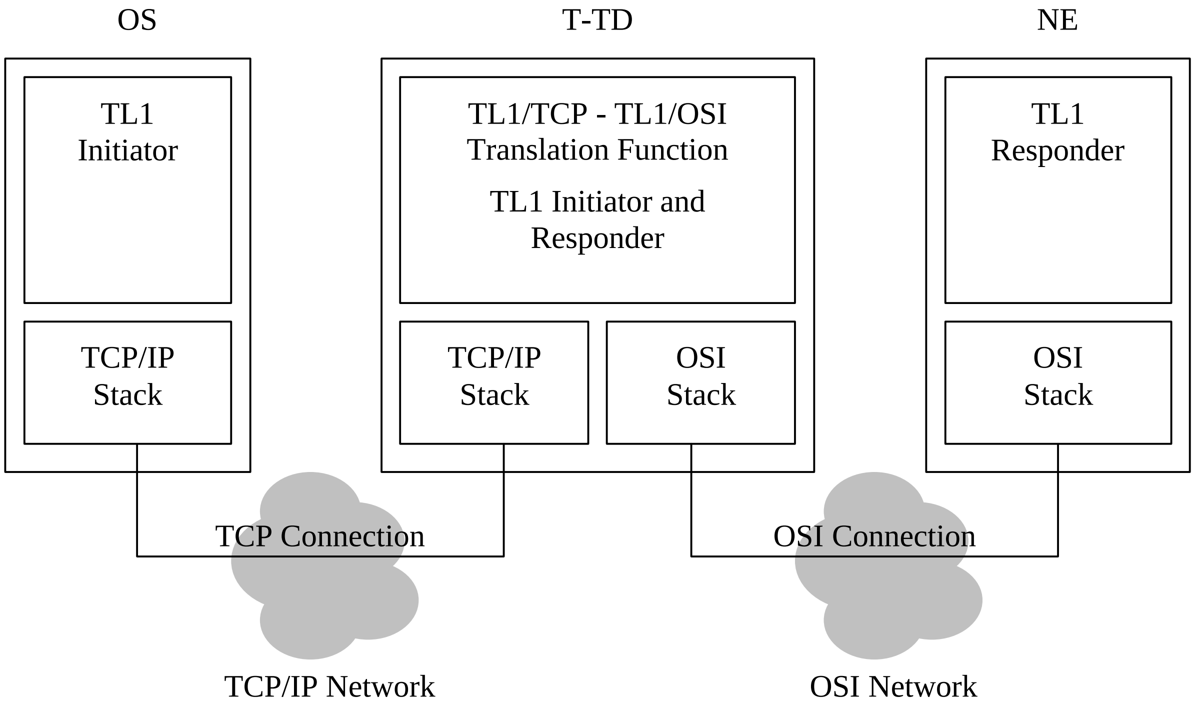


Figure 7-1: TCP/IP and OSI Reference Networks

The phrase "*the stack closes send flow-control for a connection*" means that the stack indicates that it is temporarily not willing to accept any additional data for transmission due to flow-control that has been exerted upon it from the remote side of the connection.

The phrase "*the stack opens send flow-control for a connection*" means that the stack indicates that it is willing to accept more data for transmission over that connection.

The phrase "*the T-TD closes receive flow-control for a connection*" means that the TL1/TCP-TL1/OSI translation function in the T-TD informs the stack that it is temporarily not capable of accepting data received over a connection. The remote side of the connection will eventually be stopped from sending additional data using the TCP or TP4 flow-control mechanisms.

The phrase "*the T-TD opens receive flow-control for a connection*" means that the TL1/TCP-TL1/OSI translation function in the T-TD informs the stack that is capable for accepting data received over a connection. Once the T-TD actually consumes some received data from the connection, the remote side will eventually be allowed to send additional data using the TCP or TP4 flow-control mechanisms.

The mechanisms which are used to open and close flow-control vary depending on the design of the stack API.

7.3.1 Overview of the T-TD.

The OS may (but is not required to) multiplex several TL1 sessions over a single TCP connection. Each TCP connection may map to several OSI connections but each OSI connection maps to exactly one TCP connection. In other words, there is a one-to-many (1:N) mapping between TCP connections and OSI connections. When a TL1 message arrives on a TCP connection, the T-TD uses the TID in that TL1 message to determine over which OSI connection to forward the TL1 message. Figure 7-2 below shows an example of how OSI connections are mapped to TCP connections.

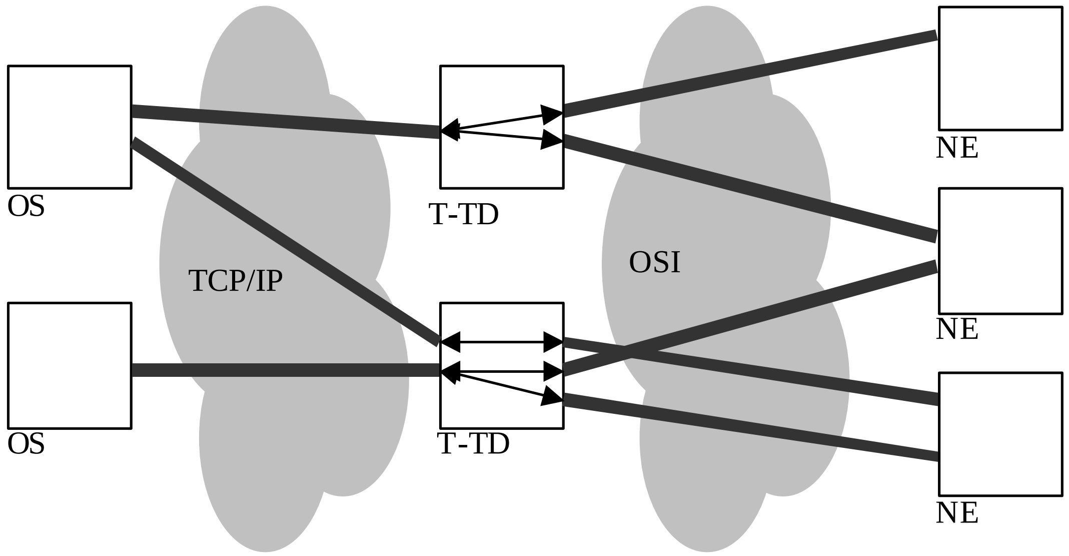


Figure 7-2: One-to-many mapping example application.

Figure 7-3 below shows the conceptual architecture of the T-TD.

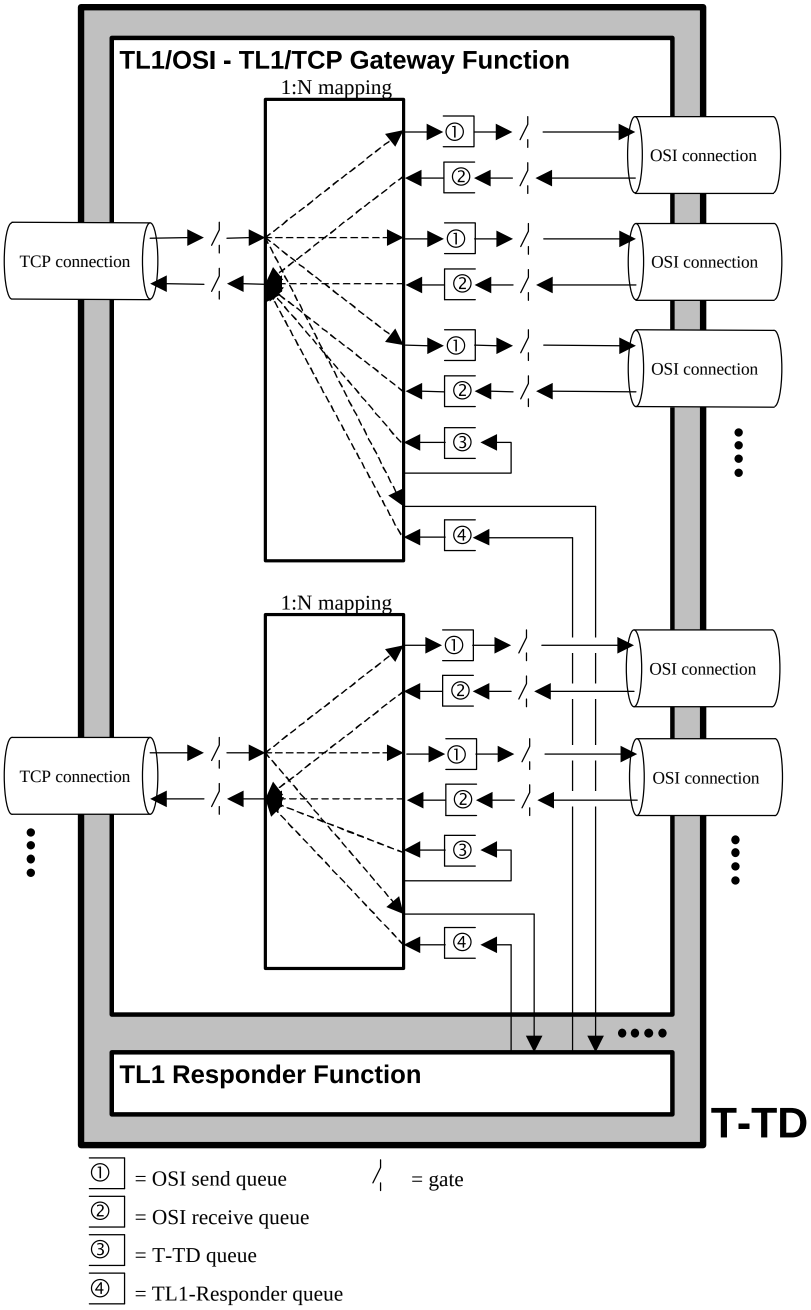


Figure 7-3: Conceptual architecture of the T-TD.

The pipes on the left side represent TCP connections to the OS. There can be multiple (zero or more) TCP connections because there can be multiple OSs and each OS can setup multiple TCP connections to a given T-TD.

The pipes on the right side represent OSI connections to remote NEs. There can be multiple (zero or more) OSI connections per TCP connection, namely one per TL1 session which is multiplexed over that TCP connection.

The gates represent the points at which flow control can be closed or opened.

The OSI send queues represent the points at which the T-TD needs to queue TL1 messages which are waiting to be sent out over an OSI connection to a remote NE. There is exactly one OSI send queue per OSI connection.

The OSI receive queues represent the points at which the T-TD needs to queue TL1 messages which have arrived from a remote NE and which are waiting to be sent out over a TCP connection. Note that we use one queue per OSI connection instead of one queue per TCP connection, because we need to control resources and exercise flow-control on a per TL1 session basis. There is exactly one OSI receive queue per OSI connection.

The T-TD queues represent the points at which the T-TD needs to queue TL1 messages which are generated by the Gateway Function on the T-TD itself (as opposed to received over an OSI connection or generated by the TL1 Responder Function) and which are waiting to be sent over a TCP connection. There is exactly one T-TD queue per TCP connection.

The TL1-Agent queues represent the points at which the T-TD needs to queue TL1 messages which are generated by the TL1 Responder Function on the T-TD itself and which are waiting to be send over a TCP connection. There is exactly one TL1-Agent queue per TCP connection.

Due to finite resources the various queues cannot grow without bounds. This document specifies how the T-TD shall exercise flow control on the OSI and/or TCP connections to attempt to prevent the queues from overflowing. These requirements are intended to greatly reduce the probability that TL1 messages are lost due to queue overflows. However, we do not claim that these requirements are sufficient to guarantee that TL1 messages are never dropped. This document also contains requirements specifying how the T-TD shall behave when TL1 messages are dropped due to resource shortages. These requirements designed to (a) assist the OS to detect and recover from the lost TL1 message and (b) be backwards compatible with existing OSs.

The TL1/TCP - TL1/OSI Gateway Function is the function within the T-TD which is responsible for relaying multiplexed TL1 sessions carried over TCP connections from OSs to non-multiplexed TL1 sessions over OSI connections to remote NEs.

The TL1 Responder Function represents the function within the T-TD which is responsible for processing TL1 commands which are directed to the T-TD itself (as opposed to a remote NE).

The conceptual architecture of the T-TD described above is intended only as a conceptual model to clarify the requirements in the remainder of this document. It is not intended to restrict the implementation of the T-TD in any way, as long as the external behavior of the T-TD is consistent with the requirements in this document.

7.3.2 Encoding Requirements

R7-44 The T-TD is subject to the requirements for TL1 encoding over TCP as specified in Section 7.2.

R7-45 The T-TD is subject to the requirements for TL1 encoding over OSI presentation as described in section 8.3.7.5 of [GR-253-CORE]. However, [GR-253-CORE] requirement R8-81 shall be replaced with the requirement that "Each TL1 message shall be carried in a separate presentation data SDU".

[GR-253-CORE] requirement R8-81 only requires that TL messages shall be exchanged using Data Presentation Protocol Data Units (TD PPDUs). The Bellcore requirement allows multiple TL1 message to be placed in a single Presentation Data SDU and it allows a single TL1 message to be spread out over several Presentation Data SDUs. The new SIF requirement requires each TL1 message to be placed in its own individual Presentation Data SDU which removes the need for parsing the TL1 messages to locate the boundaries of individual TL1 messages. This is consistent with current implementations of TL1 over OSI Presentation.

7.3.3 TCP Connection Establishment Requirements

R7-46 The T-TD shall never initiate a TCP connection for a TL1 session.

The TCP connections are always initiated by the OS and never by the T-TD.

R7-47 When an incoming TCP connection for a TL1 session arrives, the T-TD shall decide to accept or reject it based on the following criteria:

(a) *Available resources*. Incoming TCP connection may be rejected if the T-TD does not have sufficient resources to handle the connection. The vendor of the T-TD shall document the following two numbers:

i. *MaxTcpConnections*: The T-TD is guaranteed by the vendor to support at least this many TCP connections carrying TL1 sessions.

ii. *MaxOsiConnectionsPerTcpConnection*: The T-TD is guaranteed by the vendor to support at least this many OSI connections (i.e. TL1 sessions) per TCP connection.

(b) *Access control*. The access control mechanisms described in section 7.3.12 shall be implemented.

The T-TD can recognize an incoming TCP connection as a TL1 session by the port number on which the TCP connection arrives.

R7-48 The T-TD shall allow the OS to multiplex several TL1 sessions over a single TCP connection.

The OS is allowed to share a single TCP connection between all or some of the NEs which are managed through a single T-TD.

It is not a requirement that the OS shall use a single TCP connection to a T-TD and multiplex all TL1 sessions through that T-TD over that single TCP connection. The OS is allowed to create several TCP connections to a single T-TD and multiplex any number of TL1 sessions (from 1 up to *MaxOsiConnectionsPerTcpConnection*) over each TCP connection.

As a special case of this general rule, the OS is allowed to setup a separate TCP connection for each TL1 session. If the OS behaves in this way, there will be a one-to-one (1:1) mapping of TCP connections to OSI associations. The flow-control requirements (see section 7.3.8) have been carefully written so that in this case there is a strict coupling of flow-control between the TCP connection and the OSI association which lowers the probability that a TL1 message might be dropped due to resource shortages on the T-TD.

R7-49 When an incoming TCP connection carrying a TL1 session is accepted, the T-TD shall create a T-TD queue and a TL1-Agent queue for that TCP connection.

The T-TD queue is used to queue TL1 message which are generated by the Gateway Function on the T-TD itself and which can not be associated with an existing OSI connection (for example, a deny response which is sent by the T-TD when a TID resolution fails).

The TL1-Agent queue is used to queue TL1 messages which are generated by the TL1 Responder Function on the T-TD itself (for example, TL1 responses to TL1 command which are directed to the T-TD itself as opposed to a remote NE).

7.3.4 OSI Connection Establishment Requirements

R7-50 The T-TD shall reject all incoming OSI connections for TL1 sessions.

The OSI connections are always initiated by the T-TD and never by the remote NEs

R7-51 The T-TD shall use a separate OSI connection for each TL1 session.

R7-52 The T-TD shall maintain the following state information for each OSI connection:

(a) The TID of the NE at the remote side of the OSI connection.

(b) The TCP connection to which the OSI connection maps. There is a one-to-many mapping between TCP connections and OSI connections.

R7-53 For each TL1 message received over a TCP connection, the T-TD shall:

(a) Check whether the TID in the TL1 message matches the TID of the T-TD itself. If so, the TL1 message shall be passed to the TL1 Responder Function on the T-TD itself.

(b) Otherwise, amongst the set of OSI connections mapping to the TCP connection over which the TL1 message was received, locate the corresponding OSI connection whose remote TID matches the TID in the TL1 message.

(c) If no such OSI connection exists the T-TD shall:

(i) Resolve the TID in the received message to an OSI PSAP address and application context identifier as specified in section 7.3.12.

(ii) Establish an OSI connection.

(iii) Create a new OSI send queue, create a new OSI receive queue, and append the TL1 message to the OSI send queue.

The establishment of an OSI connection can be triggered by any TL1 command (as opposed to only by ACT-USER TL1 commands).

Section 7.3.7 specifies the required behavior for when the destination OSI connection was found.

R7-54 If the TID resolution described in requirement 7-53 item c-i fails, the following TL1 response message shall be appended to the T-TD queue for that TCP connection:

```
<cr><lf><lf>
^^^<rne-tid>^<year>-<month>-<day>^<hour>:<minute>:<second><cr><lf>
M^^<ctag>^DENY<cr><lf>
^^^IITA<cr><lf>
^^^/*^Input^Invalid^Target^ID^(sent^by^T-TD^<ttd-tid>)^*/<cr><lf>
;
```

<rne-tid> shall be set to the TID of the remote NE, i.e. the TID in the original TL1 command which caused the attempt to setup the OSI connection.
<ctag> shall be set to the ctag of the original TL1 command which caused the attempt to setup the OSI connection.
<ttd-tid> shall be set to the TID of the T-TD.

R7-55 If the OSI connection establishment described in requirement 7-53 item c-ii fails, the following TL1 response message shall be appended to the T-TD queue for that TCP connection:

```
<cr><lf><lf>
^^^<rne-tid>^<year>-<month>-<day>^<hour>:<minute>:<second><cr><lf>
M^^<ctag>^DENY<cr><lf>
^^^SRSD<cr><lf>
^^^/*^Status^Remote^Session^Dropped^(sent^by^T-TD^<ttd-tid>)^*/<cr><lf>
;
```

<rne-tid> shall be set to the TID of the remote NE, i.e. the TID in the original TL1 command which caused the attempt to setup the OSI connection.
<ctag> shall be set to the ctag of the original TL1 command which caused the attempt to setup the OSI connection.
<ttd-tid> shall be set to the TID of the T-TD.

R7-56 If the T-TD does not have enough resources to establish a new OSI connection (e.g. the maximum number of OSI connections has been reached), the following TL1 response message shall be appended to T-TD queue for that TCP connection:

```
<cr><lf><lf>
^^^<rne-tid>^<year>-<month>-<day>^<hour>:<minute>:<second><cr><lf>
M^^<ctag>^DENY<cr><lf>
^^^SSRE<cr><lf>
^^^/*^Status^System^Resources^Exceeded^(sent^by^T-TD^<ttd-tid>)^*/<cr><lf>
;
```

<rne-tid> shall be set to the TID of the remote NE, i.e. the TID in the original TL1 command which caused the attempt to setup the OSI connection.
<ctag> shall be set to the ctag of the original TL1 command which caused the attempt to setup the OSI connection.
<ttd-tid> shall be set to the TID of the T-TD.

Note that requirements7-54, 7-55and 7-56 violate the Telcordia requirements specified in appendix A.2 of [TR-NWT-000835]. These requirements essentially specify that no specific error cause should be specified when a TL1 session is refused, because it might provide clues to an intruder. The choice to provide good diagnostic information even though it provides a theoretical security threat was deliberate.

7.3.5 Connection Release Requirements

R7-57 The T-TD shall never initiate disconnection of a TCP connection carrying a TL1 session, except when the T-TD has determined that the TCP connection is half-open (see section 7.3.10).

The OS is always responsible for initiating disconnection of a TCP connection carrying a TL1 session. The OS should disconnect all TCP connections carrying TL1 sessions before it is shut down. In some circumstances (e.g. when it crashes) the OS will not be able close the TCP connections; section 7.3.10 deals with this situation.

R7-58 When a TCP connection carrying a TL1 session is closed by the remote side (i.e. the OS) or aborted by the TCP/IP stack, the T-TD shall do the following:

(a) Discard all remaining TL1 message in the T-TD and TL1-Agent queues.

(b) Remove the T-TD and TL1-Agent queues.

The T-TD shall also do the following for every OSI connection mapping to that TCP connection:

(a) Discard all remaining TL1 message in the OSI receive and send queues.

(b) Remove the OSI receive queues and the OSI send queues.

(c) Abort the OSI connection.

Note that the queues are flushed when the OS closes a TCP connection. Thus TL1 messages which have been sent from the OS to T-TD may not be delivered to the remote NE, and TL1 messages which have been sent from the remote NE to the T-TD may not be delivered to the OS. As a result, when the OS closes a TCP connection, any TL1 commands which it has sent to a remote NE and for which no response or acknowledgement has been received from that remote NE, are not guaranteed to be delivered to that remote NE.

R7-59 When an OSI connection carrying a TL1 session is closed by the remote side (i.e. the NE) or aborted by the OSI stack, the T-TD shall do the following:

(a) Discard all remaining TL1 message in the OSI receive and send queues of that OSI connection.

(b) Remove the OSI receive and send queues of that OSI connection.

7.3.6 Requirements for Forwarding TL1 Messages to the OSs

R7-60 When a TL1 message is received over an OSI connection, the T-TD shall append the TL1 message to the OSI receive queue of that OSI connection.

R7-61 When a TL1 message is generated by the local TL1 Responder Function, the T-TD shall append the TL1 message to the TL1-Agent queue of the corresponding TCP connection.

The TL1 Responder Function shall somehow communicate to the Gateway function for which TCP connection a TL1 message is intended. This is a local implementation matter.

R7-62 The OSI receive queues, the T-TD queues, and the TL1-Agent queues shall be 'serviced' by removing a TL1 message from the head of the queue and initiating transmission of that TL1 message on the corresponding TCP connection whenever:

(a) The transmission of the previous TL1 message over the TCP connection has been completed.

(b) A TL1 message is appended to an OSI receive queue, T-TD queue, or TL1-Agent queue while no transmission of a TL1 message over the TCP connection is in progress (i.e. all OSI receive, T-TD and TL1-Agent queues were empty).

The sending of TL1 messages over the TCP connection is subject to send flow-control as exercised by the remote side of the TCP connection (i.e. the OS).

R7-63 The OSI receive, T-TD, and TL1-Agent queues shall be services in FIFO (first-in first-out) order to guarantee that TL1 messages for a given TL1 session arrive in the correct sequence.

R7-64 The OSI receive, T-TD, and TL1-Agent queues shall be serviced in such an order that each message source obtains a fair share of the available bandwidth on the TCP connection. A single NE nor the T-TD itself shall not be allowed to monopolize the available bandwidth of the TCP connection.

For example, servicing the OSI receive, T-TD, and TL1-Agent queues in a round-robin fashion will satisfy this requirement.

7.3.7 Requirements for Forwarding TL1 Messages from the OSs

R7-65 When a TL1 message is received over a TCP connection, the T-TD shall:

(a) Amongst the set of OSI connections mapping to the TCP connection over which the TL1 message was received, locate the corresponding OSI connection whose remote TID matches the TID in the TL1 message.

(b) If that OSI connection was found the T-TD shall append the TL1 message to the OSI send queue of that OSI connection.

Section 7.3.4 specifies the required behavior for when the destination OSI connection was not found.

R7-66 The OSI send queues shall be 'serviced' by removing a TL1 message from the head of an OSI send queue and initiating transmission of that TL1 message on the corresponding OSI connection whenever:

(a) The transmission of the previous TL1 message over the OSI connection has been completed.

(b) A TL1 message is appended to a OSI send queue while no transmission of a TL1 message over the OSI connection is in progress (i.e. the OSI send queue was empty).

The sending of TL1 messages over the OSI connection is subject to send flow-control as exercised by the remote side of the OSI connection (i.e. the remote NE).

R7-67 The OSI send queues shall be services in FIFO (first-in first-out) order to guarantee that TL1 messages for a given TL1 session arrive in the correct sequence.

7.3.8 Flow-control Requirements

The following requirements specify how the T-TD shall use flow-control to stop the remote side of a connection from sending more data when a queue is in danger of overflowing. These requirements are intended to greatly reduce the probability that TL1 messages are lost due to queue overflows. However, we do not claim that these requirements are sufficient to guarantee that TL1 messages are never dropped. In theory, the receiving side of a connection can close receive flow-control when it is no longer willing or able to receive incoming TL1 messages. In real implementations, it may be difficult or impossible to prevent a residual flow of arriving TL1 message on a connection for which receive flow-control has been closed. Section 7.3.9 contains requirements specifying how the T-TD shall behave when TL1 messages are dropped.

The following two requirements specify how the T-TD shall exercise receive flow-control on the OSI connections to prevent the OSI receive queues from overflowing:

R7-68 When an OSI receive queue is in danger of overflowing, the T-TD shall close receive flow-control for that OSI connection.

R7-69 When a sufficient number of TL1 messages have been removed from the OSI receive queue so that it is no longer in danger of overflowing, the T-TD shall re-open receive flow-control for that OSI connection.

The definition of 'in danger of overflowing' is implementation dependent. The two main goals are that (a) the probability of dropping a TL1 message due to overflow of the OSI receive queue is near-zero and (b) a single OSI connection is not allowed to monopolize all resources.

The mechanism described above creates a coupling between send flow-control on the TCP connection and receive flow-control on the OSI connections. When the TCP stack closes send flow-control on a given TCP connection, messages arriving on the OSI connections will queue up in the OSI receive queues. Eventually, the OSI receive queue length will grow to the point where it causes receive flow-control to be closed on the OSI connection.

The following two requirements basically state that in the general 1:N mapping the T-TD can *not* exercise receive flow-control on the TCP connection to prevent the OSI send queues from overflowing. This is caused by the fact that several TL1 sessions are multiplexed on a single TCP connection; if we stop incoming traffic on the TCP connection all TL1 sessions on the TCP connection are affected. Thus, if we allowed TCP receive flow control to be exercised, a single slow NE could cause starvation of all other NEs.

R7-70 If there is more that a single OSI connection mapping to a TCP connection, the T-TD shall not close receive flow-control on that TCP connection as long as there is at least one OSI connection mapping to that TCP connection whose OSI send queue is not in danger of overflowing.

Note that the above requirement has an escape clause: the T-TD *may* close TCP receive flow-control if *all* OSI send queues for that TCP connection are in danger of overflowing.

If the OS chooses to setup a separate TCP connection for each TL1 session, there is in effect a one-to-one (1:1) mapping between TCP connections and OSI connections. In this case we require a stricter form of flow-control on the TCP connection to reduce the probability of dropped TL1 commands.

R7-71 If there is only a single OSI connection mapping to a TCP connection, and if the OSI send queue for that OSI connection is in danger of overflowing, the T-TD shall close receive flow-control for that TCP connection.

R7-72 When a sufficient number of TL1 messages have been removed from the OSI send queue so that it is no longer in danger of overflowing, the T-TD shall re-open receive flow-control for that TCP connection.

7.3.9 Queue Overflow Handling Requirements

R7-73 When the T-TD is forced to drop a TL1 message which has arrived on a TCP connection due to overflow of the corresponding OSI send queue, the T-TD shall append the following TL1 retry later acknowledgement to the OSI receive queue of that OSI connection:

```
<cr><lf><lf>
RL^<ctag><cr><lf>
<
```

<ctag> shall be set to the ctag of the dropped TL1 message.

Note that this retry later acknowledgement may itself *also* be dropped if the OSI receive queue also overflows. For recovery from this extreme case we rely on the command time-out on the OS.

The generation of a retry-later ack is intended to signal to the OS that (a) the TL1 command could not be processed at this time and (b) the OS should slow down sending TL1 commands. An ill-behaved OS which sends a continuous stream of TL1 commands without waiting for TL1 responses and without slowing down when a retry-later ack is received will cause chronic overflowing of the OSI send queues and/or T-TD queues, unless the OS chooses to use a separate TCP connection per TL1 session (1:1 mapping).

R7-74 The T-TD shall not generate any TL1 notifications on behalf of a remote NE (i.e. with the SID of the notification set to the TID of the remote NE). Specifically, when the T-TD is forced to drop a TL1 message due to overflow of an OSI receive queue, the T-TD queue, or the TL1-Agent queue the T-TD shall *not* generate a TL1 notification on behalf of the remote NE to report this to the OS.

By the above requirement, dropped TL1 responses, acknowledgements, and notifications are not reported to the OS. For recovery from dropped TL1 responses and acknowledgements we rely on the command time-out on the OS. For recovery from dropped TL1 notifications we rely on the fact that the OS will notice the fact that <atag> numbers were skipped.

Some legacy TL1/X25 - TL1/OSI gateways generate a REPT EVT COM notification when they are forced to drop a TL1 message coming from the remote NE (and they typically also flush all queued notifications when that happens, because adding a new REPT EVT COM notification to an already overflowing queue doesn't make sense). Experience has shown this feature may cause interoperability problems because of two reasons:

(a) Many OSs rely on the fact that TL1 notifications coming from an NE have consecutively numbered <atag> values. If the T-TD inserts a notification on behalf of the remote NE, this causes out-of-sequence or duplicate <atag> values which can cause undesired behavior by the OS.

(b) Some OSs are not prepared to receive TL1 notifications. The remote NE decides which notification to report or not to report based on the application context identifier and/or the user-id of the session. The T-TD does not have knowledge about the set of notifications which shall and shall not be reported for a given TL1 session, so it doesn't know whether or not a REPT EVT COM notification should be sent to the OS.

Note that requirement 7-74 only prohibits a T-TD from generating a notification on behalf of another NE (i.e. a notification with a SID which is not the TID of the T-TD itself). It is not forbidden for the T-TD to generate a notification of its own to report that it is experiencing a resource shortage. Such notifications would only be sent over TL1 session which are logged in to the T-TD itself. The format of such a notification is not in the scope of this document.

7.3.10 Requirements for Detecting and Eliminating Half-open TCP Connections

A "half-open TCP connection" is a TCP connection for which the remote TCP host has shut down without closing the TCP connection. If an OS shuts down without properly closing its TCP connections to a T-TD (e.g. when the OS crashes) those TCP connections to the T-TD will become "half-open". As long as the half-open TCP connection continues to exist, it will use resources on the T-TD. Also, all OSI connections mapping to that half-open TCP connection will continue to exist, and remote NEs will continue to think that the OS is alive and logged in. When the OS reboots and attempts to establish a new TL1 session to the remote NEs, it may not be able to do so because the "previous incarnation" of the OS is still logged in at the NEs and the NEs do not allow the same user to be logged in twice; some vendors allow the same user to be logged into the NE more than once and others don't. For more information about half-open TCP connections see section 3.4 of RFC-793.

R7-100 It shall be possible to provision "Half-Open-TCP-Connection-Detection-Time". This parameter determines the time (in minutes) a half-open TCP connection shall be detected and eliminated. This parameter shall be provisionable in the range of 5 minutes to 240 minutes. The default value shall be 10 minutes.

R7-75 The T-TD shall detect a half-open connection within "Half-Open-TCP-Connection-Detection-Time" minutes of the moment that the remote side of the TCP connection (i.e. the OS) terminates due to a disorderly shut down.

R7-76 When the T-TD detects a half-open TCP connection it shall:

(a) Close the half-open TCP connection.

(b) Discard all remaining TL1 messages in the T-TD and TL1-Agent queues and remove the T-TD and TL1-Agent queues.

(c) For every OSI connection mapping to that TCP connection:

(i) Discard all remaining TL1 messages in the OSI receive and send queues.

(ii) Remove the OSI receive and send queues.

(iii) Abort the OSI connection.

It is left to up to the implementer of the T-TD how detect a half-open TCP connection. However, because detecting a half-open TCP connection in an efficient way is a complex issue, this section offers some suggestions.

If a TCP connection is completely idle (i.e. no data is sent across the TCP connection in either direction) the TCP protocol will *not* detect an idle connection (i.e. a TCP/IP stack will not spontaneously abort a half-open TCP connection unless an attempt is made to send some data across it). If an application attempts to send some data over a half-open TCP connection, the TCP stack will *eventually* detect the fact that the TCP connection is half-open and abort the TCP connection. The amount of time needed for the TCP/IP stack to abort the TCP connection depends on the configuration of the TCP/IP and the history of the TCP connection (specifically, it depends on the number of attempted re-transmissions of the TCP packet and the time between successive retransmissions) but typically it takes about 90 seconds.

Many TCP/IP stack offer a *keep-alive* option for TCP connections, which essentially causes the TCP/IP stack to periodically send an "empty" data packet over the TCP connection which allows the TCP/IP stack to detect half-open connections. A T-TD implementation may choose to use this mechanism to detect half-open TCP connections. However, most TCP/IP stack send the "empty" data packet very infrequently (e.g. only once per two hours) and do not allow the application to change the frequency. In those cases the *keep-alive* option is not sufficient to meet Requirement61.

As an alternative, the T-TD may use the following mechanism to detect half-open TCP connections carrying TL1 sessions:

(a) For each encoding of TL1 over TCP, [SIF-AR-9903-034R2] specifies the format of a so-called *keep-alive probe* which is a message which the receiver shall silently discard upon reception.

(b) If the T-TD receives a TL1 message over a TCP connection, this proves that the TCP connection is not half-open.

(c) If the T-TD sends a TL1 message over a TCP connection, this message will serve to test whether or not the TCP connection is half-open. If the connection is in fact half-open, the TCP/IP stack will spontaneously abort the TCP connection approximately 90 seconds after the attempted sending of the TL1 message.

(d) When the T-TD has neither sent nor received data over a TCP connection for a period of approximately Half-Open-TCP-Connection-Detection-Time minutes minus 90 seconds, the T-TD may start sending *keep-alive probes* over that TCP connection periodically every approximately Half-Open-TCP-Connection-Detection-Time minutes minus 90 seconds. These *keep-alive probes* serve to test whether or not the TCP connection is half-open.

(e) When the T-TD either receives or sends a TL1 message over a TCP connection it immediately stops sending keep-alive probes.

This mechanism guarantees that a TCP connection is not idle for more than approximately Half-Open-TCP-Connection-Detection-Time minutes minus 90 seconds. Since it takes about 90 seconds for the TCP/IP stack to abort a connection after an attempt to send some data over a half-open TCP connection, this mechanism will detect a half-open TCP connection within approximately Half-Open-TCP-Connection-Detection-Time minutes.

7.3.11 Requirements for Detecting and Eliminating Inactive TCP Connections

An "inactive TCP connection" is a TCP connection over which no TL1 messages have been transmitted in either direction for a configurable amount of time (the "Inactive-TCP-Connection-Detection-Time"). Note that an inactive TCP connection is different from a half-open TCP connection as defined in section 7.3.10. A half-open TCP connection occurs when the OS is disorderly shut down; an inactive connection occurs when neither the OS nor any of the remote NEs send any TL1 messages for a configurable amount of time.

R7-101 It shall be possible to provision "Inactive-TCP-Connection-Elimination". This parameter determines whether or not inactive TCP connections carrying TL1 sessions are detected and eliminated. The allowed values for this parameter shall be "On" and "Off". The default value shall be "On".

R7-102 It shall be possible to provision "Inactive-TCP-Connection-Detection-Time". This parameter determines the time (in minutes) a TCP connection shall be idle (i.e. no TL1 messages are sent in either direction over that TCP connection) before the connection shall be declared inactive. This parameter shall be provisionable in the range of 5 minutes to 10,080 minutes (one week). The default value shall be 60 minutes (one hour). The "Inactive-TCP-Connection-Detection-Time" may be optionally provisionable as an integer multiplier of the "Half-Open-TCP-Connection-Detection-Time".

Note: The accuracy of the "Inactive-TCP-Connection-Detection-Time" time period is not deemed critical for interoperability and can vary based on various vendor implementations. The vendor should provide accuracy information for their specific implementations.

R7-103 When a TCP connection is declared "inactive" and if "Inactive-TCP-Connection-Elimination" is set to "On" then the following actions shall be taken:

(a) Close the inactive TCP connection.

(b) Discard all remaining TL1 messages in the T-TD and TL1-Agent queues and remove the T-TD and TL1-Agent queues.

(c) For every OSI connection mapping to that TCP connection:

(i) Discard all remaining TL1 messages in the OSI receive and send queues.

(ii) Remove the OSI receive and send queues.

(iii) Abort the OSI connection.

7.3.12 Address Resolution and Access Control Requirements.

R7-77 The T-TD shall provide a mechanism to resolve a TID to an OSI presentation address and application context identifier. This mechanism shall be compatible with NEs which support only TARP.

Using a TARP initiator on the T-TD is one obvious mechanism which satisfies this requirement. However, the vendor may implement another mechanism which is compatible with TARP NEs. For example, the T-TD may query an X.500 directory for address resolution and an TARP-X.500 gateway (T5GW) may be used to populate TARP-only NEs into the X.500 directory.

Historically, the application context identifier for TL1 over OSI connections has been used to tailor the characteristics of the TL1 session (e.g. which events are reported using TL1 notifications) to the calling OS. In TL1/X25 - TL1/OSI gateways the calling address on an X25 connection (i.e. the address of the OS) was often used to determine which application context identifier should be used on the corresponding OSI connections. More recently, many NEs base the characteristics of a TL1 session on the username passed in the activate-user command rather than the application context identifier. Still, for backwards compatibility, the feature of having the application context identifier being determined by the address of the calling OS is required on the T-TD. To provide this feature, the requirement below calls for being able to provision a list of OSs which specifies the application context identifier to be used for each OS.

This list of OSs doubles as an access control list (hence the grouping of address resolution requirements and access control requirements into this one section). If the OS does not appear in the list the incoming TCP connection is refused.

R7-81 The T-TD shall provide access control and choose the application context identifier used on an OSI connection as follows:

(a) It shall be possible provision a list of OSs on the T-TD including the following information for each OS: the IP address of the OS and the application context identifier to be used on OSI connections for that OS.

(b) When an incoming TCP connection arrives, the calling IP address is used to determine which OS is calling and the corresponding application context identifier is used on all OSI connections for that OS.

(c) If the calling OS does not appear in the list, the incoming connection is refused (access control).

The above mechanism relies on IP addresses being provisioned on the T-TD. The usage of DNS hostnames is more user friendly and more appropriate for networks contains NATs (network address translators) but relies on the presence of DNS servers in the network.

R7-85 In the list of OSs, it shall be possible to identify an OS using either an IP address or a DNS hostname.

The T-TD needs a DNS client to translate the DNS hostnames in the list of OSs to IP addresses which can be compared with the calling IP address on an incoming TCP connection. For performance reasons, the T-TD may cache hostname to IP address translations but careful consideration shall be given to the fact that the IP address of an OS can change due to manual reconfiguration of the OS or due to the presence of dynamic NATs.

The T-TD may implement additional access control mechanisms, for example:

(a) Access control mechanisms based on the TID in the TL1 messages.

(b) Access control based on the IP address in each individual IP packets (as opposed to exercising access control only at connection setup time).

(c) Mechanisms to prevent specific denial-of-service attacks (e.g. SYN attacks).

(d) Generation of intrusion alert notifications.

7.4 Security Specifications

General security guidance for data communication networks and NEs is contained in GR-1332-CORE and GR-815-CORE. Individual service providers determine the security services to be used in their systems based on their individual needs. Since the TCP/IP protocol suite of protocols is well known, the following are considered to be the minimum security requirements applicable to T-TDs.

R7-87 OS access to T-TDs shall be secured via IP address screening performed at the T-TD.

Note that this requirement is in addition to (and does not replace) the existing requirement for SONET NEs to secure themselves via TL1 access control lists (i.e., based on userID from the ACT USER command).

R7-88 The control of access to the TDs shall be based on the host name. (It is expected that the Domain Name System will be used to support this requirement.)

R7-89 All TCP connection establishments to the TD shall be logged (both permitted and denied connection establishments).

CR7-90 If a local log file is used, then a mechanism shall be specified to periodically retrieve the log file by a remote OS before any log information is lost because the log file is too small to contain all entries.

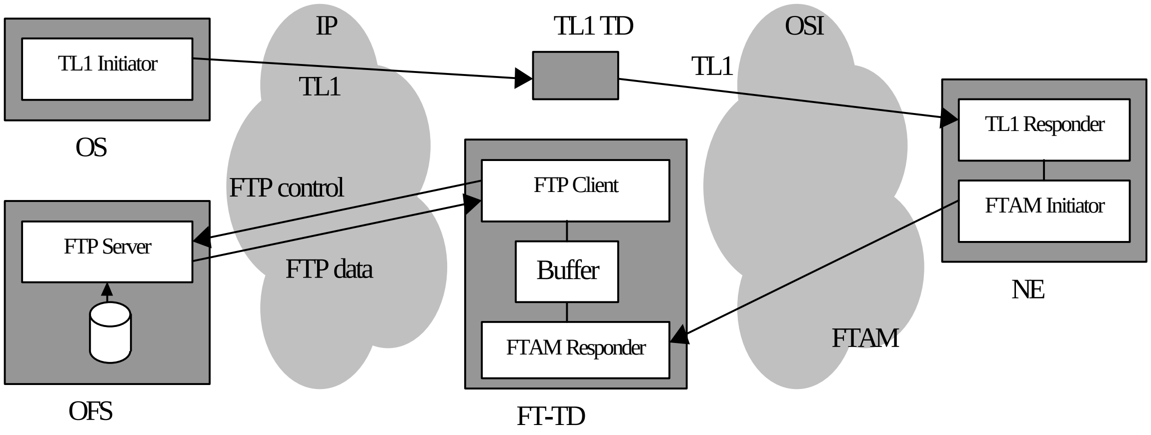
CR7-91 If the TD maintains a message log, then it will implement the NTP (or SNTP) protocol to allow synchronization of the time of day clock on the TD to a known reference standard.

Note: a vendor may choose to implement either of the above conditional requirements or both of these conditional requirements depending on the log file implementations.

O7-92 It is recommended that the Unix “syslog” protocol be used for real time logging instead of a local log file.

7.5 Specifications For File Transfer Translation Devices

Figure 7-4 depicts the FTP-FTAM File Transfer Solution.



Note1*:* The OFS and the OS may or may not be co-located. This solution allows the OFS and OS to be separate systems, but does not require them to be.

Note 2: The size of the buffer identified in the above figure should not be specified.

Figure 7-4: FTP-FTAM File Transfer Solution

Note: The TL1 manager is referred to in this document as the TL1 initiator; the TL1 Responder is referred to as the TL1 responder.

To send a file from the OS to the NE, the following scenario occurs:

1. The OS TL1 initiator sends a “retrieve file” TL1 command to the NE TL1 responder. This command includes the TID of the FT-TD, which is translated to the NSAP address of the FT-TD using TARP (or some other mechanism such as directory services), and the name and location (e.g., server and directory) of the file to be transferred using an FTP URL scheme.
2. The NE TL1 responder sends a TL1 notification to the OS TL1 initiator acknowledging the receipt of the command.
3. The NE FTAM initiator establishes an association with the FT-TD FTAM responder, retrieves the file, and closes the association.
4. The FT-TD FTAM responder retrieves the file data from the OFS using FTP. For this purpose the FT-TD FTP client sets up an FTP control connection to the OFS FTP server and sends a RETR command. This causes the OFS FTP server to setup a data connection over which the file data is sent.
5. The NE TL1 responder sends a TL1 autonomous notification to the OS TL1 initiator to report the result of the FTAM file transfer (success or failure).

To send a file from the NE to the OS, the following scenario occurs:

1. The OS TL1 initiator sends a “send file” TL1 command to the NE TL1 responder. This command includes the TID of the FT-TD, which is translated to the NSAP address of the FT-TD using TARP (or some other mechanism such as directory services), the name of the file to be transferred, and the name and location (e.g., server and directory) of the file destination using an FTP URL scheme.
2. The NE TL1 responder sends a TL1 notification to the OS TL1 initiator acknowledging the receipt of the command.
3. The NE FTAM initiator establishes an association with the FT-TD FTAM responder, sends the file, and closes the association.
4. The FT-TD FTAM responder relays the file data to the OFS using FTP. For this purpose the FT-TD FTP client sets up an FTP control connection to the OFS FTP server and sends a STOR command. This causes the OFS FTP server to setup a data connection over which the file data is sent.
5. The NE TL1 responder sends a TL1 autonomous notification to the OS TL1 initiator to report the result of the FTAM file transfer (success or failure).

The following specifications apply to File Transfer Translation Devices (FT-TDs). The file transfer process is composed of the following three elements:

1. TL1-based control messages from an OS to an NE to upload or download a file (with corresponding acknowledgements from the NE on the success or failure of the upload or download),
2. An FTAM-based file transfer between the FT-TD and a CLNP-based NE, and
3. An FTP-based file transfer between an IP-based OS and an FT-TD.

The specifications for the three components are contained in the following sections.

7.5.1 TL1-Based Control Messages

As specified in GR-1250-CORE, when files are to be retrieved or transferred by a NE, the managing system [OS] shall send an upload/download message to the NE (managed system) informing the NE to initiate an FTAM file transfer session. Included in the semantic of the upload/download message shall be the file name, the file type, and the file source. After an upload/download attempt, the SONET NE shall send a positive or negative acknowledgment to the managing system indicating the success or failure of the file transfer attempt.

R7-93 A TL1 message shall be used to inform an NE that it should initiate an FTAM session to upload/download a file. The message shall contain the file name, the file type, and the file source (i.e., TID of the FT-TD).

R7-94 The SONET NE shall use a TL1 acknowledgement message to inform an OS that a requested file transfer has succeeded or failed, as appropriate.

R7-95 Standardized TL1 messages (defined in Annex C) shall be used for all control and acknowledgement messages. All implementations shall support these standardized messages.

7.5.2 FTAM-Based File Transfer Between The FT-TD And The NEs

The GR-1250-CORE requirements for a SONET NE sending or receiving a file are unchanged by this process, i.e., NEs must support the FTAM initiator role profile specified in SIF-011-1997, *FTAM Profile for SONET Operations Communications*.

R7-96 FT-TDs shall support the FTAM responder role profile specified in SIF-011-1997. Note that in SIF-011-1997 the responder role is described as optional, however, for FT-TDs the responder role becomes mandatory.

7.5.3 FTP-Based File Transfer Between OSs And FT-TDs

The following specifications apply to the file transfer between an OS and an FT-TD.

R7-97 The FTP application will be used for file transfers between the IP-based OS and the FT-TD.

R7-98 The Passive Mode of FTP shall be used by the OSs to send or retrieve files to/from the FT-TD.

R7-99 Given the hostname of the FTP server in the URL parameter of the File Transfer TL1 command, the FT-TD shall be able to access a DNS server and determine the IP address of the host.

7.6 Management of the FT-TD

TL1 commands to accomplish management of the FT-TD are included in Annex C TL1 Control Messages for File Transfer

8 CMISE/FTAM Over TCP/IP

For future study.

9 IP End-To-End

For future study.

10 TCP/IP Protocol Specifications For The Access DCN

This section specifies the minimum number of Internet Standard Protocols required by the IP-based network. The objective of these specifications is to promote interoperability among the TDs, the IP-based DCN, and the OSs.

R10-2 The mandatory requirements of the following Internet standards shall apply to the TDs and the IP-based DCN:

1. STD0003 -- Host Requirements. (Also RFC1122, RFC1123),
2. STD0005--Internet Protocol. (Also RFC0791, RFC0950, RFC0919, RFC0922, RFC792, RFC1112),
3. STD0006-- User Datagram Protocol. (Also RFC0768),
4. STD0007-- Transmission Control Protocol. (Also RFC0793),
5. STD0008 -- Telnet Protocol (Also RFC0854, RFC0855),
6. STD0009--File Transfer Protocol. (Also RFC0959),
7. STD0013-- Domain Name System. (Also RFC1034, RFC1035),
8. (OPTIONAL) STD0015-- Simple Network Management Protocol. (Also

RFC1157),

9. (OPTIONAL) STD0016-- Structure of Management Information. (Also RFC1155, RFC1212),

STD0041-- Standard for the transmission of IP datagrams over Ethernet Networks. (Also RFC0894).

CR10-3 The support for the mandatory requirements of Internet STD0051, The Point-to-Point Protocol (PPP), RFC1661 and RFC1662, may be required for TDs and the IP-based DCN.

CR10-4 If SNMP is supported, then the mandatory requirements of the draft Internet standard, RFC 1213, Management Information Base For Network Management Of TCP/IP-Based Internets: MIB-II, shall apply to the TDs and the IP-based DCN.

R10-5 The SIF IP-based Network profile (see Annex A and Annex B) shall apply to the TDs and the IP-based DCN.

11 IP-Network Administration Support Specifications For The Access DCN

The following are IP-based network administration support specifications that must be supported by TDs.

CR11-2 If SNMP is supported, then UDP Port Unreachable, (Traceroute), shall be supported in the IP-based network.

O11-3 DHCP clients should be supported in the IP-based network.

R11-4 DNS clients shall be supported in the IP-based network.

CR11-5 If SNMP is supported, then SNMP clients shall be supported for IP network administration in the IP-based network.

Annex A: TL1 over TCP/IP Protocol Profile

This annex defines the TL1/TCP/IP protocol profile. This profile is based on the use of Internet Protocols defined by the Internet Architecture Board (IAB). The protocol stack is shown in Figure A-1 and uses the following:

- For Application Layer – TL1 length-value encoding, raw encoding, and telnet encoding.
- For Layer 4 – STD0007 “Transmission Control Protocol”, J. September 1981. (includes RFC0793.)
- For Layer 3 – STD0005 “Internet Protocol”, J. September 1981. (Includes RFC0791, RFC0950, RFC0919, RFC0922, RFC792, RFC1112).
- The lower layers are not specified.

	TL1 length-value, raw, telnet encoding
Layer 4	TCP (STD0007, RFC0793)
Layer 3	IP (STD0005) Subnetwork Access Protocol (not specified)
Layer 2	Not specified
Layer 1	Not specified

Figure A-1: TL1/TCP/IP Protocol Profile

Annex B: FTP over TCP/IP Protocol Profile

This annex defines the FTP/TCP/IP protocol profile. This profile is based on the use of Internet Protocols defined by the Internet Architecture Board (IAB). This Annex B is to be further defined in a second phase of this document. The protocol stack is shown in Figure B-1 and uses the following:

- For Application Layer – STD0009 “File Transfer Protocol”, J. October 1985. (includes RFC0959.)
- For Layer 4 – STD0007 “Transmission Control Protocol”, J. September 1981. (includes RFC0793.)
- For Layer 3 – STD0005 “Internet Protocol”, J. September 1981. (Includes RFC0791, RFC0950, RFC0919, RFC0922, RFC792, RFC1112).
- The lower layers are not specified.

	FTP (STD0009, RFC0959)
Layer 4	TCP (STD0007, RFC0793)
Layer 3	IP (STD0005) Subnetwork Access Protocol (not specified)
Layer 2	Not specified
Layer 1	Not specified

Figure B-1: FTP/TCP/IP Protocol Profile

Annex C: TL1 Control Messages for File Transfer

C.1   TL1 Command Definition for Remote File Transfer

This section defines the TL1 command/message set for use in remote file transfers. The TL1 commands/messages are **COPY-RFILE** and **REPT^EVT^FXFR** to support the general file transfer applications, and **DLT-RFILE** and **RTRV-RFILE** to support multi-stage capabilities (i.e., take advantage of store-and-forward capabilities). Refer to **Figure C1** for the file transfer reference architecture.

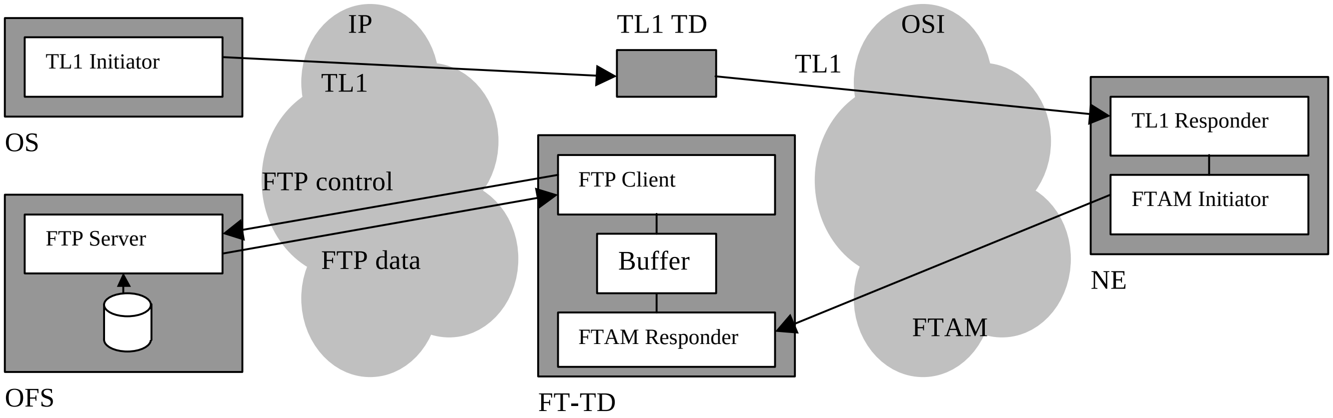


Figure C-1: Reference Architecture for File Transfer TL1 Commands

C.1.1 COPY-RFILE

The TL1 command **COPY-RFILE** initiates (i.e., starts) the file transfer from the OS/OFS to a target NE or from the target NE to the OS/OSF. This command can be used to copy a file directly from the remote server or through a FT-TD. To achieve the goal of interoperability in this command is independent of the file transfer method implemented on the FT-TD. This command creates directories as necessary if they do not exist.

When an NE initiates an FTAM transfer that will be converted to FTP by an FT-TD, it will need ***two sets of "userid:password" pairs***, one for the user ID on the **FT-TD** and one for the user ID on the **FTP fileserver or the FTAM fileserver**.

The use of the **RFILE** modifier identifies the command as a Remote File copy command and should prevent conflicts with **COPY-FILE** or **CPY-FILE** which may already be implemented in vendor’s equipment.

INPUT FORMAT

Command form with optional keywords:COPY-RFILE:tid:[[AID=]<aid>]:ctag::[TYPE=]<xfertype>,[SRC=]<srcurl>,[DEST=<desturl>],[OVWRT=<overwrite>],[FTTD=<fttdurl>];

Command form without optional keywords:

COPY-RFILE:tid:[<aid>]:ctag::<xfertype>,<srcurl>,[<desturl>],[<overwrite>],[<fttdurl>];

Input parameter definitions (Note: URL parameters are defined in more detail in a later section):

<aid> An optional AID is provided to allow specification of a target entity within an NE such as a drive partition or different areas of memory.

<xfertype> Specifies the type and direction of the file transfer. Valid values are:

SWDL - SW Download (Remote Server to TNE)

RFR - Remote File Restore (Remote Server to TNE)

RFBU - Remote File Backup (TNE to Remote Server)

<srcurl> Specifies the source of the file to be transferred. This parameter has the following forms:

<ftamurl> | <ftpurl> | <fileurl>

<desturl> Specifies the directory/file destination to store the transferred files. If not provided, then the destination directory/file defaults to the directory identified defined by <srcurl>. This parameter has the following forms:

<ftamurl> | <ftpurl> | <fileurl>

Note: All url definitions shall be enclosed in double quotes per Bellcore GR-831-CORE when entered in a TL1 command because they contain special characters.

<ftpurl> Specifies an FTP URL.
“ftp:[//[<userid>[:<password>]@]<ftphost>[:<port>]]/<url-path>”

<ftamurl> Specifies an FTAM URL.

“ftam://<userid>:<password>@<ftamhost>/<url-path>”

<fileurl> Specifies a File URL.

“file://[<localhost>]/<url-path>”

<fttdurl> Specifies the file transfer translation device (FT-TD)

“ fttd://[<userid>[:<password>]@]<fttdhost”

Note: If not specified, then the file transfer is direct (e.g. Remoteserver to NE or NE to Remoteserver.)

The following are common parameters used in **<ftpurl>, <ftamurl>, <fileurl>**, and **<fttdurl>** where applicable:

userid – The user identification for the host in the URL.
password – The user password for the host in the URL.
ftphost - The fully qualified domain name of a network host of the **ftp** function. The use of IP addresses as defined in RFC-1738 will be allowed.
localhost – specifies the local host (i.e. the NE that the TL1 command was targeted to)
ftamhost – specifies the **<tid>** or <**nsap_address>** of a network host for the **ftam** function.
fttdhost – specifies the **<tid>** or <**nsap_address>** of a network host for the **fttd** function.
tid - specifies the Target IDentifier. The tid is composed of <= 20 ASCII letters or numbers per BELLCORE GR-831-CORE, Sec 3.3.1.
nsap_address- specifies the Network Elements network address. Enter 16 to 40 hexadecimal digits, with optional hyphen ("-") delimiters that are ignored on input.
port - The port number to connect to. Most schemes designate protocols that have a default port number. Another port number may optionally be supplied, in decimal, separated from the host by a colon. If the port is omitted, the colon is as well.
url-path - The rest of the locator consists of data specific to the scheme, and is known as the "url-path". It supplies the details of how the specified resource can be accessed. The url-path has the following syntax:

<cwd1>/<cwd2>/.../<cwdN>/<name>;[type=<typecode>]

Where **<cwd1>** through **<cwdN>** are strings that identify directory levels, and **<name>** is a string that identifies the file name to be accessed.
The **<cwd1> - <cwdN>** portion of the url-path is interpreted as a series of FTP commands as each of the cwd elements is to be supplied, sequentially, as the argument to an FTP CWD (change working directory) command. This eliminates the directory deliminator (e.g., '\' and '/') problem.
The optional **<typecode>** field identifies the type of file to transfer where valid values are:
a - for ascii or text
i - for image or binary
The default, if the typecode is omitted from the url-path is image (i).

<overwrite> Indicates whether or not the files should overwrite

an existing file located at the destination. This parameter has the following values:

YES Overwrite existing file of same directory
location and filename.

NO (Default) Do not overwrite file, fail the file transfer if file already exists.

RESPONSE FORMAT:

```
<header>crlf
M^^<ctag>^COMPLDcrlf
^^^/* The target NE has initiated the file transfer process */
^^^/* Autonomous message(s) will follow to provide the completion status of the
file transfer */
;
```

Note: The above response message indicates the completion of the initiation of the file transfer process. Completion of the file transfer is reported in the autonomous message REPT^EVT^FXFR.

Error Responses:

When source file or directory that is specified in the <srcurl> does not exist, the following error response is required:

```
<header>crlf
M^^<ctag>^COMPLDcrlf
[^^^/* The file or directory does not exist */]
;
```

When userid or password specified in the <srcurl> or <desturl> or <fttdurl> does not have permission for the associated resource the following error response is required:

```
M^^<ctag>^DENYcrlf
[^^^PIUIcrlf;]
[^^^/* Permission denied for requested action*/]
;
```

EXAMPLES:

The following are COPY-RFILE examples to upload/download files:

1. SW download to a TNE (NE1) via a FT-TD (FTTD1) that requires a user and password and uses an NSAP address as the fttdhost:

   ```
   COPY-RFILE:NE1::ctag1::TYPE=SWDL,
       SRC="ftp://user1:passwd1@host1:21/dir1/file1":
       DEST="file://NE1/swdl/dir1/file1",FTTD=”fttd://user2:pw2@39840F80AAAAAABBBB
   CCCCDDDDEEEEEEEEEEEEAB”;
   ```

2. SW download to a TNE (NE1) via a FTTD (TID=FTTD1TID) that requires no user and password:

   ```
   COPY-RFILE:NE1::ctag1::TYPE=SWDL,
       SRC=”ftp://user1:passwd1@host1:21/dir1/file1”,
       DEST="file://NE1/swdl/dir1/file1",,FTTD=”fttd://@FTTD1TID;
   ```

3. File backup from a TNE (NE1) via a FTTD (TID=AB345678901234567890) that has the maximum of 20 characters for the TID without using optional keywords:

   ```
   COPY-RFILE:NE1::ctag1::RFBU,”file://NE1/dir1/file1”,
       ftp://user1:passwd1@host1:21/dir1/file1,,”fttd: AB345678901234567890;
   ```

4. Direct SW Download to a TNE (NE1) with no FT-TD:

   ```
   COPY-RFILE:NE1::ctag1::TYPE=SWDL,
       SRC=”ftp://user1:passwd1@host1:21/dir1/file1”,DEST="file://NE1/dir1/file1";
   ```

5. Download SW to a FT-TD (FTTD1) for multi-stage to multiple TNEs:

   ```
   COPY-RFILE:FTTD1::ctag1::TYPE=SWDL,
       SRC=”ftp://user1:passwd1@host1:21/dir1/file1”,DEST="file://NE1/dir1/file1";
   ```

6. SW download to a TNE (NE1) via a FT-TD (FTTD1) that is all FTAM:

   ```
       COPY-RFILE:NE1::ctag1::TYPE=SWDL,
       SRC="ftam://user1:passwd1@ftamhost1/dir1/file1",
       DEST="file://NE1/swdl/dir1/file1";
   ```

C.1.2 RTRV-RFILE

The RTRV-RFILE command is used to retrieve a list of files on the NE. The command will also return the storage usage and available (free) storage information. If no parameters are specified, then the NE will only return the storage usage information.

INPUT FORMAT:

```
RTRV-RFILE:tid:[[AID=]<aid>]:ctag:::[LOCN=<fileurl>]
[,LEVEL=<level>][,STORUSG=<storusg>];
```

Input parameter definitions:

<aid> The optional AID parameter is provided to accommodate

architectures that divide file store into multiple storage locations (e.g., primary and secondary).

<fileurl> Identifies a File URL. This parameter is used to specify a file or

directory.

<level> Identifies whether to retrieve the contents of the specified directory or all . This parameter would only apply when a <dirurl> is specified as the location. This parameter has the following values:

1 – (Default) Indicates to retrieve at the current (single) level.

ALL – Indicates to recursively retrieve all levels

<storusg> Indicates whether to return the storage usage and available free storage of the storage entity. This parameter has the following values:

YES – (Default) Return the storage usage/free information.

NO – Do not include the storage usage/free information.

RESPONSE FORMAT:

```
<header>crlf
M^^<ctag>^COMPLDcrlf
^^^["[<aid>]:SIZE=<storsize>,USED=<storused>,FREE=<storfree>"]crlf
^^^["[<aid>]:<filename>,<filesize>,<date>"]*crlf
;
```

Response parameter definitions:

<aid> The optional AID parameter is provided to accommodate

architectures that divide file store into multiple storage locations (e.g., primary and secondary).-

<storsize> Indicates the capacity of the storage device.

<storused> Indicates the amount spaced used on the storage device.

<storfree> Indicates the amount of free space available on the storage device.

<filename> Identifies the name of the file

<filesize> Identifies the size of the file in kbytes

<date> Identifies the date the file was last modified

Error Response

When file or directory that is specified in the <fileurl> does not exist, the following error response is required:

```
<header>crlf
M^^<ctag>^COMPLDcrlf
[^^^/* The file or directory does not exist */];
```

EXAMPLES:

The following are RTRV-RFILE examples:

1. Retrieve only the storage usage information for STOR1:

```
RTRV-RFILE:tid:STOR1:ctag;
Or
RTRV-RFILE:tid:STOR1:ctag:::STORUSG=YES;
```

2. Retrieve ALL from the directory root1/dir1/dir2 from STOR1:

```
RTRV-RFILE:tid:STOR1:ctag:::LOCN=”file://root1/dir1/dir2”,LEVEL=ALL;
```

3. Retrieve the RFILE “root1/dir1/file1” from STOR1:

```
RTRV-RFILE:tid:STOR1:ctag:::LOCN=”file://root1/dir1/file1”;
```

C.1.3 DLT-RFILE

The DLT-RFILE command is used to remove files from an NE. The command should also remove all directories once all associated files have been removed. The use of this command should be restricted to removing 'temporary' files transferred or created on the NE.

INPUT FORMAT:

```
DLT-RFILE:tid::ctag::[FILE=]<fileurl>;
```

Input parameter definitions (Note: URL type parameters are defined in a later section):

```
<fileurl>        Identifies a File URL.
```

RESPONSE FORMAT:

```
<header>crlf
M <ctag> COMPLDcrlf
;
```

Error Responses

When file or directory that is specified in the <fileurl> does not exist, the following error response is required:

```
<header>crlf
M^^<ctag>^COMPLDcrlf
[/* The file or directory does not exist */]
;
```

When userid or password specified in the <fileurl> does not have permission to delete the specified file the following error response is required:

```
<header>crlf
M^^<ctag>^DENYcrlf
[^^^PIUIcrlf;]
[^^^/* Permission denied for requested action*/]
;
```

EXAMPLES:

Delete file /NE1/swdl/dir1/file1 from NE1:

```
DLT-RFILE:NE1::ctag::FILE="file://NE1/swdl/dir1/file1";
```

C.1.4 REPT^EVT^FXFR

A file transfer is initiated (i.e., started) on a target NE by using the COPY-RFILE TL1 command. The REPT^EVT_FXFR autonomous message is used to notify the user of the file transfer completion status.

When the transfer is completed, a final message will be displayed to indicate completion of the transfer and results.

RESPONSE FORMAT:

```
    <header>crlf
    A^^atag^REPT^EVT^FXFRcrlf
    [^^^“filename,fxfr_status[,fxfr_rslt][,bytes_xfrd]”crlf];
```

Response parameter definitions:

<filename> Identifies the name of the file transferred.

<fxfr_status> Indicates the file transfer status as follows:

START Indicates file transfer has started.

[IP] Indicates file transfer is In-Progress.

COMPLD Indicates file transfer has completed.

<fxfr_rslt> Indicates success or failure of the file transfer. This

parameter is only displayed when the file transfer has completed (i.e., fxfr_status is COMPLD). The valid values are as follows:

SUCCESS Indicates successful file transfer.

FAILURE Indicates failed file transfer.

<bytes_xfrd> Number of bytes transferred for IP and total number of bytes for COMPLD

C.2 URL Formats

This section defines the various URL formats used in the TL1 commands defined above. The formats of the URLs are based on **RFC1738**.

Note: Per Bellcore GR-831-CORE, Sec 2.2.5, all of the URL formats below shall be enclosed in double quotes because of the use of special characters.

C.2.1 FTP and FTAM URLs

Section 7.5 of this document describes a method to download files from an OS/(File Server) to remote NEs via a FT-TD. To accomplish this task, the OS shall indicate to the GNE/NE file name and security information. Specifically, the OS shall forward to the GNE/NE the name of the file to be transferred, any directory information identifying the location of the file, hostname of the server where the file is located, and user ID and password used to access the server. A method for the OS to provide this information is to use an FTP (FTAM) URL scheme as defined in **RFC1738**[3] . This Annex briefly describes the FTP (FTAM) URL as it applies to file transfer. The format of the FTP (FTAM) URL is as follows:

"ftp://<userid>:<password>@<ftphost>:<port>/<url-path>"

"ftam://<userid>:<password>@<ftamhost>/<url-path>"

Refer to Annex C.1.1 for definitions of the URL parts.

C.2.2 FILE URL

The file URL is used to designate a file or directory for the source (SRC) or destination (DEST) that is identified in the TID. The format of the file URL is as follows:

"file://[<localhost>]/<url-path>"

For the file URL, <localhost> can be specified or omitted. If <localhost> is omitted, the host is assumed to be the local host, i.e. the host identified by the **TID** in the TL1 command. The implicit host is the Target NE.

C.2.3 FTTD URL

The FTTD (fttdurl) URL identifies the "userid:password" and fttdhost identifier for the FT-TD.

This permits either moderate or lax FTAM authentication, according to the policy of the FT-TD system administrator. Note that to have strong authentication, it would be necessary to encipher passwords with a nonrecurring value, because otherwise the password appears in clear text in the FTAM PDU. The password and userid information has been derived from ISO 8571-4: 1988, Part 4: File Protocol Specification (FTAM), Section Five, 20.3 ASN.1 Module Definition, Figure 7 – FTAM regime PDUs, lines 26, 27 and 28.

The format of this parameter is as follows:

<fttdurl> := "fttd:// [<userid>[:<password>]@]<fttdhost>"

<fttdhost> ::= <TID> | <NSAP>

<userid> ::= graphicString

-- value to use for the optional "initiator-identity"

-- parameter of the F-INITIALIZE-request service.

-- If absent, the "initiator-identity" parameter is omitted.

[3] According to RFC2396, separate RFCs are to be developed to define each URL scheme that will replace RFC1738. At the time of writing this document, no RFC for FTP URL, other that RFC1738, was identified.

```
<password> ::= graphicString
        -- value to use for the optional "filestore-password"
        -- parameter of the F-INITIALIZE-request service.
        -- If absent, the "filestore-password" parameter is omitted.
```

Appendix A: Contributors to the Document

Bob Arnston	Tellabs
Mike Brown	SBC
Richard Dunsmore	Fujitsu
Lester Ginsburg	Vertel
Alec Hothan	NEC
Chris Hunt	Lucent Technologies
Rashid Jamal	Sprint
Tom Kavanagh	Tellabs
Kip Klish	Nortel Networks
Jeff Learman	ONE
Richard Miller	Tellabs
Jerome Moisand	Lucent Technologies
Kevin Mooney	Tellabs
Jon Neubaum	Tellabs
Alan Repech	Cisco
Bruno Rijsman	Lucent Technologies
Ron Roman	Telecordia
Junji Tanabe	NEC
Linda Wu	Fujitsu