The Consultative Committee for Space Data Systems

Recommendation for Space Data System Standards

SPACE COMMUNICATIONS PROTOCOL SPECIFICATION (SCPS)—

TRANSPORT PROTOCOL

(SCPS-TP)

RECOMMENDED STANDARD

CCSDS 714.0-B-2

BLUE BOOK

October 2006

AUTHORITY

Issue:	Recommended Standard, Issue 2
Date:	October 2006
Location:	Washington, DC, USA

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS Recommendations is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:
  - -- The **standard** itself.
  - -- The anticipated date of initial operational capability.
  - -- The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures as defined in reference [B1]. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 714.0-B-1	Space Communications Protocol Specification (SCPS)—Transport Protocol (SCPS-TP)	May 1999	Original issue, superseded
CCSDS 714.0-B-2	Space Communications Protocol Specification (SCPS)—Transport Protocol (SCPS-TP), Recommended Standard, Issue 2	October 2006	Current issue: – adds optional support for Selective Acknowledgements (SACK) and Explicit Congestion Notification (ECN); – defines semantics to extend SCPS-TP signaling to allow optional inclusion of vendor- and community-specific options; – clarifies some ambiguities in the original specification regarding: • inclusion/position of compressed short-form SNACK options in compressed SCPS-TP headers (the position of the compressed SNACK option within the header is given); • what connection identifier is transmitted in compressed headers (the one sent to the peer during the SYN exchange); • what N-user protocol ID is used to calculate the pseudo header checksum when header compression is in use (decimal 105).

NOTE – Revision bars in the inside margin indicate changes from the previous issue.

CONTENTS

CONTENTS (continued)

Figure Page

Table

1 INTRODUCTION

1.1 PURPOSE

The purpose of this Recommendation is to define the services and protocols that provide the Space Communications Protocol Specification (SCPS) Transport service. This definition will allow independent implementations of the protocols in the space and ground segments of the SCPS Network to interoperate.

1.2 SCOPE

This Recommendation is intended to be applied to all systems that claim conformance to the SCPS Transport protocols.

1.3 APPLICABILITY

This Recommendation is designed to be applicable to any kind of space mission or infrastructure, regardless of complexity. It is intended that this Recommendation become a uniform standard among all CCSDS Agencies.

1.4 RATIONALE

The CCSDS believes it is important to document the rationale underlying the recommendations chosen, so that future evaluations of proposed changes or improvements will not lose sight of previous decisions. The concept and rationale for SCPS-TP may be found in reference [B2].

1.5 ORGANIZATION OF THIS RECOMMENDATION

This Recommendation contains six sections and four annexes. This section presents introductory material that establishes the context for the remainder of the document. Section 2 contains an overview of the protocols, summarizing the main technical requirements and describing the approach used to provide the protocols' services. Sections 3 and 4 present the specifications for Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) in the SCPS environment. Section 5 establishes the requirements for maintaining management information. Section 6 presents conformance requirements for implementations.

The four annexes to this Recommendation provide supporting information. Some of the annexes contain normative material, while some contain informative material. Annex A is informative and contains the acronyms and abbreviations used commonly throughout the document. Annex B is informative and contains the informative references cited throughout the document. Annex C is normative and contains the proforma for the Protocol Implementation Conformance Statement (PICS). The PICS unambiguously describes the capabilities provided by an implementation of the protocol. Annex D is normative and contains the service specification.

1.6 HOW TO READ THIS DOCUMENT

This document makes modifications and extensions to TCP and UDP for use in spacecraft communications environments, characterized by potentially long delays, unbalanced forward- and return-link data rates, and potentially high error rates. It is anticipated that some readers of this document will be protocol implementers, probably with TCP implementation experience. Other readers will be individuals more familiar with the particular application environment than with the protocols.

For readers and implementers already familiar with the internals of TCP and UDP, this document may best be used in the following manner:

1) Review section 6 of this document. It describes the implementation requirements for TCP and UDP, and gives an indication of those capabilities within TCP and UDP that have been modified (indicated by text to the effect 'as amended by a.b.c of this document', where a.b.c is a section reference in this document).

   Also in section 6 is a list of mission-specific capabilities that, depending on the needs of the mission, may be beneficial to add to the basic functionality of the protocols. Section 6 provides an introduction to these capabilities, and pointers back to sections 3 and 4, in which the capabilities are specified. Readers should use section 6 as a means of identifying the capability set required for their mission(s).

   Some of the capabilities required for a mission depend on the availability of other capabilities. These dependencies, along with a restatement of the implementation requirements, are documented in table form in the Protocol Implementation Conformance Statement that appears in annex C. Annex C specifies the format in which an implementer must document the details of his or her implementation.

2) Review sections 3 and 4, which specify SCPS-TP-unique options and modifications for TCP and UDP.

3) Review section 5, which identifies the management information requirements.

For readers and implementers who are generally familiar with the operation of TCP and UDP, but not the internals of the protocols, the following approach to reviewing this document may be useful:

1) Read over the UDP and TCP RFCs. The UDP specification is quite short (3 pages). The TCP specification is significantly longer, but also provides a substantial amount of background information. These documents were written in 1980 and 1981, respectively.

2) Read section 4 of RFC 1122. This document captures the 'lessons learned' from using TCP and UDP as of 1989. In addition to placing requirements on implementations of TCP and UDP, it provides a significant amount of explanatory information and discussion about rationale for particular requirements. RFC 1122 constitutes an extension to the base TCP and UDP specifications, in addition to describing implementation requirements.

3) Read over section 6 of this document. It refers to the TCP and UDP RFCs and to RFC 1122, and either endorses or revises the requirements put forward in RFC 1122. Read the 'Mission-Specific Capabilities' subsection, and identify whether any of these capabilities are necessary. If so, review the references identified in each of the mission-specific capability sections.

4) Finally, read sections 3 through 5.

Readers with little previous familiarity with TCP or UDP should consider reviewing an introductory text on the subject. One excellent example is TCP/IP Illustrated, Volume 1, by W. Richard Stevens (Copyright 1994, Addison-Wesley Professional Computing Series). Chapters 1, 11, and 17-24 are particularly relevant. Note that additional information about one of the mission-specific capabilities—the modifications to TCP to support Transactions—is presented in Chapters 1-12 of TCP/IP Illustrated, Volume 3, by W. Richard Stevens (Copyright 1996, Addison-Wesley Professional Computing Series).

1.7 CONVENTIONS AND DEFINITIONS

1.7.1 OCTET NUMBERING CONVENTION AND NOMENCLATURE

This document does not deal with transmission of any data elements smaller than one octet. As such, the transmission order of bits within an octet is an issue to be dealt with at lower layers. However, the relative ordering of octets within a word and the unambiguous numbering of bits within an octet are relevant here. The order in which multi-octet fields defined in this document are submitted for transmission is called 'Big Endian' byte ordering. When applied to networking, it is called 'network byte order'. In this ordering scheme, bit 0 of a 32-bit value is the most significant bit; bit 31 is the least significant bit. The octet containing bits 0-7 is transmitted first, followed by the octet containing bits 8-15, followed by the octet containing bits 16-23, and finally the octet containing bits 24-31. Note that 'Big Endian' byte ordering is NOT what some machines (notably the 80x86 class of machines) use internally. Implementers must ensure that headers are converted to network byte order for transmission.

The following conventions apply throughout this Recommendation:

a) the words 'shall' and 'must' imply a binding and verifiable specification;

b) the word 'should' implies an optional, but desirable, specification;

c) the word 'may' implies an optional specification;

d) the words 'is', 'are', and 'will' imply statements of fact.

1.7.2 DEFINITIONS

Address Family: An address family specifies the structural rules required to interpret the internal fields of an address. The SCPS Network supports three address families: the SCPS address family, the Internet Protocol (IP) address family, and the Internet Protocol version Six (IPv6) address family.

Address Type: An address type defines the meaning that the addresses have (that is, whether they identify end systems or a path between end systems), the number of addresses that appear in a SCPS Network Protocol header (two addresses if the addresses identify end systems, only one if the address identifies a path between end systems), and the address family that is valid for the addresses.

Connection: A connection is defined by information that is named, persistent, and shared across the systems supporting an instance of communication. For transport protocols, these systems are the endpoints that terminate the transport protocol, but not intermediate systems.

End System: An addressable network entity within the SCPS Network.

Extended End System Address: The Extended End System Address identifies a single end system or an end-system group. The Extended End System Address conforms to the structural rules of either the SCPS Address Family or the IP Address Family. Extended End System Addresses may be parameters to the primitives of the Unit Data service.

Gateway: A network-addressable system that terminates a protocol at a given layer and invokes similar services at the same layer of an adjacent network.

Host: A network-addressable system that may send or receive network-layer packets, but does not forward packets.

Internet Protocol Number: The Internet Protocol Number is the transport protocol identifier used by Internet Protocols. Values may range from 0 through 255, and valid values are defined in reference [1].

IP Address Family: The IP Address Family specifies a set of structural rules for the interpretation of Extended End System Addresses, and is defined in reference [1], and the possible formats are refined in section 3.2.1.3 of RFC 1122 (reference [2]).

Maximum Segment Size: The maximum amount of user data that can be carried in a Segment. This value is calculated by subtracting the size of the network, security, and transport layer headers from the MTU size.

Maximum Transmission Unit: The Maximum Transmission Unit (MTU) specifies the maximum amount of data that the subnetwork layer will accept in a single subnetwork service request. The MTU for a route is the minimum of all known MTUs along that route.

NOTE – It is anticipated that this value will be known and managed as part of the routing table information; however, techniques for dynamically discovering the MTU of a route exist. Refer to RFC 1191, 'Path MTU Discovery' (reference [B3]) for more information.

N-Address: an address in the SCPS Network. The attributes of an N-Address are the Address Type and the Address Family.

N-Basic_Quality_of_Service parameter: The Basic Quality of Service (QOS) parameter of the N-UNITDATA service primitives carries information necessary to provide special network processing services for the datagram. It is a data structure that contains three sub-parameters: precedence, routing requirements, and a program-specific field.

N-Destination_Address: The N-Destination_Address is a parameter of all of the SCPS Network service primitives. It is an N-Address that identifies the destination end system of a packet in the SCPS Network. The N-Destination_Address parameter must be of the Extended End System address type, and may be of either the IP or the SCPS address family.

Network-Service Data Unit: See N-SDU.

N-Expanded_Quality_of_Service parameter: The Expanded QOS parameter provides a mechanism for specifying ground-relevant QOS requests. The valid values of this parameter are defined in RFC 2474 (reference [B7]).

N-SDU: The Network Service Data Unit (N-SDU) is a parameter of the Unit Data service primitives. It is a variable-length, octet-aligned data unit of arbitrary format. The maximum length of an N-SDU is 8145 octets.

NOTE – The maximum size of the N-SDU field is limited to the length resulting from subtracting the maximum length of a SCPS-NP header from the maximum SCPS-NP PDU length. The maximum length of the SCPS-NP header is 46 octets. The length field in the SCPS-NP header is 13-bits, which allows an 8191-octet total packet length. Therefore, the maximum size of an N-SDU that is guaranteed to fit in a SCPS-NP PDU is 8145 octets. Local restrictions on packet size or extensions to the protocol may further limit this size, and the maximum implementations size of N-SDU must be documented by the implementer.

N-Source_Address: The N-Source_Address is a parameter to many of the primitives of the SCPS network service. It is an N-Address that identifies the end system originating a packet in the SCPS Network. The N-Source_Address must be of the Extended End System address type and may be of either the IP or the SCPS address family. The N-Source_Address may not be a multicast or a broadcast address.

N-Source_Timestamp parameter: The N-Source_Timestamp is a parameter of several SCPS Network service primitives. This parameter permits the Network Service user to provide a source timestamp to accompany the N-SDU. The Source Timestamp parameter consists of a timestamp format field and a timestamp value field.

N-User_Internet_Protocol_Number parameter: The N-User_Internet_Protocol_Number is a parameter to several of the SCPS Network service primitives. The values of this parameter used by the SCPS-TP are shown in table 1-1.

Table 1-1: Values of the N-User_Internet_Protocol_Number Parameter Used by SCPS-TP

Network Service User	Internet Protocol Number (decimal)
TCP	6
UDP	17
Compressed TCP	105

Port: An identifier of the transport service user.

Precedence parameter: The precedence parameter is an element of the N-Basic_Quality_of_Service parameter of the N-UNITDATA service primitives. The precedence parameter is specified by a network service user to identify the relative importance of this data compared to other data within the network. It is an integer with a valid range from 0 to 15, with 0 being the lowest precedence and 15 being the highest. Local policy may cause the user-specified precedence parameter to be overridden. The network service user may also supply a null value for the precedence parameter, in which case the network service would assign a default value for the precedence parameter.

Program Specific parameter: The program-specific parameter is an element of the N-Basic_Quality_of_Service parameter that provides a mechanism for programs to carry two bits of information in the SCPS-NP header. This information is interpreted by program-specific extensions to the SCPS-NP and has a default value of 0.

Pseudo-Header: A pseudo-header, in TCP and UDP, is a collection of information that is used for the purposes of checksum calculation, but not actually shipped as part of the transport layer protocol data unit. The information in the pseudo-header consists of the source and destination addresses, the Internet Protocol Number of the transport protocol, and the length of the transport protocol data unit.

Router: A network-addressable system that may send, receive, or forward network-layer packets.

Routing Requirements parameter: The Routing Requirements parameter is an element of the N-Basic_Quality_of_Service parameter of the N-UNITDATA service primitives. The Routing Requirements parameter has two currently defined values: ‘normal’ routing and ‘flood’ routing.

SCPS Network Address: A SCPS Network Address specifies one of the possible SCPS Address formats (via the FMT-ID parameter) and the values of the parameters required by that format.

Segment: A segment is the Protocol Data Unit of the Transmission Control Protocol (TCP).

Service-Access-Point: A Service-Access-Point (SAP) is the point at which the services of a layer are made available to the layer above it.

Silently Discard: A packet is 'silently discarded' if no error message is generated (either to a local user or to a remote user) as a result of the discard.

NOTE – The practice of silently discarding packets reduces the possibility that a misconfigured host will uncontrollably generate erroneous traffic. The term 'silent discard' differs from 'discard' in that certain actions, such as informing network service users about the discard, are not performed in a silent discard. When the term 'discard' is used, other information must be used to determine whether the network service user is informed.

T-SDU: The Transport Service Data Unit (T-SDU) is a parameter of several of the SCPS-TP service primitives. It is a variable-length, octet-aligned data unit of arbitrary format. The maximum length of a T-SDU is an implementation issue.

Timestamp Format field: The Timestamp Format field of the N-Source_Timestamp parameter identifies the format of the source timestamp that is supplied by the Network User. The available formats are specified in reference [10].

Timestamp Value: The Source Timestamp Value field of the N-Source_Timestamp parameter contains the value of the timestamp that shall accompany the Network Service Data Unit.

Transport-Service Data Unit: See T-SDU.

1.8 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommendations.

[1] J. Postel. *Internet Protocol*. STD 5, September 1981. [RFC 791, RFC 950, RFC 919, RFC 922, RFC 792, RFC 1112]†

[2] R. Braden. *Hosts Requirements*. STD 3, October 1989. [RFC 1122, RFC 1123]

[3] J. Postel. *User Datagram Protocol*. STD 6, August 1980. [RFC 768]

[4] J. Postel. *Transmission Control Protocol*. STD 7, September 1981. [RFC 793]

[5] D. Borman, R. Braden, and V. Jacobson. *TCP Extensions for High Performance*. RFC 1323, May 1992.

[6] P. Karn & C. Partridge. "Round Trip Time Estimation." In *Proceedings of SIGCOMM '87: Symposium on Communications Architectures and Protocols*, August 1987.

[7] V. Jacobson. "Congestion Avoidance and Control." In *Proceedings of SIGCOMM '88: Symposium on Communications Architectures and Protocols*, August 1988.

[8] J. Nagle. *Congestion Control in IP/TCP Internetworks*. RFC 896, January 1984.

[9] K. McCloghrie and M. Rose. *Management Information Base*. STD 17, March 1991. [RFC1213]

[10] *Space Communications Protocol Specification (SCPS)—Network Protocol (SCPS-NP)*. Recommendation for Space Data System Standards, CCSDS 713.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

[11] *Space Communications Protocol Specification (SCPS)—Security Protocol (SCPS-SP)*. Recommendation for Space Data System Standards, CCSDS 713.5-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1999.

[12] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. "TCP Vegas: New Techniques for Congestion Detection and Avoidance." In *Proceedings of SIGCOMM '94: Symposium on Communications Architectures and Protocols*, August 1994.

[13] R. Braden. *T/TCP—TCP Extensions for Transactions—Functional Specification*. RFC 1644, July 1994.

[14] R. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168, September 2001.

[15] M. Mathis, et al.. *TCP Selective Acknowledgement Options*. RFC 2018, October 1996.

† Internet Request for Comments (RFC) texts are available on line in various locations (e.g., http://ietf.org/rfc/). In this list, Internet Standards are identified by 'STD' followed by the number of the standard, and RFCs are identified by 'RFC' followed by the number of the RFC. RFCs comprised by Internet Standards are given in square brackets following the citation.

2 OVERVIEW

This SCPS Recommendation is designed to support current communication environments and those of upcoming missions. The modifications to the base protocols are intended to address the communication environments and resource constraints that space-based systems face.

The Technical Requirements for the Recommendation include:

- support for communication with full reliability, best-effort reliability, and minimal reliability;
- efficient operation in a wide range of delay, bandwidth, and error conditions;
- efficient operation in space-based processing environments;
- support for precedence (priority) based handling;
- support for connectionless multicasting;
- support for packet-oriented applications.

The SCPS Transport Protocol (SCPS-TP) refers collectively to the protocols that provide the full reliability, best-effort reliability, and minimal reliability services. The full reliability service is provided by TCP. The best-effort service is provided by TCP with minor modifications. The minimal reliability service is provided by UDP.

The SCPS-TP addresses the environmental requirements with the following extensions:

- TCP for Transactions (RFC 1644, reference [13])—reduces the handshaking necessary to start a TCP connection and provides 'reliable datagram' operation to handle command-response traffic, for very long delay environments in which it is desirable to begin data transfer without waiting for a connection handshake;
- Window scaling (RFC 1323, reference [5])—addresses communication environments that may have more than 65k octets of data in transit at one time;
- Round Trip Time Measurement (RFC 1323, reference [5])—addresses environments that have high loss, changing delays, or large amounts of data in transit at one time;
- Protect Against Wrapped Sequence Numbers (RFC 1323, reference [5])—addresses very long delay environments or very high bandwidth missions;
- Selective negative acknowledgment (adapted from RFC 1106, reference [B5])—addresses high loss environments;
- Selective acknowledgement (RFC 2018, reference [15])—addresses high loss environments;
- Record Boundary Indication—the ability to mark and reliably carry end-of-record indications for packet-oriented applications;

- Best Effort Communication—the ability for an application to select correct, in-sequence, but possibly incomplete delivery of data;
- Header compression (adapted from RFC 1144, reference [B6])—addresses low-bandwidth environments;
- Low-loss congestion control or optional non-use of congestion control;
- Explicit Congestion Notification (ECN)—can provide improved performance in ECN-capable networks;
- Retransmission strategies for space environments that accommodate loss due to data corruption, link outages, and congestion.

3 SCPS-TP EXTENSIONS TO STANDARD TCP

3.1 RELATIONSHIP BETWEEN SCPS-TP AND TCP

SCPS-TP adopts the Transmission Control Protocol (TCP) as specified in Internet Standard 7 (reference [4]) and its supporting RFCs, with the modifications and options specified in section 3 of this document.

NOTE – Section 6 of this document summarizes requirements for implementing TCP in the SCPS environment.

3.2 CONNECTION MANAGEMENT

3.2.1 INITIAL SEQUENCE NUMBER SELECTION

A SCPS-TP conforming implementation is not required to use a clock as the basis for Initial Sequence Number (ISN) selection. As long as ISN selection is robust against a possible crash, increases slightly faster than the maximum possible transmission rate, and does not wrap too quickly, then the algorithm used for ISN selection meets the intent of the requirement and is acceptable (refer to reference [4], RFC 793, section 3.3, and reference [2], RFC 1122, section 4.2.2.9).

NOTE – The ISN does not have to be updated at every clock tick. Rather, it only needs to be computed at the time a connection is established.

3.2.2 PRECEDENCE HANDLING

3.2.2.1 Security

The TCP shall convey a user's security requests and replies to the security provider and shall report responses and indications as required.

NOTES

1 Security is handled external to the TCP, at a protocol layer that is conceptually lower in the 'stack'.

2 All other references to security in this document shall be considered non-normative.

3.2.2.2 Precedence

3.2.2.2.1 The precedence parameter used in TCP is as defined in the SCPS Network Protocol (reference [10]), the intent being that connection shall be allowed at the higher of the precedence levels requested by the two ports attempting to connect.

3.2.2.2.2 If the TCP is operating over a network protocol that does not support precedence, or supports fewer precedence levels than are defined in [10], then a locally defined mapping between the user-specified precedence level and the system-supported precedence levels shall be performed.

3.2.2.2.3 The following paragraphs describe specific actions for handling the precedence parameter in SCPS-TP:

a) an endpoint shall request the precedence level specified by the calling application;

NOTE – Any local policy constraints on precedence level requested by the application are outside the scope of this Recommendation.

b) if the remote TCP responds with a higher precedence level, the precedence of the connection shall be raised to the higher requested level;

c) if local policy prohibits raising the precedence of the connection, a reset shall be sent;

d) a listening endpoint that receives a connection request containing a higher precedence level than the endpoint's configuration shall, if permitted by local policy, raise its precedence level to that specified by the remote endpoint and proceed with connection establishment;

e) if local policy does not permit such an increase in precedence level, the listening endpoint shall reject the connection with a reset;

f) if an endpoint has more than one socket with service requests to the transport protocol pending, the service order of those requests shall be determined by the precedence level of the socket and shall proceed starting with the highest precedence level.

3.2.3 NEGOTIATION OF SCPS CAPABILITIES

3.2.3.1 SCPS Capabilities Option Format

The SCPS Capabilities Option shall be located in the options area of the TCP SYN segment[1] header and shall contain the following fields:

	Length in octets
– Option Type (mandatory)	1
– Option Length (mandatory)	1
– Capabilities Option Bit-Vector (mandatory)	1
– Connection ID (mandatory)	1

[1] Refer to reference [B3] for a definition of the TCP SYN segment.

3.2.3.2 SCPS Capabilities Option Fields

3.2.3.2.1 The Option Type field

a) is mandatory and shall occupy the first octet of the Option;

b) shall contain the decimal value 20.

3.2.3.2.2 The Option Length field

a) is mandatory and shall occupy the second octet of the Option;

b) shall contain the decimal value 4.

3.2.3.2.3 The Capabilities Option Bit-Vector field

a) is mandatory and shall occupy the third octet of the Option;

b) shall indicate which SCPS Capabilities are in effect for a connection, as detailed in table 3-1.

3.2.3.2.4 The Connection ID field

a) is mandatory and shall occupy the fourth octet of the Option;

b) shall contain a non-zero Connection Identifier if the Com bit is set to '1'.

NOTE – The SCPS Capabilities Option format is illustrated in figure 3-1.

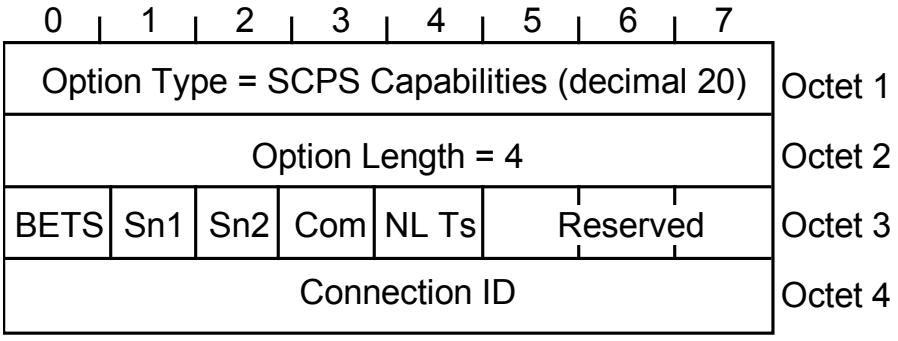

Figure 3-1: SCPS Capabilities Option

Table 3-1: SCPS Capabilities Option Bit-Vector Contents

Bit	Meaning if = 0 ('Not OK')	Meaning if = 1 ('OK')
BETS	Connection may not operate in BETS mode.	Sender willing to operate connection in BETS mode.
SN1	Do not send short form (length = 4) of SNACK Option (refer to 3.5).	OK to send short form of SNACK Option.
SN2	Do not send long form (length > 4) of SNACK Option (refer to 3.5).	OK to send long form of SNACK Option. Note: if the SN2 bit is set to '1' then the SN1 bit must also be set to '1'.
Com	Do not send compressed TCP headers.	OK to compress TCP headers—send connection identifier.
NL Ts	Network-layer timestamps not available or unsuitable for use in compressing TCP timestamps Option.	Network-layer timestamps available and a timestamp accompanies this segment. If received, suitable, and available at both ends, use for compressing TCP Timestamps Option.

NOTES

1. If both TCP endpoints send the BETS bit set to '1', the connection will operate in BETS mode.

2. The semantics for the combination of the Com bit and the connection identifier are as follows:

Com	Connection ID	Meaning
'0'	'0'	Will not send or accept compressed headers.
'0'	X≠'0'	Will not accept compressed headers. Would like to send compressed headers, using Connection ID X. (If SYN, will send compressed headers if and only if SYN-ACK contains Com bit set to '1'. If SYN-ACK, not a valid response if Com bit set to '0' on SYN.)
'1'	'0'	Will accept compressed headers, will not send compressed headers.
'1	X≠'0'	Will accept compressed headers, would like to send compressed headers, using Connection ID X. (If SYN-ACK, not a valid response if Com bit set to '0' on SYN.)

3. If compressed headers are in use and both TCP endpoints indicate that use of Network Layer Timestamps (NL Ts) is acceptable, then outbound timestamps shall be carried in the timestamps field of the Network-layer header.

3.2.4 SCPS CAPABILITIES INVOCATION

3.2.4.1 The TCP initiating a connection shall invoke SCPS Capabilities by including the SCPS Capabilities Option in the header of the SYN segment.[1]

3.2.4.2 The listening TCP shall indicate its willingness to use the specified capabilities by including the SCPS Capabilities Option in the header of its SYN ACK segment:

a) accepted capabilities shall be indicated with their corresponding Capabilities Option Bit-Vector bits set to '1';

b) if the Com bit is set to '1', the Connection ID field shall contain a non-zero value.

[1] Refer to reference [B3] for a definition of the TCP SYN segment.

NOTE – The determination of which capabilities to use on a particular connection is implementation specific, possibly based on information from the routing structures as specified in 5.2. An application MUST be involved in the decision to use the Best Effort Transport Service (BETS)—refer to 6.2.4.1.2.

3.2.4.3 If either endpoint does not send the SCPS Capabilities Option, the respective SCPS capabilities are unavailable on the connection.

3.2.4.4 If use of SCPS capabilities is declined by a listening TCP (by its failure to return a corresponding SCPS capabilities option after one was sent by the initiating TCP):

a) the connection may continue using standard TCP capabilities, if standard TCP capabilities have been implemented;

b) if standard TCP capabilities have not been implemented, the initiating TCP shall abort the connection by sending a RST segment.

3.2.4.5 If the SCPS Capabilities Option is not present on a SYN, then the protocol shall conform to the end-to-end semantics of Internet TCP, as specified in Internet Standards 7 and 3 (references [4] and [2], respectively), and possibly as extended by other RFCs.

NOTE – A connection using Internet (i.e., 'normal') TCP may include the non-SCPS options of window scaling, timestamps, RFC 1644 (reference [13]), fast retransmit, fast recovery, and so on.

3.2.4.6 If the SCPS Capabilities Option is present on a SYN but none of the bits are set to '1', then none of the capabilities in the Capabilities Option Bit-Vector will be used on the connection.

3.2.5 SCPS-TP EXTENDED CAPABILITIES OPTION

3.2.5.1 General

The following format is used to signal 'extended' SCPS capabilities. Extended capabilities allow endpoints to perform signaling in addition to that supported by the 'standard' SCPS Capabilities Option described above.

Extended capabilities are identified by reuse of the SCPS Capabilities Option (option 20) two or more times on a particular SYN packet (TCP packet with the SYN bit set). These extended option 20s are prohibited from having length = 4 to help differentiate them from 'standard' SCPS Capabilities Options. The *first* SCPS Capabilities Option present on a SYN segment must have length = 4 and is interpreted as above.

The intent of allowing extended capabilities is to allow vendors and communities of interest to implement features unique to particular environments. Each vendor or community of interest will be assigned a unique identifier.

3.2.5.2 Extended Capability Format

3.2.5.2.1 General

The first part of an extended capability signal is the presence of a second (third, etc.) SCPS Capabilities Option (TCP option 20). This option has the standard TCP option format, as shown in figure 3-2. The type of this option is SCPS Capabilities (20), and the length is variable, but prohibited from being 4 (the mandated length of the base SCPS Capabilities option above).

NOTE – The first SCPS Capabilities option on a SYN segment **must** be the 'standard' SCPS Capabilities Option defined in 3.2.3 with length = 4. It is not possible to have an extended SCPS capabilities option (length ≠ 4) without preceding it with a 'standard' SCPS capabilities option of length = 4, formatted as in section 3.2.3.

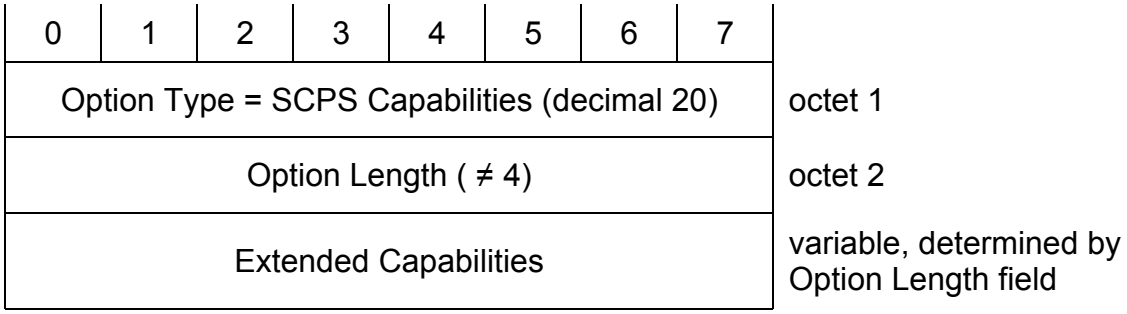

Figure 3-2: Beginning of Extended Capabilities Signaling

Option Length: The value of this field equals the total number of octets in all of the extended capability options signaled by this (extended) SCPS capabilities option, including octets 1 and 2. That is, the length field of extended SCPS capabilities options behaves as a normal TCP option length field. The value of the length field may **not** be equal to 4 to ensure that implementations that do not understand extended capabilities do not confuse an extended capability SCPS option with a 'standard' SCPS capabilities option (3.2.3).

Figure 3-3 shows the format for each individual extended capability.

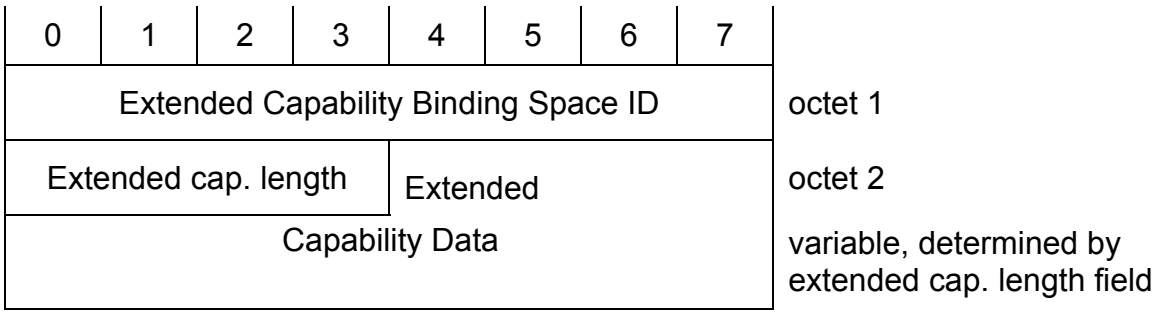

Figure 3-3: Format for Extended Capabilities

3.2.5.2.2 Extended Capability Fields

3.2.5.2.2.1 Extended Capability Binding Space ID

The extended capability binding space ID identifies the particular extended capability binding space. The extended capability data is interpreted in the context of this identifier.

The extended capability binding space identifier is an arbitrary integer that identifies the format of the extended capability data, which could contain information about one or more extended capabilities. For example, a particular vendor might use a single binding space identifier and signal a number of individual capabilities with flags in the extended capability data field. The format for identifying the particular capabilities would be completely vendor-specific. In general there would be at most one extended capability binding space ID per vendor or community of interest. If vendors or communities want to implement multiple extended capabilities, they will need to define their own signaling/multiplexing method for those capabilities within a particular extended capability *data* area. Vendors are strongly encouraged to make the format of their extended capabilities public knowledge.

NOTE – Assignment of extended capability binding space IDs is discussed in 3.2.5.4 below.

3.2.5.2.2.2 Extended Capability Length

The extended capability length field specifies the length of the extended capability, exclusive of the octets containing the extended capability type and length, in 16-bit words. Thus in figure 3-3 an extended capability length of '0' would indicate that only octets 1 and 2 in the figure were present, a value of '1' would indicate a total length of four octets (octets 1 and 2 in figure 3-3 plus two additional octets of extended capability data). A value of '2' would indicate a total length of six octets, etc.

NOTE – The minimum length of an extended capability is two octets (the octet containing the binding space identifier and the octet whose first four bits contain the length). The last four bits of the second octet are the first four bits of the extended capability data, and are always present, even if unused by the particular capability.

3.2.5.2.3 Support for Multiple Extended Capability Binding Spaces

More than one extended capability binding space identifier can be indicated under a single extended SCPS Capabilities Option, as shown in figure 3-4.

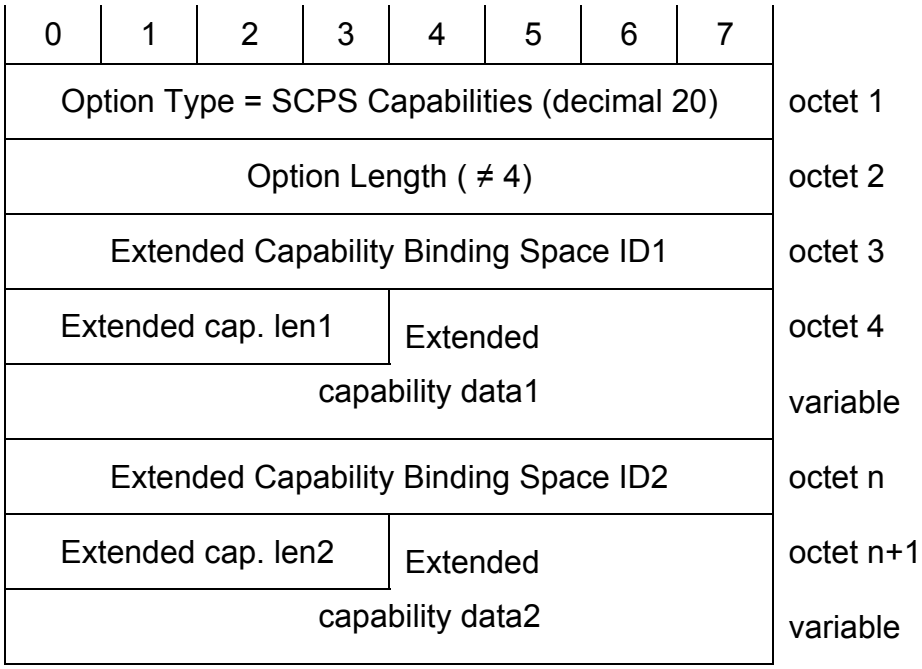

Figure 3-4: Single Extended SCPS Capabilities Option with Multiple Extended Capability Binding Spaces

3.2.5.2.4 Alternate Support for Multiple Extended Capability Binding Spaces

Although it is less bit-efficient than the method described in 3.2.5.2.3, multiple extended capabilities can also be signaled with additional SCPS Capabilities Options as shown in figure 3-5.

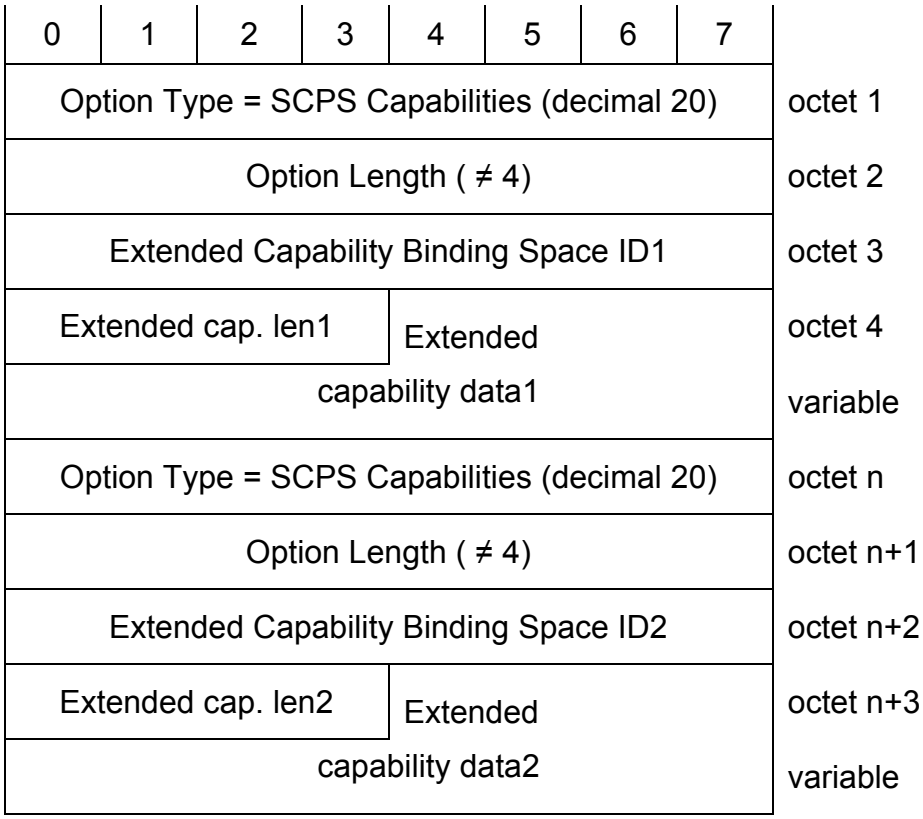

Figure 3-5: Using Multiple SCPS Capabilities Options to Express Multiple Extended Capabilities

3.2.5.2.5 Extending the Extended Capability Binding Identifier Space

An extended capability binding space identifier value of 255 decimal shall be used to extend the extended capability binding space from 255 to 511 values. A value of 255 in the extended capabilities binding space identifier field shall indicate that the actual extended capability binding space identifier is calculated by adding 256 to the value of the octet following the octet containing the extended capability length (octet 5 in figure 3-6 below). If octet 5 in figure 3-6 were itself 255, the extended capability binding space identifier would be 512 plus the value of the octet following octet 5.

NOTE – Using this extension method, the position of the extended capability length field is fixed relative to the start of the extended capability binding space identifier, regardless of the number of octets needed to express the extended capability binding space identifier.

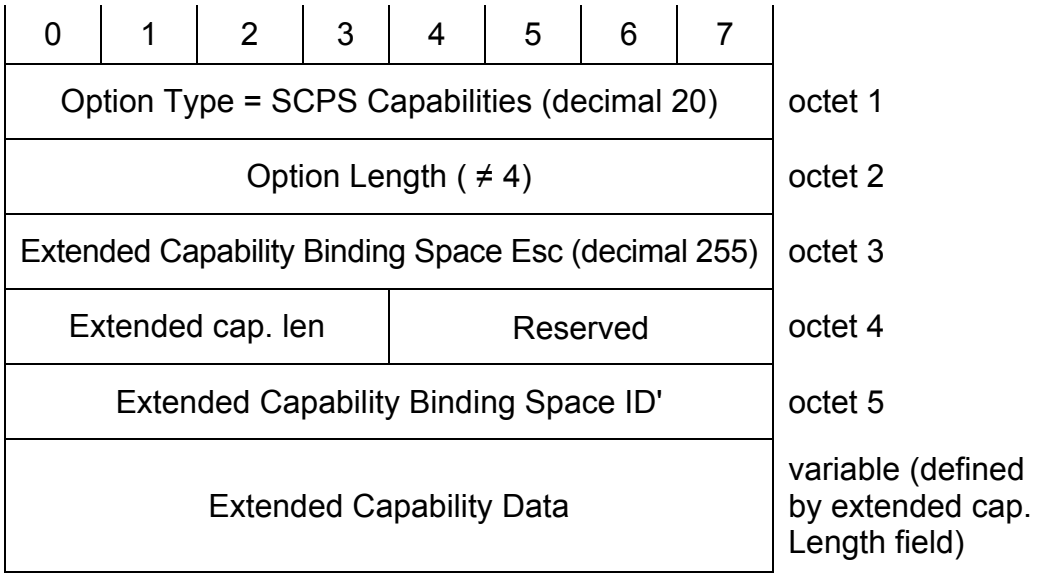

Figure 3-6: An Extended SCPS-TP Capability Specified by a Binding Space Identifier in the 256-511 Range

If extended capability binding space extensions are in use (i.e., octet 3 of figure 3-6 is 255) then the four bits following the extended capability length shall be set to 'zero' on transmission and must be ignored on receipt.

3.2.5.3 Meanings of Specific Extended Capability Binding Space Identifiers

3.2.5.3.1 Processing Unrecognized Extended Capability Binding Spaces

As with TCP options, implementations must ignore extended capability binding spaces that they do not understand. In these cases the implementations can read the extended capability length field and skip over the unknown data to continue processing the rest of the extended capabilities (if any).

NOTE – Extended capabilities that require negotiation (presumably most) should be handled similarly to other TCP options. That is, if the sender of the SYN signals that it wants to use extended capability X, presumably the sender of the SYN-ACK will accept/reject the use of that capability via an appropriate option in the SYN-ACK. As with other TCP options, failure to explicitly accept an extended option should be treated as a reject, so that the correct action is taken in the case that the receiver simply did not parse the extended capability.

3.2.5.3.2 Standard Extended Capabilities Binding Space Identifiers

Decimal values 0 through 15 for the extended capability binding space identifier are reserved for standards use.

3.2.5.3.3 Experimental Extended Capabilities Binding Space ID

Extended capability binding space identifier 254 decimal is reserved for experimentation. Implementations using this experimental extended capability binding space **must** take precautions to ensure that they are interpreting the extended capability data correctly.

3.2.5.4 Assignment of Extended Capability Binding Space Identifiers

CCSDS will assign extended capability binding space identifiers to interested parties. These parties may be individual vendors of SCPS-TP products, or communities of interest that jointly want to define a set of extended capabilities for a particular environment.

3.3 DATA TRANSFER

3.3.1 RECORD BOUNDARY OPTION

3.3.1.1 Record Boundary Option Format

The Record Boundary Option shall be located in the options area of the TCP header and shall contain the following fields:

	Length in octets
– Option Type (mandatory)	1
– Option Length (mandatory)	1

3.3.1.2 Record Boundary Option Fields

3.3.1.2.1 The Option Type field

a) is mandatory for the Record Boundary Option and shall occupy the first octet of the Option;

b) shall contain the decimal value 22.

3.3.1.2.2 The Option Length field

a) is mandatory for the SCPS Capabilities Option and shall occupy the second octet of the Option;

b) shall contain the decimal value 2.

3.3.1.3 Record Boundary Option Invocation

3.3.1.3.1 A sending application shall invoke the Record Boundary Option by providing an Application Program Interface (API) specific indication to the Transport service that the final octet of data in the associated service request represents the end of a record.

3.3.1.3.2 The end-of-record indication supplied by the sending application shall be associated with the last octet in the service data unit with which it was submitted.

3.3.1.3.3 The ending delimitation of the record shall be preserved during transmission (and possible retransmission) and across the remote service interface interfaces.

3.3.1.3.4 The sequence number of the octet associated with the record marker shall be the highest sequence number in the segment carrying the Record Boundary Option (i.e., it shall be the final octet in the segment), and a read shall associate the end of record with that octet.

NOTE – This modifies the effect of the Nagle algorithm and constrains the Transport protocol in performing repacketization during retransmission. Refer to RFC 896 (reference [8]) for a description of the Nagle algorithm.

3.3.1.3.5 The octet associated with the Record Boundary Option shall not be changed by the Transport Service Provider (TSP).

3.3.1.3.6 If any portion of a segment carrying the Record Boundary Option falls to the right of the receiver's window, the Record Boundary Option shall be discarded until the final octet of the segment with which the Record Boundary Option is associated is received.

3.3.1.3.7 Applications at both ends of a prospective connection must determine that a Transport service provides record boundary capability before connection establishment.

3.3.1.3.8 The Protocol Implementation Conformance Statement (PICS) shall state whether an implementation provides the Record Boundary Option.

NOTE – For a run-time method of determining whether the Record Boundary Option is available, refer to the Application Program Interface specification for the particular system on which the protocol is hosted.

3.3.1.3.9 If an application requests BETS and Record Boundaries, the application developer must ensure that the applications are robust against the possible loss of a record boundary.

NOTE – This may include incorporating a record length indication in the application record structure.

3.3.2 BEST EFFORT TRANSPORT SERVICE

3.3.2.1 Best Effort Transport Service Invocation

The Best Effort Transport Service shall be invoked using the SCPS Capabilities Option.

NOTE – Refer to 3.2.3 for a discussion of the SCPS Capabilities Option and its use.

3.3.2.2 R1 and R2 Thresholds

3.3.2.2.1 In Best Effort mode, the two thresholds, R1 and R2 (refer to RFC 1122, section 4.2.3.5, in reference [2], for a full description of these two thresholds), shall be interpreted as a count of transmissions (not time nor retransmissions).

3.3.2.2.2 The value of R1 shall be set by the sending TCP.

3.3.2.2.3 The value of R2 shall set by the sending application via a Transport-interface option.

3.3.2.2.4 The value of R2 shall be interpreted by the sending TCP as the threshold at which attempted retransmission of a segment is discontinued.

3.3.2.2.5 If the value of R2 is set to a non-zero positive number greater than one, then the value of R1 shall be set to a value less than that of R2.

3.3.2.2.6 When the number of transmissions of the same segment reaches or exceeds the value of R1, the sending TCP shall operate as it would in fully reliable mode.

NOTE – Exceeding the R1 threshold typically results in an attempt to identify a different route to the destination.

3.3.2.2.7 If the value of R1 is forced to zero because R2 is set to one,

a) no action shall be taken in response to the R1 threshold's being exceeded;

b) segments shall be discarded after being initially transmitted (rather than being queued for retransmission).

3.3.2.2.8 When either the R2 limit is reached (or exceeded) for a segment, or SND.NXT = SND.UNA + SND.WND, the sending TCP shall behave exactly as if it had received a positive acknowledgment that advances SND.UNA to SEG.SEQ of the next segment in the retransmission queue and that makes no change to SND.WND (refer to 3.9 of RFC 793 and 4.2.2.20 of RFC 1122, in references [4] and [2], respectively).

NOTES

1 SND.UNA is the sequence number of the first unacknowledged octet.

2 SND.NXT is the sequence number of the next octet to be sent.

3 SEG.SEQ is the sequence number of the first octet in a segment.

4 SND.WND is the number of octets of unacknowledged data that the sender is authorized by the receiver to have outstanding.

3.3.2.2.9 If R2 is set to zero by an application, it is an escape value that indicates that the TCP should not break a connection due to excessive retransmissions, nor should it invoke the

Best Effort Transport Service. Rather, it is a request to retransmit an 'infinite' number of times.

3.3.2.2.10 If R2 is set to zero, the value of R1 shall be determined by the implementer.

3.3.2.2.11 For SYN segments:

a) an R2-SYN shall be defined and shall be able to be set independently of R2 (in an implementation-dependent manner, a socket option with the Reference Implementation);

b) R2-SYN shall be greater than or equal to R2;

c) if R2 is increased to a value greater than R2-SYN, then R2-SYN shall be increased to match the value of R2 (independent of user input).

3.3.2.3 Missing ACK Segments

NOTE – The sending application may query the sending TCP in an API-specific manner to determine the location and length of segments that were not acknowledged by the receiver.

3.3.2.3.1 The sending TCP shall provide the sending application with a means to determine which data elements may not have been acknowledged by the receiver.

3.3.2.3.2 The sending TCP shall specify the missing data elements by reporting a pair of numbers for each missing data element:

a) the first number in the pair shall be the octet offset from the start of the connection to the first octet of unacknowledged data;

b) the second number in the pair shall be the octet offset from the start of the connection to the last octet of unacknowledged data.

3.3.2.3.3 Once read by the sending application, the missing-data-element specification pair shall be removed from the list.

3.3.2.3.4 System-dependent limits on the amount of information that can be retained by the sending TCP shall be documented for the implementation.

NOTE – The TCP implementation may restrict the amount of memory that is available for storing these pairs of numbers and, if so, may treat the available memory as a circular buffer. If an application does not request a report of missing data elements before the available memory fills, information could be lost.

3.3.2.3.5 For TCP implementations that support the Record Boundary Option, the implementation may specify missing data elements by reporting an additional pair of numbers for each missing data element:

a) the first number of the additional pair shall be the record offset from the start of the connection to the beginning of the unacknowledged data;

b) the second number of the additional pair shall be the record offset from the start of the connection to the end of the unacknowledged data.

3.3.2.4 Missing Data Segments

3.3.2.4.1 In Best Effort mode, the receiving TCP shall operate in the same manner as in fully reliable mode until an out-of-sequence segment (i.e., a segment having a higher-than-expected sequence number) arrives.

3.3.2.4.2 The receiving TCP shall store out-of-sequence segments in an out-of-sequence queue.

3.3.2.4.3 When an out-of-sequence queue is formed, the receiving TCP shall start an interval timer.

3.3.2.4.4 The receiving TCP shall use two thresholds:

a) a size-based threshold, BE1, defined as a value in octets that corresponds to a locally administered percentage of the receiver's buffer space;

b) a time-based threshold, BE2, defined as the interval in locally sized clock ticks after which out-of-sequence data will be delivered to the user.

3.3.2.4.5 The receiving TCP shall provide the receiving application with a means to set BE1 and BE2.

3.3.2.4.6 When the size of the out-of-sequence queue reaches or exceeds BE1 or the interval BE2 elapses, the receiving TCP shall

a) issue an acknowledgment that acknowledges the data in the missing segment plus any data in the out-of-sequence queue that would be acknowledgeable were the missing segment received;

b) issue an error or warning or advisory to the receiving application identifying the size of the missing segment(s) in octets;

c) deliver the subsequent in-sequence data to the receiving application.

NOTE – If missing and acknowledged segments arrive after RCV.NXT has advanced, the receiving TCP may discard them. Alternatively, the receiving TCP may store them via some out-of-band storage means for off-line merging with the rest of the data. However, this requires the receiving TCP to maintain the sequence number and size of each area of missing data. A Best Effort receiver that will be receiving large volumes of data will probably exist in a (ground-based) system with substantial receive buffer capacity. Therefore, it is anticipated that the threshold BE1 will be able to be set sufficiently high that most retransmissions will be cleared from the network by the time BE1 is met or exceeded.

3.3.2.4.7 The BE2 timer shall be used as follows:

a) the receiving TCP shall start the BE2 timer when all in-sequence data have been read by the application and out of sequence data exist in the receive queue;

b) when the hole at SND.UNA is closed (either by receipt of the missing data, exceeding the BE1 threshold, or expiration of the BE2 timer), the receiving TCP shall cancel the BE2 timer;

c) if additional holes in the out of sequence queue exist, the receiving TCP shall restart the BE2 timer as described above.

3.4 ERROR RECOVERY

3.4.1 TRANSMISSION TIMEOUT

3.4.1.1 The retransmission timeout should be dynamically determined.

3.4.1.2 If the dynamic computation of the retransmission timeout cannot be accomplished, it **may** be statically configured via the MIB, which supports static selection of the retransmission timeout mechanism as defined in 5.3.1 of this document.

3.4.1.3 When retransmission timeouts are dynamically determined, the following requirements are in force:

a) the Jacobson algorithm for computing smoothed Round-Trip Time (RTT) shall be used to estimate the RTT and its variance (references [7], [2], and [4]);

b) the Karn algorithm shall be used to select those RTT measurements that are valid except when the TCP Timestamps Option is present (references [6], [2], and [4]);

c) when the TCP Timestamps Option is present, and the round trip of a retransmitted segment shall be factored into the RTT estimate.

3.4.2 CONGESTION CONTROL AND CORRUPTION

3.4.2.1 When the implementation has elected not to provide congestion control, the following algorithms are not available:

a) Van Jacobson's Slow Start algorithm;

b) Van Jacobson's Congestion Avoidance algorithm;

c) the exponential back-off of the retransmission timer for successive retransmissions.

3.4.2.2 Implementations that provide congestion control may do so in one of two ways:

a) by using the standard means within TCP (i.e., Van Jacobson's Slow Start Algorithm, Van Jacobson's Congestion avoidance algorithm, and exponential back-off); or

b) by using the TCP Vegas congestion-control mechanisms documented in reference [12].

3.4.2.3 When congestion control is implemented, the Jacobson algorithms for slow start and congestion avoidance or corresponding TCP Vegas algorithms shall be used when the default source of loss is congestion and there is no evidence that data loss is due to a cause other than congestion.

NOTE – The use of the Jacobson and TCP Vegas algorithms when there is evidence of corruption is optional, but may result in reduced performance.

3.4.2.4 When congestion control is implemented but there is evidence that data loss is due to corruption and not network congestion, the 'exponential backoff' algorithm may optionally not be invoked.

NOTE – Evidence of corruption rather than congestion may be obtained from inter-layer signaling, a MIB-query, or other mechanism.

3.4.2.5 When explicit congestion notification (RFC 3168) is implemented, an explicit congestion notification via the receipt of a segment with the ECN-Echo (ECE) bit set must be treated as an indication of network congestion. In this case, receipt of a segment with the ECE bit set must invoke the same congestion response as a lost segment.

3.4.2.6 When a connection receives an indication that corruption has been experienced:

a) the sending TCP shall send the Corruption Experienced Option (Type = 23, Length = 2) to the receiving TCP at an implementation-defined rate not to exceed once per round-trip time;

b) the sending TCP shall continuing sending the Corruption Experienced Option for no more than two round-trip times after the previous indication of corruption was received;

c) upon receipt of the Corruption Experienced Option from a sending TCP, the receiving TCP shall not send a corresponding Corruption Experienced Option to its peer.

3.4.2.7 For retransmissions of data in which there is no indication that corruption is being experienced, the exponential backoff algorithm shall be used to compute successive Retransmission Timeout values for the same segment.

3.4.2.8 The following actions support the operation of a TCP that receives a Corruption Experienced Option:

3.4.2.8.1 At the initialization of a connection, four variables shall be created:

- corrupted_path_time, initialized to the time when the connection is opened;
- cwnd_commit_time, initialized to the time when the connection is opened;
- prev_ssthresh, initialized to the initial value of ssthresh, typically 65536;
- prev_cwnd, initialized to 1 Maximum Segment Size (MSS).

3.4.2.8.2 Upon indication of a requirement for retransmission (e.g., duplicate ACKs, SNACK, retransmission timeout):

a) if corrupted_path_time is less than the current time (i.e., the path is not corrupted):

   1) cwnd_commit_time shall be set to the current time plus the current retransmission timeout value;

   2) prev_ssthresh shall be set to the value of ssthresh;

   3) prev_cwnd shall be set to the value of cwnd;

   4) if the retransmission is the result of duplicate acknowledgments (i.e., fast retransmit), the value of SendMSS * the duplicate acknowledgment threshold shall be added to prev_cwnd, where SendMSS is the MSS value received from the remote host;

   5) the current values of cwnd and ssthresh shall be modified according to the rules of the congestion control algorithm in use;

b) if corrupted_path_time is greater than or equal to the current time (i.e., the path has been marked as corrupted), cwnd and ssthresh shall not be modified.

3.4.2.8.3 When a TCP receives a Corruption Experienced Option from a remote TCP:

a) if corrupted_path_time is less than the current time and if cwnd_commit_time is greater than the current time:

   1) if ssthresh is less than prev_ssthresh, ssthresh shall be set to the value of prev_ssthresh;

   2) if cwnd is less than prev_cwnd, cwnd shall be set to the value of prev_cwnd;

   3) cwnd_commit_time shall be set to the current time;

b) corrupted_path_time shall be set to the current time + 2 * the smoothed RTT.

3.4.2.8.4 If the TCP receives a Source Quench indication:

a) cwnd_commit_time shall be set to the current time;

b) cwnd shall be set to SendMSS;

c) corrupted_path_time shall be set to the current time.

3.4.3 LINK OUTAGES

When the sending TCP receives an indication of a link outage, either from inter-layer signaling or from MIB-query, the sending TCP

a) shall suspend transmission;

b) shall suspend the retransmission timer;

c) may continue to queue user data for transmission, to the extent that retransmission buffer space permits (this is a local implementation option);

d) shall periodically, at a locally determined rate (recommended to be some small multiple of the previous retransmission timeout), send a link-probe segment, consisting of the first octet of unacknowledged data:

   1) if no unacknowledged data exist, no link probe shall be sent;

   2) if data are submitted for transmission during a link outage, the sending of link probes shall resume;

e) shall not interpret link probe segments as retransmissions for the purposes of modification of the retransmission timeout or for invocation of slow start or congestion avoidance;

f) may resume normal communication when the sending TCP

   1) receives a 'link restored' or 'link redirect' control message (refer to reference [10]);

   2) determines that the link is available from a MIB-query (refer to 5.2.2.8 of this document);

   3) receives an acknowledgment that arrives at least one RTT after the commencement of the link outage.

3.5 SELECTIVE NEGATIVE ACKNOWLEDGMENT OPTION

3.5.1 SELECTIVE NEGATIVE ACKNOWLEDGMENT OPTION FORMAT

The Selective Negative Acknowledgment (SNACK) Option shall be located in the options area of the TCP header and shall contain the following fields:

	Length in octets
– Option Type (mandatory)	1
– Option Length (mandatory)	1
– Hole1 Offset (mandatory)	2
– Hole1 Size (mandatory)	2
– SNACK Bit-Vector (optional)	variable

3.5.2 SELECTIVE NEGATIVE ACKNOWLEDGMENT OPTION FIELDS

3.5.2.1 Option Type

3.5.2.1.1 The Option Type field is mandatory and shall occupy the first octet of the Option.

3.5.2.1.2 The Option Type field shall contain the decimal value 21.

3.5.2.2 Option Length

3.5.2.2.1 The Option Length field is mandatory and shall occupy the second octet of the Option.

3.5.2.2.2 The Option Length field shall contain the total number of octets used by the Option:

a) if the SNACK Bit-Vector is not included, the length of the Option shall be fixed at six octets;

b) if the SNACK Bit-Vector is included, the length of the Option shall be six octets plus the number of octets in the SNACK Bit-Vector.

NOTE – The length of the SNACK Bit-Vector is an implementation issue and may vary from one SNACK Option to another.

3.5.2.3 Hole1 Offset

3.5.2.3.1 The Hole1 Offset field is mandatory and shall occupy the third and fourth octets of the Option.

3.5.2.3.2 The Hole1 Offset field shall contain the offset from the current acknowledgment number of the first hole being reported in this Option:

a) the offset value is specified in terms of maximum-segment-sized units, calculated by subtracting the acknowledgment number from the offset sequence number and dividing that difference (using integer arithmetic) by the amount of user data carried in a maximum-sized segment;

b) if the division produces a remainder, it is added to the size of the hole.

3.5.2.4 Hole1 Size

3.5.2.4.1 The Hole1 Size field is mandatory and shall occupy the fifth and sixth octets of the Option.

3.5.2.4.2 The Hole1 Size field shall contain the size of the first hole being reported in this Option:

a) the Hole1 Size field is specified in terms of maximum-segment-sized units, calculated by dividing (using integer arithmetic) the size of the hole in octets by the amount of user data carried in a maximum-sized segment;

b) if the division produces a non-zero remainder, the Hole1 Size is rounded up.

3.5.2.5 SNACK Bit-Vector

3.5.2.5.1 The SNACK Bit-Vector field is optional and shall, if included, occupy contiguous octets immediately following the Hole1 Size field.

3.5.2.5.2 The SNACK Bit-Vector field shall contain information about missing data in the receiver's buffer subsequent to Hole1:

a) the SNACK Bit-Vector field shall map the sequence space of the receiver's buffer into MSS-sized blocks, beginning one octet beyond the end of the block specified by Hole1;

b) each '0' in the SNACK Bit-Vector field shall signify missing data in the corresponding MSS-sized block in the receiver's resequencing queue;

c) zeroes shall be added to the right of the last '1' if necessary to assure that the SNACK Bit-Vector ends on an octet boundary;

d) zeroes to the right of the last '1' shall not be interpreted as missing MSS-sized blocks.

3.5.3 SNACK OPTION OPERATION

3.5.3.1 The SCPS SNACK Option shall be enabled when both TCPs include the SNACK capability in the SCPS Capabilities Option in their TCP SYN segment headers.

3.5.3.2 The receiving TCP may invoke the SNACK Option by sending an appropriately formed SNACK Option on an ACK segment whenever an out-of-sequence queue forms or a new hole in the out-of-sequence queue forms.

NOTES

1 There are mission-specific considerations regarding whether transmission of the SNACK Option should be delayed, and by how much, in anticipation of the out-of-order arrival of missing segment(s). This decision is implementation specific and should take into account the probability of segment misordering by the underlying network(s). Unless segment misordering is highly unlikely, delaying transmission of the SNACK Option may be beneficial. (The SNACK Option forces immediate retransmission; as such, the SNACK sender should be relatively certain that the retransmission is necessary.)

2 Implementations should ensure that the receiving TCP allows sufficient time for the SNACK Option to reach the sender and for the sender to retransmit the data before sending a subsequent SNACK specifying the same hole. In environments prone to significant loss, the SNACK Option or resulting retransmissions may be lost. In these environments, it may be desirable to send a SNACK Option for a hole that has been reported in a previous SNACK Option. This is legal, but if sent too soon will result in unnecessary retransmissions. Note that it may be advisable to send SNACK Options for existing holes when a FIN is received. At this point, no other (in-sequence) data is forthcoming from the remote TCP. Outstanding holes that are not retransmitted as a result of duplicate acknowledgments or SNACK Options that are in flight will result in retransmission timeouts unless SNACK Options are sent to stimulate prompt retransmission. Whether and when to send a SNACK for holes that have been previously reported by a SNACK Option are details that are implementation specific.

3.5.3.3 Upon receipt of a SNACK Option, the data sender shall retransmit all segments necessary to fill the signaled holes.

NOTE – This relatively aggressive retransmission scheme is appropriate when the goal is to prevent retransmission time-outs, which cost more in terms of link idle time than unnecessary retransmissions cost in terms of wasted bandwidth. The data sender may process the SNACK Option by first retransmitting the segments corresponding to the Hole1 specification. If a SNACK Bit-Vector field is present, the data sender may left-shift the SNACK Bit-Vector until the last '1' is shifted out, retransmitting the segment corresponding to each '0' encountered in the process. This is an example for illustrative purposes; the exact mechanism is implementation specific.

3.5.3.4 It is strongly recommended that retransmissions occur in the order of ascending sequence numbers.

NOTES

1 Figure 3-7 shows a hypothetical out-of-sequence queue formed at the data-receiver. RCV.NXT refers to the next expected sequence number (that is, all data before this sequence number has been correctly received). The ruled lines along the arrow represent MSS boundaries, which are 1024-octet blocks in this example. There are three holes in the out-of-sequence queue: a three-segment hole at RCV.NXT, followed by four correctly received segments; a one-segment hole, followed by two correctly received segments; and then a two-segment hole, followed by one correctly received segment.

2 Figure 3-8 illustrates the SNACK Option applied to the out-of sequence queue depicted in figure 3-7. Each row of figure 3-8 is four octets long. In this instance, complete information could be specified with a 16-bit vector. The choices of SNACK Bit-Vector field length are strictly implementation dependent, within the constraints of TCP's maximum header length.

3 Figure 3-9 illustrates the use of the SNACK Option without the SNACK Bit-Vector. In this case, three separate SNACK Options are used to fully report the state of the out-of-sequence queue. These SNACK Options may appear on the same acknowledgment or may appear on separate acknowledgments.

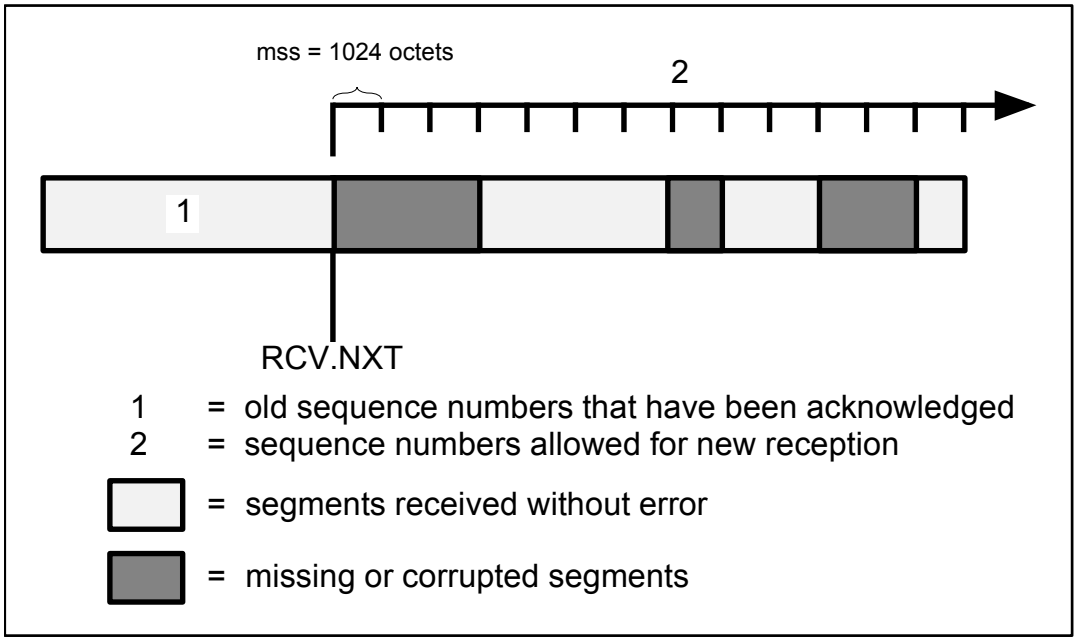

Figure 3-7: Out-of-Sequence Queue for SNACK Example

1	8	16	24	32
Type=21	Length=8	Hole1 Offset = 0		
Hole1 Size = 3		11110110	01000000	

Figure 3-8: SNACK Option Resulting from Out-of-Sequence Queue Example

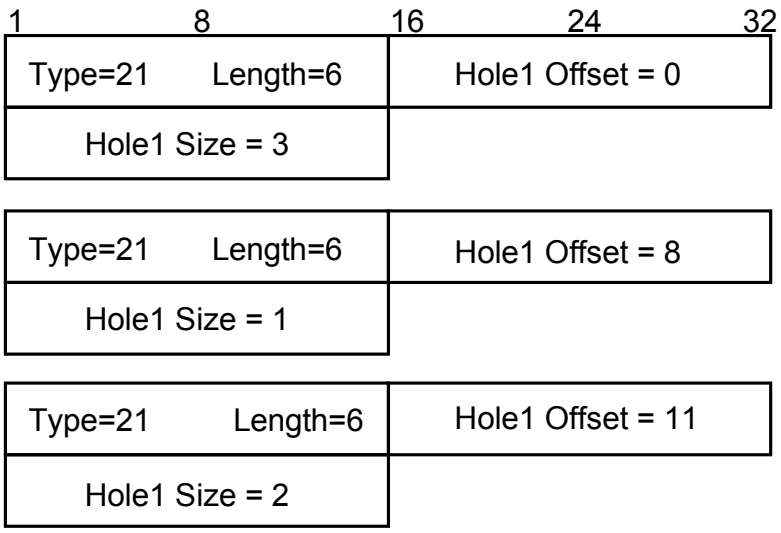

Figure 3-9: SNACK Options (without SNACK Bit-Vector) Resulting from Out-of-Sequence Queue Example

3.6 SCPS-TP HEADER COMPRESSION

3.6.1 SCPS-TP COMPRESSED HEADER FORMAT

The SCPS-TP compressed header shall contain some or all of the following fields:

NOTE – Fields designated 'mandatory' are required for all compressed headers; the presence of other fields depends on the contents of the uncompressed TCP header.

	Length in octets
– Connection Identifier (mandatory)	1
– Compressed Header Bit-Vector (mandatory)	1-2
– Urgent Pointer	2
– Window	2
– ACK Number	4
– Sequence Number	4
– Short-form SNACK	4
– Outbound Timestamp	format dependent
– Echo Reply Timestamp	format dependent
– Options Length	1
– Options	data dependent
– Pad Field	1
– Checksum (mandatory)	2

3.6.2 SCPS-TP COMPRESSED HEADER FIELDS

3.6.2.1 Connection Identifier

3.6.2.1.1 The Connection Identifier field is mandatory for all SCPS-TP compressed headers and shall occupy the first octet of the compressed header.

3.6.2.1.2 The Connection Identifier field shall contain the Connection Identifier supplied to the peer during the SYN-segment exchange of the SCPS Capabilities Option.

3.6.2.2 Compressed Header Bit-Vector

3.6.2.2.1 The Compressed Header Bit-Vector field is mandatory for all SCPS-TP compressed headers and shall occupy at least one and no more than two octets beginning with the first octet following the Connection Identifier field.

3.6.2.2.2 The Compressed Header Bit-Vector field shall contain information necessary for decompressing the compressed header, as detailed in table 3-2.

Table 3-2: Compressed Header Bit-Vector Contents

Bit Name	Meaning when set to '1'	Notes
More	Compressed Header Bit-Vector is 16 bits long rather than 8 bits long.	
TS1	TCP Timestamp Option is present.	See 6.2.2.5
TS2	A timestamp reply (TS Echo Reply) appears in the compressed header.	See 6.2.2.5
RB	The last octet of data accompanying this segment is the end of a user-defined record.	See 3.3.1
Snack	The compressed header contains a short-form SNACK option.	See 3.5
P	The Push bit from the uncompressed TCP header is set to '1'.	
S	The compressed header contains a 4-octet sequence number.	
A	The compressed header contains a 2-octet window specification and a 4-octet acknowledgment number.	
Opts	The compressed header contains uncompressed options.	
Pad	The compressed header contains one octet of padding.	
URG	The URG bit from the uncompressed TCP header is set to '1'.	
ECE	The ECN-Echo flag from the uncompressed TCP header is set to '1'.	RFC 3168
AckR	The ACK bit from the uncompressed TCP header is set to '1' (this field is valid only when the RST bit is set to '1').	
RST	The RST bit from the uncompressed TCP header is set to '1'.	
CWR	The ECN congestion window reduced flag from the uncompressed TCP header is set to '1'.	RFC 3168
FIN	The FIN bit from the uncompressed TCP header is set to '1'.	

3.6.2.3 Urgent Pointer

3.6.2.3.1 The Urgent Pointer field shall be included if the URG flag is set to '1' in the TCP header and shall occupy two octets immediately following the Compressed Header Bit-Vector field.

3.6.2.3.2 The Urgent Pointer field shall contain the unmodified urgent pointer value from the TCP header.

3.6.2.4 Window

3.6.2.4.1 The Window field shall be included if either the window value of the acknowledgment number has changed from the last segment sent on the connection, or if nothing has changed from the last segment sent on the connection, and shall occupy two octets immediately following the location for the Urgent Pointer field.

3.6.2.4.2 The Window field shall contain the unmodified window value from the TCP header.

3.6.2.5 ACK Number

3.6.2.5.1 The ACK Number field shall be included if either the window value of the acknowledgment number has changed from the last segment sent on the connection, or if nothing has changed from the last segment sent on the connection, and shall occupy four octets immediately following the Window field.

3.6.2.5.2 The ACK Number field shall contain the unmodified ACK number from the TCP header.

3.6.2.6 Sequence Number

3.6.2.6.1 The Sequence Number field shall be included if the segment is retransmittable (i.e., user data is included or the FIN flag is set to '1'), and shall occupy four octets immediately following the location for the ACK Number field.

3.6.2.6.2 The Sequence Number field shall contain the unmodified sequence number from the TCP header.

3.6.2.7 Compressed Short-form SNACK Option

3.6.2.7.1 The compressed short-form SNACK bit shall be set whenever a compressed short-form (i.e., **not** including a SNACK bit vector) SNACK option is present. The compressed short-form SNACK option shall occupy the four octets immediately following the sequence number field.

3.6.2.7.2 The first two octets of the compressed short-form SNACK option shall contain the hole1 offset (see 3.5.2.3). The next two octets shall contain the hole1 length (see 3.5.2.4).

NOTE – Other SNACK options, including other short-form and all long-form options (those with uncompressed option lengths greater than six octets) must be carried in the **uncompressed** TCP Options portion of the compressed header.

3.6.2.8 Outbound Timestamp

3.6.2.8.1 The Outbound Timestamp (TS1) field shall be included if

- a TCP Timestamps Option is present; **and**
- the SCPS Capabilities Option negotiation indicated that NL Ts were *not* available;

and shall occupy one or more octets immediately following the location for the Sequence Number field.

3.6.2.8.2 The TS1 field shall contain the timestamp value to be echoed.

3.6.2.9 Echo Reply Timestamp

3.6.2.9.1 The Echo Reply Timestamp (TS2) field shall be included if a TCP Timestamps Option is present and shall occupy one or more octets immediately following the location for the TS1 field.

3.6.2.9.2 The TS2 field shall contain the echo reply timestamp value.

NOTE – Because separate 'TS1' and 'TS2' bits are used in the Compressed Header Bit-Vector, the equivalent of RFC 1072 (reference [B4]) timestamps may be compressed, if desired.

3.6.2.10 TCP Options Length

3.6.2.10.1 The TCP Options Length field shall be included if any TCP options remain after header compression and shall occupy one octet immediately following the location for the TS2 field.

3.6.2.10.2 The TCP Options Length field shall contain the length in octets of the remaining TCP options.

NOTE – The Record Boundary flag, the TS1 and TS2 flags, and the SNACK flag in the Compressed Header Bit-Vector field exist for the purpose of compressing TCP options.

3.6.2.11 TCP Options

3.6.2.11.1 The TCP Options field shall be included if any TCP options remain after header compression and shall occupy one or more octets immediately following the TCP Options Length field.

3.6.2.11.2 The TCP Options field shall contain any TCP options that have not been compressed.

3.6.2.12 Pad

3.6.2.12.1 The Pad field may be included to ensure the compressed header ends on an even octet boundary and shall occupy one octet immediately following the location for the TCP Options field.

3.6.2.12.2 If included, the Pad field shall have a value of zero.

3.6.2.12.3 A Pad field shall not be included if its inclusion necessitates adding a second octet to a single-octet Compressed Header Bit-Vector field.

NOTE – A zero-value second octet of the Compressed Header Bit-Vector field may also be used for padding a compressed header to an even octet boundary.

3.6.2.13 Checksum

3.6.2.13.1 The Checksum field is mandatory for all SCPS-TP compressed headers and shall occupy the two final octets of the compressed header.

3.6.2.13.2 The Checksum field shall contain the value obtained using the standard TCP checksum algorithm for the contents of the compressed header, the user data, and the TCP pseudo-header. The decimal value 105 (compressed SCPS-TP) shall be used as the N-user Internet Protocol Number in the pseudo header when calculating the checksum when header compression is in use.

NOTE – The SCPS-TP compressed segment format is illustrated in figure 3-10. The fields shown with dashed outlines (after the first octet of the Compressed Header Bit-Vector but before the checksum) are only included when necessary. Their presence or absence is indicated by the corresponding bits in the Compressed Header Bit-Vector. In the figure, each optional field shows three elements of information: the bit of the Compressed Header Bit-Vector that signals its presence, the name of the field, and the length of the field in octets. These are shown in the format 'Bit: Name (Length)'.

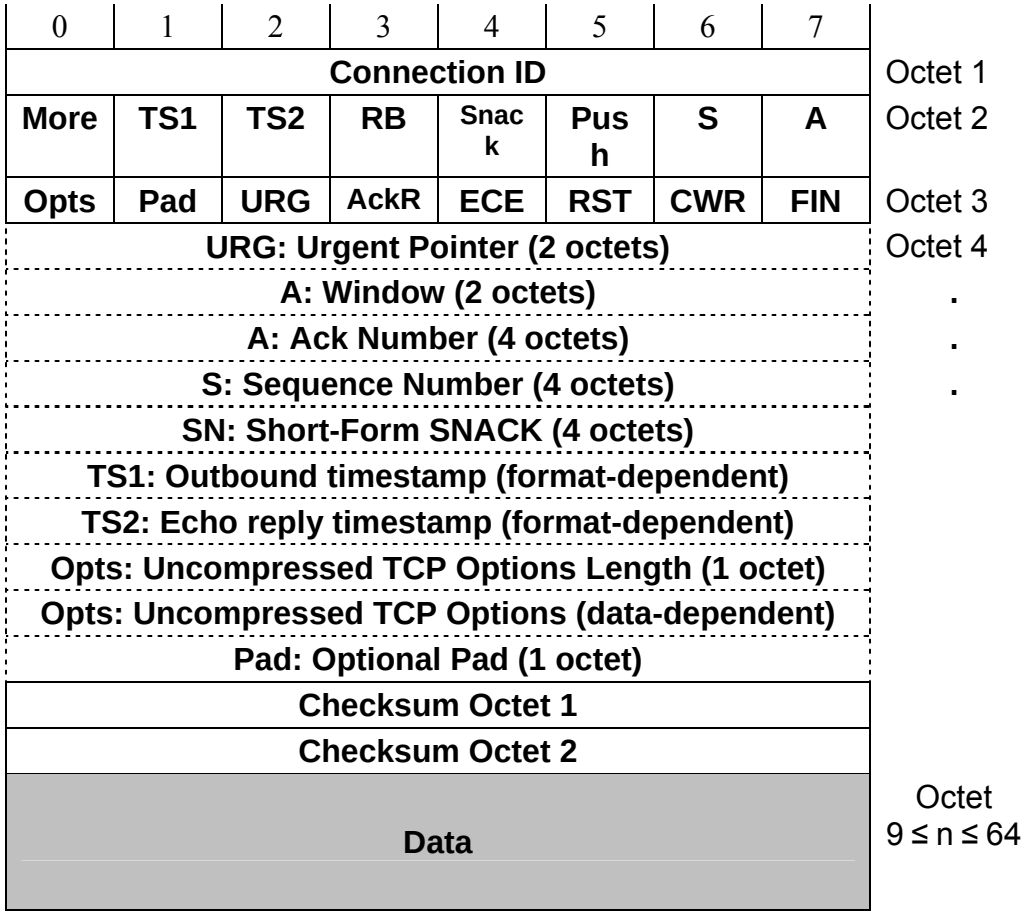

Figure 3-10: Compressed SCPS-TP Header

3.6.3 SCPS-TP COMPRESSED HEADER INVOCATION

3.6.3.1 The TCP initiating a connection shall request SCPS-TP Header Compression by including the SCPS Capabilities Option with Com bit set to '1' in its uncompressed SYN segment.

3.6.3.2 The responding TCP endpoint shall either

a) accept the offer to perform SCPS Header Compression by including its own SCPS Capabilities Option with Com bit set to '1' in its SYN ACK segment;

b) reject the offer to perform SCPS Header Compression.

NOTES

1 Refer to 3.2.3 for a detailed discussion of the SCPS Capabilities Option.

2 The SCPS Header Compression algorithm supports 'piggy-backing' of acknowledgments on data-carrying segments (as in uncompressed TCP), but *it is an implementation option* whether to exercise this ability. The compressor for a particular implementation may send acknowledgments (and other information not directly related to the data being transferred) separately from the data in order to ensure a constant header size for data-carrying segments. (This can aid in packing fixed-length lower-layer frames when bulk data is to be transferred.) In accordance with the robustness principle stated in RFC 1122 (reference [2]) and in TCP, a decompressor must be prepared to accept piggy-backed acknowledgments even if the compressor in that implementation does not generate them. (The robustness principle roughly states 'be generous in what you accept and conservative in what you send', and is intended to promote interoperability.)

3 When the SCPS-NP is in use, Transport Protocol Identifier (TP-ID) 6 identifies uncompressed TCP and TP-ID 5 identifies compressed segments. Alternative methods for differentiating these packet types exist if the SCPS-NP is not being used; for example, two CCSDS Path IDs may be allocated to differentiate between compressed and uncompressed segments. Refer to the SCPS-NP specification (reference [10]) for a full description of SCPS-NP Transport Protocol Identifiers.

3.6.4 DECOMPRESSOR PROCESSING

Upon receipt of a segment of type 'Compressed TCP' from the Network Layer, along with relevant Network-layer information, such as the length of the packet, incoming source timestamp, and the source and destination addresses:

a) the decompressor shall use the Connection Identifier and network addresses to find the TCP endpoint with which this packet should be associated;

b) if the endpoint is not located, the decompressor shall discard the segment and log an error;

c) if the endpoint is located, the decompressor shall use the information from the TCP pseudo header to verify the checksum in the compressed segment;

d) if the checksum fails, the decompressor shall discard the segment and log an error;

e) the decompressor shall reconstruct the TCP header using a template created from the uncompressed header of the previous segment received on the connection with all fields except the port information initialized to zero;

f) the decompressor shall save the reconstructed header for use in decompressing subsequent compressed packets;

g) the decompressor shall output the decompressed segment for immediate processing by TCP or insertion into an out-of-sequence queue.

3.7 MULTIPLE TRANSMISSIONS FOR FORWARD ERROR CORRECTION

3.7.1 INVOCATION

3.7.1.1 The transport service shall provide a connection-specific means, referred to as the Multiple Forward Transmissions (MFX) parameter, to specify the number of times that acknowledged segments shall be repeated when queued for transmission or retransmission by TCP.

3.7.1.2 The Transport Service User (TSU) invokes the MFX capability for a connection by changing the 'MFX' parameter for that connection from its default value of one to a value greater than one.

3.7.1.3 The MFX capability **should not** be used on a connection that also employs closed-loop congestion control (such as Van Jacobson's congestion control scheme or the TCP Vegas scheme).

3.7.2 SENDING TCP OPERATION

3.7.2.1 The sending TCP shall send all segments that require acknowledgment (that is, those that carry user data, or the SYN flag, or the FIN flag) the number of times indicated by the MFX parameter.

3.7.2.2 Segments that do not carry data, or the SYN flag, or the FIN flag, shall not be transmitted multiple times.

3.7.2.3 Multiple forward transmissions shall not count as retransmissions with respect to the R1 and R2 thresholds (refer to section 3.3.2 of this document).

3.7.2.4 If it is necessary to retransmit a segment (as a result of retransmission timeout, or SNACK, or fast retransmit), the segment shall be transmitted the number of times indicated by the MFX parameter.

3.7.2.5 The multiple transmission of a retransmitted segment shall be counted as a single retransmission for the purposes of R1 and R2 accounting.

NOTE – Successive segment transmissions by a TCP entity may be interleaved with transmissions of subsequent segments. Such interleaving is an implementation issue and is not specified here.

3.7.3 RECEIVING TCP OPERATION

The receiving TCP shall use normal TCP protocol mechanisms to detect and discard duplicate segments received as a result of the multiple transmissions.

4 USER DATAGRAM PROTOCOL EXTENSION

4.1 SCPS-TP adopts the User Datagram Protocol (UDP) as specified in Internet Standard 6 (reference [3]) and its supporting RFCs, with the modification specified in section 4 of this document.

4.2 UDP shall use the precedence management capabilities of the Network-layer protocol to ensure that datagrams are delivered in accordance with their precedence.

NOTE – UDP maintains no internal queues, and therefore consumes data in FIFO order from both the TSU and the Network service provider. UDP depends upon the Network service to deliver datagrams to UDP in accordance with their assigned precedence levels for subsequent delivery to the TSU.

5 MANAGEMENT INFORMATION BASE (MIB) REQUIREMENTS

5.1 TYPES OF MANAGEMENT INFORMATION

Three types of management information shall be maintained:

- information that pertains to a route;
- information that pertains to the TCP portion of TP; and
- information that pertains to the UDP portion of TP.

5.2 MIB REQUIREMENTS FOR ROUTE-SPECIFIC INFORMATION

5.2.1 GENERAL REQUIREMENTS

5.2.1.1 The MIB shall contain information on a per-route basis.

NOTE – To the SCPS-TP, a 'route' is a data structure that contains information regarding the characteristics of the path between the local host and the destination host.

5.2.1.2 The route-specific MIB information shall be made available to TP in an implementation-specific manner.

5.2.1.3 For each route supported, the information indicated in 5.2.2 is required.

5.2.1.4 When the SCPS-TP is operating over SCPS-NP (see reference [10]), the SCPS-TP MIB parameters in 5.2.2 are provided by the SCPS-NP MIB parameters as follows:

SCPS-TP MIB Parameter	SCPS-TP Subsection	SCPS-NP MIB Parameter	SCPS-NP Subsection
routeMTU	5.2.2.1	npRouteMTUout	4.3.16
routeSendSpeed	5.2.2.2	npRouteSendBPS	4.3.18
routeReceivePipe	5.2.2.3	npRouteRcvPipe	4.3.19
routeSendPipe	5.2.2.4	npRouteSendPipe	4.3.20
routeSSThresh	5.2.2.5	npRouteSSThresh	4.3.21
routeRTT	5.2.2.6	npRouteRTT	4.3.22
routeRTTVar	5.2.2.7	npRouteRTTVar	4.3.23
routeAvail	5.2.2.8	npRouteAvail	4.3.24

5.2.2 REQUIRED ROUTE-SPECIFIC INFORMATION

5.2.2.1 Route Maximum Transmission Unit

Route Maximum Transmission Unit is the maximum sized message supported by the respective route. This information is used in TCP segmentation and UDP maximum datagram size enforcement. It is a local matter how this field is established (for many implementations, it will be set manually and updated very infrequently).

Name: routeMTU
Syntax: Integer
Access by TP: Read-only

5.2.2.2 Route Send Speed

Route Send Speed is a measure of the estimated data rate in bits per second available on the respective route. This information may be used by the TP (either TCP or UDP or both) to apply a locally specified form of rate-based flow control.

Name: routeSendSpeed
Syntax: Gauge
Access by TP: Read-only

NOTE – Rate-based flow control is a means of limiting the transmission rate of outbound data either on the basis of a fixed rate parameter, as specified here, or on the basis of feedback from remote systems.

5.2.2.3 Route Receive Pipe

Route Receive Pipe is an estimate of the inbound bandwidth-delay product of the respective route. This information is used by the TP to allocate memory to the receive buffers for this endpoint. This statistic may be updated to reflect a revised RTT by TCP upon close of a connection. (The route receive pipe is read by TP when the route is initially established and remains static for the duration of the communication session.)

Name: routeReceivePipe
Syntax: Gauge
Access by TP: Read/Write

5.2.2.4 Route Send Pipe

Route Send Pipe is an estimate of the outbound bandwidth-delay product of the respective route. This information is used by the TP to allocate memory to the send buffers for this endpoint. This statistic may be updated to reflect a revised RTT by TCP upon close of a connection.

Name: routeSendPipe
Syntax: Gauge
Access by TP: Read/Write

5.2.2.5 Route Slow Start Threshold

Route Slow Start Threshold is the initial value, in octets, at which a SCPS-TP using the respective route should convert from slow start operation to congestion avoidance operation. This statistic may be updated by SCPS-TP upon close of a connection.

Name: routeSSThresh
Syntax: Integer
Access by TP: Read/Write

5.2.2.6 Route Round Trip Time

Route Round Trip Time is the RTT, in milliseconds, expected on the respective route. This statistic may be updated by SCPS-TP upon close of a connection.

Name: routeRTT
Syntax: Integer
Access by TP: Read/Write

5.2.2.7 Route Round Trip Time Variance

Route Round Trip Time Variance is an estimate of the mean deviation, in milliseconds, expected for the respective route. This statistic may be updated by SCPS-TP upon close of a connection.

Name: routeRTTVar
Syntax: Integer
Access by TP: Read/Write

5.2.2.8 Route Availability

Route Availability is an indication of the current operational status of the respective route. If a TCP is experiencing retransmission, it may examine this variable to attempt to determine the reason. The testing (3) state indicates that no operational packets can be passed. The out (4) state indicates that a link-outage has been reported for some link associated with this route and is currently in effect.

Name: routeAvail
Syntax: Integer {
up (1),
down (2),
testing (3),
out (4)
}
Access by TP: Read only

5.2.2.9 Route Corruption

Route Corruption is an indication that some link associated with the respective route has reported corruption within an administratively controlled time interval. The SCPS-TP uses this indication to enable its corruption response in the event of a requirement for retransmissions. The route corruption indication maintains a timestamp (of locally specified format) to indicate the local time at which the route corruption state last changed. It also maintains information about how to age-out the route corruption indication once it is set. This aging information may be a count of RTTs or a simple number of seconds. If the route is marked as corrupted, the route will remain marked as corrupted for the period of time specified by the aging information.

```
Name:  routeCorrupted
Syntax:  Sequence {
        rtCorrState {Integer
                Uncorrupted (0),
                Corruption experienced (1)
        }
        rtCorrTransition { timestamp
        }
        rtCorrAge Sequence {
                units Integer {
                        roundTripTimes (0),
                        elapsedTimeSeconds (1)
                }
                timeToAgeOut Integer
        }
}
Access by TP:  Read/Write
```

5.2.2.10 Route Congestion

Route Congestion is an indication that some link associated with the respective route has reported congestion. The SCPS-TP uses this indication to enable its congestion response in the event of a requirement for retransmissions. The route congestion indication maintains a timestamp (of locally specified format) to indicate the local time at which the route congestion state last changed. SCPS-TP resets this indication in a manner that is specific to the particular congestion control algorithm in use.

```
Name:  routeCongested
Syntax:  Sequence {
        rtCongState { Boolean
        }
        rtCongTransition { timestamp
        }
}
Access by TP:  Read/Write
```

5.3 MIB REQUIREMENTS FOR SCPS TRANSMISSION CONTROL PROTOCOL

5.3.1 MIB-II REQUIREMENTS

The TCP implementation shall maintain the information specified in the TCP Group of the MIB-II definition (Internet Standard 17—reference [9]).

5.3.2 SCPS-DEFINED REQUIREMENTS

5.3.2.1 Endpoint Congestion Control Algorithm

The Congestion Control Algorithm MIB entry identifies the type of congestion control that is in effect at the transport endpoint. If the congestion control algorithm may be selected on a per-connection basis, then the value of this MIB parameter shall be set to 'per-connection'.

Name: endpointCongControlAlgo
Syntax: Integer {
None (0),
Van Jacobson Congestion Control (1),
TCP Vegas Congestion Control (2),
Open Loop Rate Control (3),
Other (4),
Per-Connection (5)
}
Access by TP: Read/write

5.3.2.2 Per Connection Congestion Control Algorithm

If the Endpoint Congestion Control Algorithm MIB indicates that the congestion control algorithm may be selected on a per-connection basis, then the connectionCongControlAlgo information, specified below, shall be added to the tcpConnEntry information specified in reference [9], page 49.

Name: connectionCongControlAlgo
Syntax: Integer {
None (0),
Van Jacobson Congestion Control (1),
TCP Vegas Congestion Control (2),
Open Loop Rate Control (3),
Other (4)
}
Access by TP: Read/write

5.3.2.3 Best Effort Send Invocations

Best Effort Send Invocations is the number of times that a Best Effort pseudo-acknowledgment has been generated on a connection that is configured to perform retransmissions (i.e., maximum retransmission count is greater than zero).

Name: bestEffortSendInv
Syntax: Counter
Access by TP: Read/write

5.3.2.4 Best Effort Receive Invocations

Best Effort Receive Invocations is the number of times that a Best Effort receive error reporting missing data has been generated on a connection. (A Best Effort receive error is issued to the receiving application to identify that a 'hole' is present in the received data.)

Name: bestEffortReceiveInv
Syntax: Counter
Access by TP: Read/write

5.3.2.5 Header Compression Send Invocations

Compressed Header Send Invocations is the number of times that a compressed header has been sent from this transport endpoint.

Name: comprHeaderSendInv
Syntax: Counter
Access by TP: Read/write

5.3.2.6 Header Compression Receive Invocations

Compressed Header Receive Invocations is the number of times that a compressed header has been received by this transport endpoint (whether the decompression was successful or not).

Name: comprHeaderRecvInv
Syntax: Counter
Access by TP: Read/write

5.3.2.7 Header Compression Receive Failures

Compressed Header Receive Failures is the number of times that a received compressed header has failed decompression.

Name: comprHeaderRecvFails
Syntax: Counter
Access by TP: Read/write

5.3.2.8 Default Connection Precedence

Default Connection Precedence is the default precedence to use for a connection when the user does not specify a precedence value.

Name: defConnPrecedence
Syntax: Integer (0..15)
Access by TP: Read only

5.4 MIB REQUIREMENTS FOR SCPS USER DATAGRAM PROTOCOL

5.4.1 MIB-II REQUIREMENTS

The UDP implementation shall maintain the information specified in the UDP Group of the MIB-II definition (Internet Standard 17—reference [9]).

5.4.2 SCPS-DEFINED REQUIREMENTS

Default Datagram Precedence is the default precedence to use for a datagram when the user does not specify a precedence value.

Name: defDatagramPrecedence
Syntax: Integer (0..15)
Access by TP: Read only

6 CONFORMANCE REQUIREMENTS

6.1 GENERAL REQUIREMENTS

6.1.1 PROTOCOL IMPLEMENTATION

A conforming implementation of this protocol shall

- conform to TCP (Internet Standard 7—reference [4]) and/or UDP (Internet Standard 6—reference [3]);
- conform to Host Requirements (Internet Standard 3—reference [2]) except as modified by sections 3, 4, and 6 of this document;
- implement the modifications in sections 3 and 4 of this document that apply to the mission-specific capabilities required and to the protocols being implemented;
- implement the Management Information Base interfaces specified in section 5 of this document that apply to the protocol capabilities being implemented.

6.1.2 PICS PROFORMA

An implementer shall prepare a Protocol Implementation Conformance Statement (PICS) based on a SCPS-defined PICS proforma for the SCPS-TP.

NOTE – The PICS proforma is provided in annex C of this document.

6.2 TRANSMISSION CONTROL PROTOCOL REQUIREMENTS

6.2.1 MAJOR CAPABILITIES

A conforming implementation of the SCPS Transport Protocol shall support the following capabilities in accordance with the indicated base standards:

a) data transfer as specified in RFC 793†, sections 1.5, 2.8, and 3.7, and as amended by 3.2.5 of this document;

b) addressing as specified in RFC 793, section 2.7 and RFC 1122, section 4.2.2.1;

c) fragmentation and reassembly as specified in RFC 793, sections 1.5, 2.8, and 3.3; and RFC 1122, section 4.2.2.6;

d) acknowledgment as specified in RFC 793, sections 1.5, 2.6, and 3.7; and RFC 1122, section 4.2.3.2, as modified by 6.2.2.10 of this document;

e) flow control as specified in RFC 793, section 1.5, 2.6, and 3.7; and RFC 1122, section 4.2.2.16, 4.2.2.17, and 4.2.3.3;

† In the remainder of this section, RFC numbers are used to facilitate reference to subsections within the Internet Standards. RFCs 793, 1122, and 768 are contained in references [4], [3], and [2], respectively.

f) checksum computation as specified in RFC 793, sections 1.5 and 3.1; and RFC 1122, section 4.2.2.7;

g) error recovery as specified in RFC 793, sections 1.5 and 3.7; and RFC 1122, sections 4.2.3.1 and 4.2.2.15, and as amended by 6.2.2 and 3.5 of this document;

h) precedence as specified in RFC 793, sections 3.4, 3.6, 3.8, and 3.9; and RFC 1122, section 4.2.4.2 as amended by 3.2.2 of this document;

i) multiplexing as specified in RFC 793, sections 1.5, 2.7, and 3.8; and RFC 1122, section 4.2.2.1 and 4.2.2.18;

j) connection establishment as specified in RFC 793 section 3.4 and modified by section 3.2 of this document;

k) connection termination as specified in RFC 793 section 3.5 and RFC 1122 section 4.2.2.13.

6.2.2 MISSION-SPECIFIC CAPABILITIES

6.2.2.1 General

Missions must decide whether the capabilities indicated in the following paragraphs are available on a per-connection basis, on a per-implementation basis, or on some basis in between, such as on a per-route basis.

NOTES

1 The missions for which this protocol is targeted have a broad range of characteristics. As a result of this, all capabilities will not necessarily be required in all implementations. This section describes those capabilities and the mission-specific needs that might cause them to be required.

2 Some capabilities, such as the use of header compression, are functions of the route. Others, such as the use of BETS, are functions of the application's requirements, and thus are connection-specific. All depend upon implementation support. A profile should be developed by missions (or groups of missions) that clearly states which mission-specific capabilities are available. The use of some of these capabilities must be negotiated at the time a connection is established. For existing Internet capabilities, such as window scaling and timestamps, the negotiating mechanisms are already defined as part of the capability. For SCPS-defined extensions, the negotiating mechanism is defined in 3.2.3.

6.2.2.2 Window Scale Option

If a mission has any *single connection* that may generate a bandwidth-delay product of greater than 65k octets, it should implement the Window Scale Option.

NOTES

1 If a connection maintains a 100,000 bps data rate, a round-trip delay of 5.2 seconds will create a bandwidth-delay product of approximately 65,000 octets. Similarly, if a connection maintains a 1,000,000 bps data rate, a round-trip delay of 0.52 seconds will create a bandwidth-delay product of 65,000 octets.

2 The Window Scale Option is defined in RFC 1323 (reference [5]), section 2.

6.2.2.3 Timestamps Option

6.2.2.3.1 SCPS environments that have large bandwidth-delay products, as well as those with a high probability of errors or packet loss, should use Round Trip Time Measurement (RTTM).

6.2.2.3.2 If a network has synchronized time sources throughout the network and runs the SCPS Network Protocol, the SCPS-NP source timestamp may be used reliably to ensure a short time to live in the network.

6.2.2.3.3 The Protect Against Wrapped Sequence numbers (PAWS) Option (see reference [5]) may be used

a) to resolve possible ambiguities resulting from use of the TCP Timestamps Option when use of SCPS-NP source timestamp is not possible;

b) to avoid long quiet time.

6.2.2.4 TCP For Transactions Options

6.2.2.4.1 Implementations providing the TCP for Transactions capabilities shall provide TSUs with the ability either

a) to send data as soon as possible on a connection (potentially before the completion of a three-way handshake); or

b) to wait for the completion of connection establishment before sending user data.

6.2.2.4.2 If an implementation supports the TCP for Transactions capabilities, the following features shall be implemented as described in RFC 1644 (reference [13]):

a) data structures, section 3.1;

b) options, section 3.2;

c) connection states, section 3.3;

d) processing rules, section 3.4.

6.2.2.5 Selective Acknowledgment Option

The Selective Acknowledgment (SACK) Option as defined in RFC 2018 may be used to improve performance in lossy environments.

NOTE – The use of SACK requires additional bandwidth in the acknowledgment stream; as such it may not be applicable in highly asymmetric environments.

6.2.2.6 Explicit Congestion Notification (ECN)

Implementations that may operate in ECN-capable networks should support ECN functionality as described in RFC 3168 (reference [14]). Such implementations must:

a) negotiate ECN capability during the SYN exchange (reference [14]);

b) signal to the network their ability and willingness to respond to ECN signals, (reference [14]);

c) invoke congestion control in response to ECN signals in essentially the same way as to dropped packets,(reference [14]).

NOTE – If ECN is employed, a specific congestion notification signal overrides any default assumption about the source of loss. In particular, if the default source of loss is assumed to be corruption, an ECN notification must invoke the congestion control response of the implementation/configuration.

6.2.2.7 Selective Negative Acknowledgment Option

The Selective Negative Acknowledgment (SNACK) Option, combined with modifications to the rules affecting congestion control, should be used to maximize the use of the available link capacity in moving user data.

NOTE – When faced with limited contact times and large volumes of data to transfer, it becomes imperative to ‘keep the pipe full’, that is, to maximize the use of the available link capacity in moving user data and improve throughput.

6.2.2.8 Record Boundaries Option

The Record Boundary Option may be used for missions supporting packet-oriented applications to augment TCP’s octet-stream operation with the ability to associate an end-of-record marker with a particular octet of data.

6.2.2.9 Header Compression Option

6.2.2.9.1 If a conforming implementation operates in a bandwidth-constrained environment, it may use the header-compression capabilities defined in 3.6. Use of header

compression will be on a per-route basis, and its use shall be determined by querying a route-specific MIB parameter (refer to 5.2).

6.2.2.9.2 For missions operating in an environment in which both bandwidth and code space are extremely scarce, an implementation may support only the Compressed Header formats, knowing that interoperability with standard TCP will not be possible. (To do so, however, will result in non-conformance to this Recommendation.)

6.2.2.10 Acknowledgment Frequency Reduction

6.2.2.10.1 In highly unbalanced link data-rate situations, the requirement that every second full sized segment be acknowledged (RFC 1122, section 4.2.3.2) must be waived.

NOTE – This waiver may require a change from the default TCP congestion control mechanism, since that congestion control scheme uses the volume of received acknowledgments as the basis for opening the congestion window.

6.2.2.10.2 The specific point at which the relative data rates in each direction of a link become highly unbalanced depends, to a significant degree, on planned use. However, if the ratio of link transmit rate to receive rate exceeds 50:1, failure to implement acknowledgment frequency reduction will result in performance degradation.

6.2.2.10.3 Implementations SHOULD NOT delay acknowledgments more that one RTT, and the delay should be as consistent between acknowledgments as possible to avoid adversely affecting round-trip timing.

6.2.2.11 Default Source of Data Loss

6.2.2.11.1 If a conforming implementation operates in an environment in which the primary source of data loss is not congestion, it may bypass the use of congestion control unless explicitly informed that congestion exists (via a Source Quench message—refer to SCPS-NP, reference [10], and to the Internet Control Message Protocol (ICMP) specification, RFC 792, reference [1]).

6.2.2.11.2 Specification of the default source of data loss **should** be made with knowledge of the network supporting that connection.

NOTE – Route-specific MIB information in 5.2 supports this determination.

6.2.2.12 Non-Use of Congestion Control

6.2.2.12.1 Congestion control may be omitted entirely from an implementation supporting missions in which congestion is not a possibility.

6.2.2.12.2 Implementers are STRONGLY CAUTIONED not to exercise this option without first ensuring that congestion will not cause network collapse.

6.2.2.12.3 If congestion control is not in use, the implementation shall use some mechanism for limiting transmission rate. The Route Send Speed parameter in the Management Information Base (refer to section 5.2.2.2) is provided to support local implementation of rate control.

6.2.2.12.4 Implementers are strongly cautioned not to omit congestion control without first verifying that network-wide means have been put in place to ensure that congestion will not cause network collapse.

6.2.2.12.5 When an implementation elects not to provide congestion control, the requirements to provide the following algorithms (or their TCP Vegas equivalents) are waived:

- Van Jacobson's Slow Start algorithm;
- Van Jacobson's Congestion Avoidance algorithm; and
- exponential back-off of the retransmission timer after a loss.

6.2.2.12.6 An implementation may address non-use of congestion control either on a per-connection basis (at run time) or on an implementation basis (at build-time).

NOTE – If use/non-use of congestion control is specified at run time, the routing structure may be used to retain information about whether particular routes are susceptible to congestion.

6.2.2.13 Best Effort Transport Service Option

Use of the Best Effort Transport Service (BETS) should be considered for missions having some or all of the following types of communication scenarios:

- high-speed data transfers of repeated data (such as images), for which the loss of a portion of the data (i.e., a portion of a scan line) is important only in that it might cause synchronization to be lost;
- reliable transfer of mission data in a buffer-limited environment, for which availability of new data is more important than retransmissions of old data, if buffers are exhausted;
- highly reliable operation when no acknowledgment channel is available.

6.2.2.14 Multiple Forward Transmission Capability

The MFX capability, defined in 3.7, provides an application the ability to send each segment multiple times, reducing the possibility that a retransmission will be required. (This is at the cost of network capacity to transmit the data multiple times preemptively.) This capability

provides the ability essentially to rate 1/n code particular connections without incurring that overhead for all communications on a link or network, when n is the number of times that each segment is transmitted. The MFX capability is incompatible with congestion control, and hence, if congestion control is in use, the MFX capability shall not be used.

6.2.3 HEADER FORMAT

A conforming implementation shall support the header formats described in section 3.1 of RFC 793 and in 3.6 of this document, subject to the mission-specific requirements of 6.2.2.9 of this document.

6.2.4 DETAILED REQUIREMENTS

NOTE – This section makes heavy reference to section 4 of RFC 1122 (reference [2]). Readers should have this section available while reviewing these requirements.

6.2.4.1 Interface Requirements

6.2.4.1.1 Inter-Layer Messages from the Lower Layers

6.2.4.1.1.1 A conforming implementation shall act upon the following Network-layer control messages as stated in RFC 1122, section 4.2.3.9:

a) Destination Unreachable;

b) Source Quench;

c) Time Exceeded;

d) Parameter Problem.

6.2.4.1.1.2 A conforming implementation shall act upon the following additional Network-layer control messages as defined in reference [10]:

a) Congestion Experienced;

b) Corruption Experienced;

c) Link Outage.

6.2.4.1.2 TCP/Application Layer Protocol Interface

6.2.4.1.2.1 The following features must be included as described in RFC 1122, section 4.2.4:

a) Error Report Routine;

b) Ability to pass Type Of Service (TOS) to the Network layer;

c) Flush Call;

d) Multihoming.

6.2.4.1.2.2 A conforming implementation shall provide the Application Layer Protocol (ALP) with a mechanism to specify the following parameters relating to BETS (refer to 3.3.2 for a detailed discussion of BETS operation):

a) BETS Enable—BETS operation must be enabled by an application before or at the time of connection establishment if it is to be used at any time on the connection;

b) BETS Retransmission Limit (R2)—maximum number of retransmissions before BETS send-side operation commences;

c) BETS Receive-side Buffer Threshold—percent of receive buffer occupancy at which to commence BETS receive side operation;

d) BETS Receive-side Timeout—maximum delay before delivering out-of-sequence data to application.

6.2.4.1.2.3 A conforming implementation shall provide the ALP with a mechanism to request information about data elements that were not acknowledged by the receiver before the BETS retransmission limit (R2) was exceeded. (Refer to 3.3.2 for more detail on this requirement.)

6.2.4.1.2.4 A conforming implementation shall provide the ALP with a mechanism to specify the security services that are required. For connection-oriented Transport services, this specification shall pertain to the connection, and shall be made before or at the time of connection establishment. For connectionless Transport services, the specification of required security services may occur on a per-datagram basis.

6.2.4.1.3 Network Layer Options

A conforming implementation shall provide the ability to convey Transport-user-supplied Network-layer options to the Network service.

6.2.4.1.4 Address Validation

A conforming implementation shall perform the following as described in RFC 1122, section 4.2.3.10:

a) reject OPEN call to invalid network address;

b) reject SYN from invalid network address;

c) silently discard SYN to broadcast/multicast address.

6.2.4.1.5 Management Information Base (MIB) Interface

NOTE – Refer to 6.4 of this document for Network Management requirements.

6.2.4.1.5.1 A conforming implementation shall respond to information obtained from the Management Information Base as described in 5.2 and 5.3 of this document.

6.2.4.1.5.2 A conforming implementation shall maintain TCP-writable information in the Management Information Base as described in 5.2 and 5.3 of this document.

6.2.4.2 Connection Management and Data Transfer Requirements

6.2.4.2.1 Push Flag

The push flag shall be implemented as described in RFC 1122, section 4.2.2.2; and RFC 793, section 2.8.

6.2.4.2.2 Window Management

Window management shall be implemented as described in RFC 1122, sections 4.2.2.3, 4.2.2.16, and 4.2.2.17; and RFC 793, section 3.1.

6.2.4.2.3 Urgent Data

6.2.4.2.3.1 The provision of Urgent Data is optional. If supported, Urgent Data shall be supported as defined in RFC 1122, section 4.2.2.4.

NOTE – RFC 1122 and the Berkeley implementation of TCP interpret the meaning of the Urgent Data pointer differently. RFC 793 was inconsistent in its documentation of the meaning of the urgent pointer. Users should be aware that RFC 1122 interprets the urgent pointer as pointing to the last octet of urgent data. The Berkeley implementation and implementations derived from it interpret the urgent pointer as pointing to the octet following the last octet of urgent data.

6.2.4.2.3.2 If an application attempts to write urgent data to a SCPS-TP implementation that does not support it, the SCPS-TP implementation shall fail the write and return an error indication to the application that indicates that the requested operation is not supported. A similar error shall be returned if an application attempts to read urgent data from a SCPS-TP implementation that does not support urgent data.

6.2.4.2.4 Initial Sequence Number Selection

ISN selection shall be performed as described in RFC 1122, section 4.2.2.9; and RFC 793, section 3.3, as amended by 3.2.1 of this document.

6.2.4.2.5 Opening Connections

The following features shall be implemented as described in RFC 793, section 3.4; and the following sections of RFC 1122:

a) simultaneous open attempts as in section 4.2.2.10;

b) SYN-RCVD remembers last state as in section 4.2.2.11;

c) interference from passive OPEN call as in section 4.2.2.18;

d) simultaneous LISTENs as in section 4.2.2.18;

e) broadcast/multicast address as in section 4.2.2.10;

f) discard SYN segments addressed to broadcast/multicast addresses as in section 4.2.2.10.

6.2.4.2.6 Closing Connections

The following features shall be implemented as described in RFC 793, section 3.5; and RFC 1122, sections 4.2.2.12 and 4.2.2.13:

a) RST contains data;

b) inform application of aborted connection;

c) half-duplex close connections;

d) send RST to indicate data lost;

e) TIME-WAIT state;

f) accept SYN from TIME-WAIT state.

6.2.4.2.7 Retransmissions

The following features shall be implemented as described in RFC 1122, sections 4.2.2.15 and 4.2.3.1:

a) Jacobson Slow Start algorithm, as amended by 3.4 of this document;

b) Jacobson Congestion-Avoidance algorithm, as amended by 3.4 of this document;

c) Karn's algorithm, as amended by 3.4 of this document;

d) Jacobson's Retransmission Time Out (RTO) estimation algorithm;

e) exponential backoff, as amended by 3.4 of this document;

f) SYN RTO calculation same as data;

g) recommended initial values and bounds, subject to validation in laboratory and simulation environments.

6.2.4.2.8 Generating Acknowledgments (ACKs)

6.2.4.2.8.1 A conforming implementation shall generate ACKs as described in RFC 1122, sections 4.2.2.20, 4.2.2.21, 4.2.3.2, and 4.2.3.3.

6.2.4.2.8.2 The 'delayed ACK' processing described in RFC 1122, section 4.2.3.2, shall be implemented, but the threshold for sending an ACK shall be mission dependent, possibly longer than the .5 seconds mentioned.

6.2.4.2.8.3 Missions with bandwidth limitations may choose to acknowledge less frequently than every other full-sized segment, as recommended in RFC 1122, section 4.2.3.2.

NOTE – Acknowledgments should not occur less frequently than once per RTT when RTTs are longer than .5 seconds.. The requirement in section 4.2.2.20 of RFC 1122, that a receiver process all queued data on a connection before issuing an acknowledgment, is considered a recommendation rather than a firm requirement. A receiver may acknowledge early if an acknowledgment is required to allow continued transmission by the sender (that is, receive buffers have been emptied, increasing the window size, and the receiver has acknowledged recently).

6.2.4.2.9 Sending Data

6.2.4.2.9.1 A conforming implementation shall perform the following as described in RFC 1122:

a) sender SWS-Avoidance Algorithm as in section 4.2.3.4;

b) Nagle algorithm as in section 4.2.3.4.

6.2.4.2.9.2 When an application uses the Record Boundary Option, specified in 3.3.1 of this document, the behavior of the above-mentioned algorithms shall be modified as specified in 3.3.1.3 of this document.

6.2.4.2.10 Connection Failures

6.2.4.2.10.1 When the Best Effort mode of communication is being employed, a conforming implementation shall act upon 'connection failures' as evidenced by the R1 and R2 thresholds' being exceeded in the manner described in 3.3.2 of this document.

6.2.4.2.10.2 When the Best Effort mode is not being employed, a conforming implementation shall act upon a connection failure as described in RFC 1122, section 4.2.3.5, as amended by 3.4 of this document.

6.2.4.2.11 Send Keep-Alive Packets

There are no additional requirements to those described in RFC 1122, section 4.2.3.6.

NOTE – If SCPS Header Compression is in use and the compression algorithm is implemented separately from the remainder of the TCP state machine, then the keep-alive packet MUST contain one octet of data. (This is an implementation detail, but according to the description of the compression algorithm, if there is neither data nor acknowledged flags, the sequence number does not accompany the packet. If the compression software is integrated into the protocol machine, the compressed header can be built knowing that a sequence number is required, even if no data is present. Note that this issue is independent of the means by which the decompression algorithm is implemented.)

6.3 USER DATAGRAM PROTOCOL REQUIREMENTS

6.3.1 GENERAL REQUIREMENTS

6.3.1.1 Applications wishing to use UDP for reliable communication must provide their own mechanisms for identifying data loss and for recovering from it.

6.3.1.2 Applications using UDP are responsible for ensuring that their traffic does not cause congestion within the network.

6.3.2 MAJOR CAPABILITIES

A conforming implementation of UDP shall support the following:

a) data transfer as specified in RFC 768, page 2; and RFC 1122, section 4.1.1;

b) port addressing as specified in RFC 768, pages 1 and 2; and RFC 1122, sections 4.1.1, 4.1.3.1, and 4.1.3.5, and 4.1.3.6;

c) checksum as specified in RFC 768, page 2; and RFC 1122, sections 4.1.1 and 4.1.3.4.

6.3.3 DETAILED REQUIREMENTS

6.3.3.1 Interface Requirements

6.3.3.1.1 Inter-Layer Messages from the Network Layer

A conforming implementation shall pass all error messages received from the Network layer to the Application layer.

NOTE – UDP is stateless, and any error messages resulting from sending a UDP datagram will be received asynchronously. It is the responsibility of the application to maintain sufficient state information to process these error messages, if required.

6.3.3.1.2 UDP/ALP Interface

A conforming implementation shall provide UDP Application Data Transfer Services as described in RFC 768, page 2, and shall provide the TSU access to the following network services (refer to RFC 1122, section 3.4, and/or SCPS-NP, section 3.8):

- GET_SRCADDR;
- GET_MAXSIZES;
- ADVISE_DELIVPROB;
- RECV_NETWORK_CONTROL_MSG.

6.3.3.1.3 Lower-Layer Options

6.3.3.1.3.1 A conforming implementation shall provide the ability to convey TSU-supplied Network-layer options to the Network service and TSU-supplied security-layer options to the Security service.

6.3.3.1.3.2 A conforming implementation shall provide the ability to pass incoming Network-layer options from the Network service to the TSU.

6.3.3.2 Data Transfer Requirements

6.3.3.2.1 Checksum

A conforming implementation shall allow for a checksum as described in RFC 1122, section 4.1.3.4.

6.3.3.2.2 UDP Multihoming

A conforming implementation shall provide for UDP Multihoming as described in RFC 1122, section 4.1.3.5.

6.3.3.2.3 Addressing

A conforming implementation shall provide for UDP transmission to unicast, multicast, and broadcast Network-layer addresses.

6.4 NETWORK MANAGEMENT REQUIREMENTS

Conforming systems shall maintain the management information identified in 5.4 of this document. Neither the method for remote access to this information, e.g., via Simple Network Management Protocol (SNMP), nor the format of this information for remote transmission, e.g., ASN.1 Basic Encoding Rules, are specified by this document.

ANNEX A

SYMBOLS AND ABBREVIATIONS

(This annex **is not** part of the Recommendation.)

ACK	Acknowledgment
ALP	Application Layer Protocol
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
IAB	Internet Architecture Board
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISN	Initial Sequence Number
kbps	kilobits per second
MSS	Maximum Segment Size
PAWS	Protect Against Wrapped Sequence Numbers
PICS	Protocol Implementation Conformance Statement
RFC	Request For Comments
RTO	Retransmission Time Out
RTT	Round Trip Time
RTTM	Round Trip Time Measurement
SNMP	Simple Network Management Protocol
STD	Standard
SWS	Silly Window Syndrome
SYN	Synchronization
TCP	Transmission Control Protocol

TP-ID	Transport Protocol Identifier
TSP	Transport service provider
TSU	Transport service user
TTL	Time To Live
UDP	User Datagram Protocol

ANNEX B

INFORMATIVE REFERENCES

(This annex **is not** part of the Recommendation.)

[B1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-9. Yellow Book. Issue 9. Washington, D.C.: CCSDS, November 2003.

[B2] *Space Communications Protocol Specification (SCPS)—Rationale, Requirements, and Application Notes*. Report Concerning Space Data System Standards, CCSDS 710.0-G. [Not yet published.]

[B3] J. Mogul and S. Deering. *Path MTU Discovery*. RFC 1191, November 1990.†

[B4] V. Jacobson and R. Braden. *TCP Extensions for Long-Delay Paths*. RFC 1072, October 1988.

[B5] R. Fox. *TCP Big Window and Nak Options*. RFC 1106, June 1989.

[B6] V. Jacobson. *Compressing TCP/IP Headers for Low-Speed Serial Links*. RFC 1144, February 1990.

[B7] K. Nichols, S. Blake, F. Baker, and D. Black. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474, December 1998.

† Internet Request for Comments (RFC) texts are available on line in various locations (e.g., http://ietf.org/rfc/).

ANNEX C

PROTOCOL IMPLEMENTATION CONFORMANCE STATEMENT PROFORMA

(This annex **is** part of the Recommendation.)

C1 INTRODUCTION

C1.1 OVERVIEW

This annex provides the Protocol Implementation Conformance Statement (PICS) Requirements List (PRL) for implementations of SCPS-TP. The PICS for an implementation is generated by completing the PRL in accordance with the instructions below. An implementation shall satisfy the mandatory conformance requirements of the base standards referenced in the PRL.

An implementation's completed PRL is called the PICS. The PICS states which capabilities and options of the protocol have been implemented. The following can use the PICS:

- the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
- the supplier and acquirer or potential acquirer of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- the user or potential user of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSes);
- a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

C1.2 NOTATION

The following are used in the PRL to indicate the status of features:

Status Symbols

M mandatory.

M.<n> support of every item of the group labeled by the same numeral <n> required, but only one is active at a time.

O optional.

O.<n> optional, but support of at least one of the group of options labeled by the same numeral <n> is required.

C conditional.

- non-applicable field/function (i.e., logically impossible in the scope of the PRL).

I out of scope of PRL (left as an implementation choice).

X excluded or prohibited.

Two character combinations may be used for dynamic conformance requirements. In this case, the first character refers to the static (implementation) status, and the second refers to the dynamic (use) status; thus 'MO' means 'mandatory to be implemented, optional to be used'.

Notations for Conditional Status

The following predicate notations are used:

<predicate>:: This notation introduces a group of items, all of which are conditional on <predicate>.

<predicate>: This notation introduces a single item which is conditional on <predicate>.

In each case, the predicate may identify a protocol feature, or a Boolean combination of predicates. ('^' is the symbol for logical negation, '|' is the symbol for logical OR, and '&' is the symbol for logical AND.)

<index>: This notation indicates that the status following it applies only when the PICS states that the features identified by the index are supported. In the simplest case, <index> is the identifying tag of a single PRL item. The symbol <index> also may be a Boolean expression composed of several indices.

<index>:: This notation indicates that the associated clause should be completed.

Notations Used in the Protocol Feature Column

<r> Symbol used to denote the receiving system.

<t> Symbol used to denote the transmitting system.

Support Column Symbols

The support of every item as claimed by the implementer is stated by entering the appropriate answer (Y, N, or N/A) in the support column:

Y Yes, supported by the implementation.

N No, not supported by the implementation.

N/A Not applicable.

C1.3 REFERENCED BASE STANDARDS

The base standards referenced in the PRL are:

- SCPS-TP (this document);
- RFC 793 (reference [4]);
- RFC 768 (reference [3]);
- RFC 1122 (reference [2]);
- RFC 1644 (reference [13]);
- RFC 1323 (reference [5]).

In the tables below, the notation in the Reference column combines one of the short-form document identifiers above (e.g., SCPS-TP) with applicable subsection numbers in the referenced document. RFC numbers are used to facilitate reference to subsections within the Internet Standards.

C1.4 GENERAL INFORMATION

C1.4.1 IDENTIFICATION OF PICS

Ref	Question	
1	Date of Statement (DD/MM/YYYY)	
2	PICS serial number	
3	System Conformance statement cross-reference	

C1.4.2 IDENTIFICATION OF IMPLEMENTATION UNDER TEST (IUT)

Ref	Question	Response
1	Implementation name	
2	Implementation version	
3	Machine name	
4	Machine version	
5	Operating System name	
6	Operating System version	
7	Special Configuration	
8	Other Information	

C1.4.3 IDENTIFICATION

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification - e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

C1.4.4 PROTOCOL SUMMARY

Protocol Version	
Addenda Implemented	
Amendments Implemented	
Have any exceptions been required? (Note: A YES answer means that the implementation does not conform to the protocol. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming.	Yes_____ No_____
Date of Statement	

C1.5 INSTRUCTIONS FOR COMPLETING THE PRL

An implementer shows the extent of compliance to the protocol by completing the PRL; that is, compliance to all mandatory requirements and the options that are not supported are shown. The resulting completed PRL is called a PICS. In the Support column, each response shall be selected either from the indicated set of responses, or it shall comprise one or more parameter values as requested. If a conditional requirement is inapplicable, N/A should be used. If a mandatory requirement is not satisfied, exception information must be supplied by entering a reference Xi, where i is a unique identifier, to an accompanying rationale for the noncompliance. When the requirement is expressed as a two-character combination (as defined above), the response shall address each element of the requirement; e.g., for the requirement 'MO', the possible compliant responses are 'YY' or 'YN'.

C1.6 BASIC REQUIREMENTS

The following table lists the detailed requirements for a SCPS-TP implementation, and asks if the listed protocols have been implemented:

Item	Protocol Feature	Reference	Status	Support
tcp	SCPS-TP TCP	RFC 793; SCPS-TP: 6.2, 3	O	
udp	SCPS-TP UDP	RFC 768; SCPS-TP: 6.3, 4	O	

C2 PICS PROFORMA FOR SCPS TRANSPORT PROTOCOL TCP (SCPS-TP TCP)

C2.1 GENERAL INFORMATION

C2.1.1 Identification

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification, e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

C2.1.2 Protocol Summary

Protocol Version	
Addenda Implemented	
Amendments Implemented	

Date of Statement	

C2.2 GENERAL/MAJOR CAPABILITIES

Item	Protocol Feature	Reference	Status	Support
dt	Data Transfer	RFC 793:1.5, 2.8,3.7	M	
padr	Port Addressing	RFC 793: 2.7 RFC 1122: 4.2.2.1	M	
frg	Fragmentation	RFC 793: 1.5,2.7,3.3 RFC 1122: 4.2.2.6	M	
ras	Reassembly		M	
pa	Piggyback Acknowledgment	RFC 793: 1.5,2.6,3.7 RFC 1122: 4.2.3.2	^afred:M	
flc	Flow Control	RFC 793: 1.5,2.6,3.7 RFC 1122: 4.2.3.3	M	
chks	Checksum	RFC 793: 1.5,3.1 RFC 1122: 4.2.2.7	M	
errec	Error Recovery	RFC 793: 1.5.3.7 RFC 1122: 4.2.3.1, 4.2.2.15	^scpser: M	
scpser	SCPS Error Recovery	SCPS-TP 6.2.2, 3.3.2	O	
secpr	Precedence and Security	RFC 793: 1.5, 2.9	M	
scpspr	SCPS Precedence	SCPS-TP: 3.2.2	O	
mult	Multiplexing	RFC 793: 1.5, 2.7, 3.8 RFC 1122: 4.2.2.1, 4.2.2.18	M	
conest	Connection Establishment	RFC 793: 3.4, SCPS-TP: 3.2	M	
contrm	Connection Termination	RFC 793: 3.5	M	

C2.3 MISSION-SPECIFIC CAPABILITIES

Item	Protocol Feature	Reference	Status	Support
ttcp	TCP for Transactions	RFC 1644 SCPS-TP: 6.2.2.4	O	
winsc	Window Scaling	RFC 1323	O	
timest	TCP Timestamps	RFC 1323	O	
snack	Selective Negative Acknowledgment	SCPS-TP: 6.2.2.5, 3.5	O	
sack	Selective Acknowledgment	RFC 2018	O	
ecn	Explicit Congestion Notification	RFC 3168	O	
recbnd	Record Boundaries	SCPS-TP: 6.2.2.8, 3.3.1	O	
hdrcm	SCPS-TP Header Compression	SCPS-TP: 6.2.2.9, 3.5	O	
afred	ACK Frequency Reduction	SCPS-TP 6.2.2.10	O	
deflos	Default Source of Data Loss	SCPS-TP: 6.2.2.11	O	
nocc	Non-use of congestion control	SCPS-TP: 6.2.2.12	O	
bets	Best Effort Transport Service	SCPS-TP: 6.2.2.13, 3.3.2	O	
mfx	Multiple Forward Transmission	SCPS-TP: 6.2.2.14	nocc:O ^nocc:X	

C2.4 INTERFACES

Item	Protocol Feature	Reference	Status	Support
nlcm	Network Layer Control Messages	RFC 1122: 4.2.3.9 SCPS-TP: 6.2.4.1.1	M	
alp	TCP/Application Layer Protocol	RFC 1122: 4.2.4 SCPS-TP: 6.2.4.1.2	M	
nlopt	Network Layer Options	SCPS-TP: 6.2.4.1.3	M	
av	Remote Address Validation	RFC 1122: 4.2.3.10 SCPS-TP: 6.2.4.1.4	M	
mib	Management Information Base	SCPS-TP: 6.2.4.1.5, 5.2, 5.3	M	
afred	ACK Frequency Reduction	SCPS-TP 6.2.2.10	O	
deflos	Default Source of Data Loss	SCPS-TP: 6.2.2.11	O	
nocc	Non-use of congestion control	SCPS-TP: 6.2.2.12	O	
bets	Best Effort Transport Service	SCPS-TP: 6.2.2.13, 3.3.1	O	

C2.5 FRAME STRUCTURE

C2.5.1 Frame Structure—Uncompressed SCPS-TP TCP Headers

Item	Protocol Feature	Reference	Status	Support
hdr	Header Format	RFC 793: 3.1	M	
sp	Source Port		M	
dp	Destination Port		M	
sn	Sequence Number		M	
an	Acknowledgment Number		M	
dof	Data Offset		M	
res	Reserved field (must be zero)		M	
ece	Explicit Congestion Notification ECE-Echo flag	RFC 3168	O	
cwr	Explicit Congestion Notification Congestion Window Reduced flag	RFC 3168	O	
	Control Bits: (following six fields)			
urgfl	Urgent Pointer field—significant	RFC 793: 3.1	M	
ack	Acknowledgment field—significant	RFC 793: 3.1	M	
psh	Push function	RFC 793: 3.1 RFC 1122:4.2.2.2	M	
rst	Reset the connection	RFC 793: 3.1	M	
syn	Synchronize sequence numbers		M	
fin	No more data from sender		M	
win	Window	RFC 793: 3.1 RFC 1122:4.2.2.3 4.2.2.16, 4.2.2.17	M	
	Checksum			
urg	Urgent data field present	RFC 793: 3.1	M	
urg1	Urgent data semantics	RFC 793: 3.1 RFC 1122:4.2.2.4 SCPS-TP: 6.2.4.2.3	O	
urg2	Run time error to application if urgent data semantics not supported and application attempts to use	SCPS-TP: 6.2.4.2.3		
opt	Options	RFC 793: 3.1 RFC 1122: 4.2.2.5, 4.2.2.6	M	
eol	End of Option List	RFC 793: 3.1	M	
nop	No-Operation		M	
mss	Maximum Segment Size		M	
winsc	Window Scaling	RFC 1323	O	
timest	TCP Timestamps	RFC 1323	O	
scps	SCPS Capabilities Option	SCPS TP: 3.2.3	snack | hdrcm | bets:M	
scpsext	Extended SCPS Capabilities	SCPS TP: 3.2.5	O	
scps1	Operate as standard TCP if SCPS Capabilities option not returned	SCPS TP: 3.1.3	scps&errec&pa:M	
scps2	Abort connection attempt if SCPS Capabilities option not returned and standard TCP functions not implemented	SCPS TP: 3.1.3	scps&(^errec | ^pa)	
sn1	Selective Negative Acknowledgment Option	SCPS TP: 6.2.2.5, 3.5	snack:M	
recbnd	Record Boundary Option	SCPS TP: 6.2.2.8, 3.3.1	O	

C2.5.2 Frame Structure—Compressed SCPS-TP TCP Headers

Item	Protocol Feature	Reference	Status	Support
chdr	Compressed Header Format	SCPS TP: 3.6.1	hdrcm:M	
conid	Connection Identifier	SCPS TP: 3.6.2	hdrcm:M	
bitv1	Bit-Vector: (following eight fields)	SCPS TP: 3.6.2		
more	More Bit-Vector follows		hdrcm:M	
ts1	Outbound timestamp present		hdrcm:M	
ts2	Echo reply timestamp present		hdrcm:M	
rb	End of user data is end of record		hdrcm:M	
sn2	Short-form SNACK present		hdrcm:M	
psh1	Push function		hdrcm:M	
seq1	Sequence number present		hdrcm:M	
ack1	Window and acknowledgment number present		hdrcm:M	
bitv2	Bit-Vector: (following eight fields)			
copts	Uncompressed options present		hdrcm:M	
cpad	Padding to 16-bit boundary present		hdrcm:M	
curg	Urgent pointer field present		hdrcm:M	
cackr	ACK Flag set to '0' on RST		hdrcm:M	
ece	ECN-Echo flag	RFC 3168	hdrcm:M	
crst	Reset the connection		hdrcm:M	
cwr	ECN 'Congestion Window Reduced' flag	RFC 3168	hdrcm:M	
cfin	No more data from sender		hdrcm:M	
curp	Urgent Pointer	RFC 793: 3.1 RFC 1122:4.2.2.4	hdrcm& urg1:M	
cwin	Window	RFC 793: 3.1; RFC 1122: 4.2.2.3, 4.2.2.16,4.2.2.17	hdrcm:M	
cack	Acknowledgment Number	RFC 793: 3.1	hdrcm:M	
cseq	Sequence Number	RFC 793: 3.1	hdrcm:M	
csn	SNACK hole	SCPS TP: 6.2.2.5, 3.5, 3.5.2.3	hdrcm& snack:M	
cts1	Outbound timestamp	SCPS TP: 3.6.2.8	hdrcm& timest& ^netts:M	
cts2	Echo reply timestamp	SCPS TP: 3.6.2.9	hdrcm& timest:M	
copt1	Uncompressed options length	SCPS TP: 3.6.2.10	hdrcm:M	
copt2	Uncompressed options list	SCPS TP: 3.6.2.11	hdrcm:M	
pad1	Pad	SCPS TP: 3.6.2.12	hdrcm:O	
ccksm	Checksum	SCPS TP: 3.6.2.13	hdrcm:M	

C2.6 SCPS-TP TCP PROCEDURE

Item	Protocol Feature	Reference	Status	Support
mux	Multiplexing	RFC 793: 1.5, 2.7, 3.8 RFC 1122: 4.2.2.1, 4.2.2.18	M	
isnsel	Use clock-driven ISN selection	RFC 793: 3.3 RFC 1122: 4.2.2.9 SCPS-TP: 3.2.1	O	
	Opening Connections			
op1	Support simultaneous open attempts	RFC 793: 3.4 RFC 1122: 4.2.2.10	M	
op2	SYN-RCVD remembers last state	RFC 793: 3.4 RFC 1122: 4.2.2.11	M	
op3	Passive open call interfere with others	RFC 793: 3.8 RFC 1122: 4.2.2.18	x	
op4	Function: simultaneous LISTENs for same port	RFC 793: 3.8 RFC 1122: 4.2.2.18	M	
op5	Ask network protocol for source address for SYN if necessary	RFC 1122: 4.2.3.7	M	
op5a	Otherwise, use local address of connection	RFC 1122: 4.2.3.7	M	
op6	OPEN to broadcast/multicast IP Address	RFC 1122: 4.2.3.10	X	
op7	Silently discard segment to broadcast/multicast address		M	
op8	Provide ability to send user data before completion of connection establishment	RFC 1644	ttcp:M	
op9	Allow user ability to wait until connection established before sending data	RFC 1644	ttcp:M	
	Closing Connections:			
cl1	RST can contain data	RFC 1122: 4.2.2.12	O	
cl2	Inform application if connection is aborted	RFC 793: 3.4	M	
cl3	Half-duplex close connections	RFC 1122: 4.2.2.13	O	
cl31	Send RST to indicate data lost		O	
cl4	In TIME-WAIT state for 2xMSL seconds		M	
cl41	Accept SYN from TIME-WAIT state		O	
	Retransmissions:			
rtc	Retransmission Timeout Calculation	RFC 793, 1.5,3.7, RFC 1122, 4.2.3.1 , 4.2.2.15	M	
rt1	Jacobson slow start algorithm	RFC 1122: 4.2.2.15 SCPS-TP: 3.3.2	^nocc:O.1	
rt2	Jacobson congestion avoidance algorithm		^nocc:O.2	
rt3	TCP Vegas slow start algorithm	Brackmo94 SCPS-TP: 3.3.2	^nocc:O.1	
rt4	TCP Vegas congestion avoidance algorithm		^nocc:O.2	
rt5	Retransmit with same IP ident	RFC 1122: 4.2.2.15	O	

Item	Protocol Feature	Reference	Status	Support
rt6	Karn's algorithm	RFC 1122: 4.2.3.1	M	
rt7	Jacobson's RTO estimation algorithm	SCPS-TP: 3.3.2	M	
rt8	Exponential Backoff[1]		^nocc:M nocc:O	
rt9	SYN RTO calculation same as data		O	
rt10	Recommended initial values and bounds		O	
	Generating ACKs:			
ga1	Queue out of order segment	RFC 1122: 4.2.2.20	O	
ga2	Process all queued before send ACK	SCPS-TP: 6.2.4.2.8	M	
ga3	Send ACK for out of order segment	RFC 1122: 4.2.2.21 SCPS-TP: 6.2.4.2.8	O	
ga4	Delayed ACKs	RFC 1122: 4.2.3.2	O	
ga41	Delay < 0.5 seconds	SCPS-TP: 6.2.4.2.8	^afred&ga4:M afred:O	
ga42	Every 2nd full sized segment ACKed		^afred&ga4:M afred&ga4:O	
ga5	Receiver SWS-Avoidance Algorithm	RFC 1122: 4.2.3.3	M	
	Sending Data:			
sd1	Configurable TTL	RFC 1122: 4.2.2.19	M	
sd2	Sender SWS-avoidance algorithm	RFC 1122: 4.2.3.4	M	
sd3	Nagle algorithm		O	
sd31	Application can disable Nagle algorithm		sd3:M	
	Connection Failures:			
cf1	Negative advice to network layer on R1 retransmissions	RFC 1122: 4.2.3.5	^bets:M bets:O	

[1] Exponential backoff is mandatory only when congestion is the suspected or confirmed cause of loss. If another cause of loss is signaled, or if no cause of loss is signaled and the assumed cause of loss is not congestion, *use* of the exponential backoff is optional. However, when congestion control is implemented, *implementation* of exponential backoff is mandatory.

C3 PICS PROFORMA FOR USER DATAGRAM PROTOCOL (UDP)

C3.1 GENERAL INFORMATION

C3.1.1 Identification

Supplier	
Contact Point for Queries	
Implementation name(s) and Versions	
Other Information Necessary for full identification—e.g., name(s) and version(s) for machines and/or operating systems; System Name(s)	

C3.1.2 Protocol Summary

Protocol Version	
Addenda Implemented	
Amendments Implemented	
Have any exceptions been required? (Note: A YES answer means that the implementation does not conform to the Transmission Control Protocol/User Datagram Protocol. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming.	Yes _____ No _____
Date of Statement	

C3.2 GENERAL/MAJOR CAPABILITIES

Item	Protocol Feature	Reference	Status	Support
dt	Data Transfer	RFC 768, RFC 1122: 4.1.1	M	
pa	Port Addressing	RFC 768, RFC 1122: 4.1.1, 4.1.3.1, 4.1.3.5, 4.1.3.6	M	
uchks	Checksum	RFC 768 RFC 1122: 4.1.1, 4.1.3.4	M	

C3.3 INTERFACES

Item	Protocol Feature	Reference	Status	Support
uuc	User/UDP Interface	RFC 768	M	
urp	Creation of new receive ports	RFC 768	M	
usend	Send operations that specify data, source, and destination ports	RFC 768	M	
urecc	Receive operations that return the data octets and source port and source address	RFC 768	M	
uprec	Send operations can specify network layer precedence	RFC 768	M	
usec	Send operations can specify security service request	RFC 768	M	
ulli	UDP/Lower Level Interface	RFC 768	M	
ulli1	Determine source network layer address	RFC 768	M	
ulli2	Determine destination network layer address	RFC 768	M	
ulli3	Pass ALP-requested network layer precedence to network layer	RFC 768	M	
ulli4	Pass ALP-requested security service request to security layer	RFC 768	M	

C3.4 FRAME STRUCTURE

Item	Protocol Feature	Reference	Status	Support
uhf	Header Format	RFC 768	M	
usp	Source Port	RFC 768	M	
udp	Destination Port	RFC 768	M	
ucb	Length	RFC 768	M	
ucks	Checksum	RFC 768	M	

C3.5 UDP PROCEDURE

Item	Protocol Feature	Reference	Status	Support
us	UDP Send port unreachable	SCPS-TP: 6.3.3.1.2	O	
	Network layer options in UDP:			
nu1	Pass received network layer options to application layer	SCPS-TP: 6.3.3.1.3	M	
nu2	Application layer can specify network options to Send	SCPS-TP: 6.3.3.1.3	M	
nu3	UDP passes network layer options down to network layer	SCPS-TP: 6.3.3.1.3	M	
ncm	Pass network control msgs up to application layer	SCPS-TP: 6.3.3.1.1	M	
	UDP checksums:			
uc1	Able to generate/check checksum	RFC 1122: 4.1.3.4	M	
uc2	Silently discard bad checksum	RFC 1122: 4.1.3.4	M	
uc3	Sender Option to not generate checksum	RFC 1122: 4.1.3.4	O	
uc31	Default is to checksum	RFC 1122: 4.1.3.4	M	
uc4	Receiver Option to require checksum	RFC 1122: 4.1.3.4	O	
	UDP Multihoming:			
um1	Pass specific-dest address to application	RFC 1122: 4.1.3.5	M	
um2	Applic Layer can specify local network layer address	RFC 1122: 4.1.3.5	M	
um3	Applic layer specify wild local network layer address	RFC 1122: 4.1.3.5	M	
um4	Application layer notified of local network layer address used	RFC 1122: 4.1.3.5	O	
bnla	Bad network layer source address silently discarded by UDP or network layer	RFC 1122: 4.1.3.6	M	
vnla	Only send valid network layer source address	RFC 1122: 4.1.3.6	M	
	UDP Application Interface Services:			
ua1	Network layer services available to application: GET_SRCADDR GET_MAXSIZES ADVISE_DELIVPROB RECV_NETWORK_CONTROL_MSG	SCPS-TP: 6.3.3.1.2	M	

ANNEX D

SERVICES OF THE TRANSPORT PROTOCOL

(This annex **is** part of the Recommendation.)

D1 DEFINITIONS

confirm (primitive): a primitive issued by a service-provider to complete, at a particular service-access-point, some procedure previously invoked by a request at that service-access-point.

indication (primitive): a primitive issued by a service provider either to invoke some procedure or to indicate that a procedure has been invoked by the service user at a peer service-access-point.

primitive (also service primitive): an abstract, implementation-independent interaction between a service-user and the service-provider.

request (primitive): a primitive issued by a service-user to invoke some procedure.

response (primitive): a primitive issued by a service-user to complete, at a particular service-access-point, some procedure previously invoked by an indication at that service-access-point.

service primitive: see primitive.

D2 OVERVIEW OF THE TRANSPORT SERVICE

D2.1 INTRODUCTION

This annex specifies the services offered by the Transport Protocol (SCPS-TP) to Transport Service Users (TSUs).[1] The goals of this annex are to provide the motivation for protocol mechanisms and to provide the TSUs with a definition of the capabilities provided by SCPS-TP. The SCPS-TP specifies three distinct *types* of Transport service: fully reliable, partially reliable, and unacknowledged. Note that TSUs are typically either upper-layer protocols or end-user processes, *not* human users.

In brief terms, the fully reliable type of service ensures that *all* data submitted to the Transport service are received by the specified destination *in order* and *without error*. The partially reliable type of service guarantees *error-free* and *in-order* delivery, but not necessarily completeness. The unacknowledged type of service guarantees *error-free* delivery, but neither completeness nor in-order delivery.

[1] This service definition is derived from work initially published in MIL-STD-1778 (Transmission Control Protocol), but it has been extended to reflect SCPS-TP enhancements to the Transmission Control Protocol (TCP) and to incorporate a service specification for the User Datagram Protocol (UDP).

The fully reliable and partially reliable types of service are provided by a variation of the Transmission Control Protocol (TCP), while the unacknowledged type of service is provided by the User Datagram Protocol (UDP). Table D-1 shows the specific services available for each of the three types of service. Note that there are four major service categories: Multiplexing, Connection Management, Data Transport, and Error Reporting. Each of these service categories is described in annex D.

Note that the unacknowledged type of service has no connection management services. This is because it is provided by UDP, which is a *connectionless* protocol, meaning that no state exchange is required before data can be transferred. Rather, all data necessary to effect the transfer are carried with each packet of data. This affects the nature of the data transport service offered to the TSU and is described in D2.4.

The SCPS-TP presumes the availability of an underlying network service that provides, at a minimum, a unit data service. This service may be provided by a number of different protocol mechanisms but is specified here assuming that services similar to those provided by the SCPS-SP and SCPS-NP or by the Internet Protocol (IP) are available. If such services are not available, a convergence function to emulate those services may be required.

Table D-1: SCPS-TP Services and Types of Service

Services	Types of Service		
	Fully reliable	**Partially reliable**	**Unacknowledged**
Multiplexing	√	√	√
Connection Management	√	√	
Data Transport (see table D-2)	√	√	√
Error Reporting	√	√	√

The following paragraphs describe the four major SCPS-TP services introduced in table D-1.

D2.2 MULTIPLEXING SERVICE

The multiplexing service allows a single local SCPS-TP entity to provide service to many local TSUs. The term ‘local’ indicates that the SCPS-TP entity and the TSUs are co-resident within a host computer.

The SCPS-TP shall identify a particular Transport service within a host with a ‘port’, which is a number within the range of 0 through 65535. A port shall be concatenated with the host’s network address to identify a TSU. This concatenation of port and network address is

called a socket.[1] For the connection-oriented types of SCPS-TP service (i.e., the fully reliable and partially reliable types of service), a connection is identified by the pair of sockets corresponding to that connection.

D2.3 CONNECTION MANAGEMENT SERVICE

D2.3.1 General

SCPS-TP uses an extended version of TCP to provide the fully reliable and partially reliable types of service. These types of service use *connections* between pairs of TSUs to maintain pertinent shared state information about the communication. Connection management consists of three phases: connection establishment, connection maintenance, and connection termination.

Note that no connection management services are supported in the unacknowledged type of SCPS-TP service.

D2.3.2 Connection Establishment

SCPS-TP shall provide a means to open connections between pairs of TSUs. Connections maintain certain properties that are constant for the duration of the connection. These properties, including security and precedence levels, are specified by the TSUs at the time that the connection is opened. A TSU may request the establishment of a connection in one of two modes: active or passive. The SCPS-TP shall provide a means for a TSU to initiate an active-mode connection to another TSU that is uniquely named with a socket. SCPS-TP shall establish a connection to the named TSU if:

a) no connection between the two named sockets already exists;

b) internal SCPS-TP resources are sufficient;

c) the other TSU exists, and has simultaneously executed a matching active open request to this TSU, or previously executed a matching passive open, or previously executed a ‘global’ matching passive open.

SCPS-TP shall provide a means for a TSU to listen for and respond to active open requests from correspondent TSUs. Correspondent TSUs are named in one of two ways:

a) fully specified: A TSU is uniquely named by a socket. A connection is established when a matching active open is executed (as described above) by the named TSU.

[1] Note: the term ‘socket’ in this context does NOT refer to the Application Programming Interface referred to as the ‘socket interface’, nor does it refer to the data structures called ‘sockets’ maintained by that interface. A socket, in this context, simply refers to a port and network address combination. This document does not require a particular Application Programming Interface for SCPS-TP.

b) unspecified: No socket is provided. A connection is established with any TSU executing a matching active open naming this TSU.

D2.3.3 Connection Maintenance

SCPS-TP shall maintain established connections supporting the data transfer service described in D2.4. Additionally, SCPS-TP shall provide a means for a TSU to acquire current connection status with regard to connection name, data transfer progress, and connection qualities.

D2.3.4 Connection Termination

SCPS-TP shall provide a means to terminate established connections and to nullify connection attempts. Established connections can be terminated in two ways:

a) Graceful Close: Both TSUs close their side of the duplex connection, either simultaneously or sequentially, when data transfer is complete. SCPS-TP shall coordinate connection termination and prevent loss of data in transit as specified by the data transfer service.

b) Abort: One TSU independently forces closure of the connection. SCPS-TP shall not coordinate connection termination. Any data in transit may be lost.

D2.4 DATA TRANSPORT SERVICE

D2.4.1 Data Transport Characteristics

Table D-2 lists the characteristics of the SCPS-TP data transport service and identifies which of those characteristics are exhibited by each of the SCPS-TP types of service.

A check mark in the cell at the intersection of a particular data transport characteristic and a SCPS-TP type of service indicates that the data transport characteristic is exhibited by that type of service. The paragraphs that follow describe each of the characteristics in more detail.

Table D-2: SCPS-TP Data Transport Characteristics

Data Transport Characteristics	Types of Service		
	Fully Reliable	Partially Reliable	Unacknowledged
Full Duplex	√	√	
Timely	√	√	
Ordered	√	√	
Complete	√		
Labeled	√	√	√
Flow controlled	√	√	
Error checked	√	√	√

Full duplex: SCPS-TP shall support simultaneous bi-directional data flow between the correspondent TSUs.

Timely: When system conditions prevent timely delivery, as specified by the user, SCPS-TP shall notify the local TSU and take one of the following actions based on the type of service:

- in the case of the fully reliable service, SCPS-TP shall terminate the connection;
- in the case of the partially reliable service, SCPS-TP shall continue as if the data had been received and acknowledged.

Ordered: SCPS-TP shall deliver data to a destination TSU in the same sequence as it was provided by the source TSU.

Complete: SCPS-TP shall, within the limits of the underlying network (see 'Timely', above), deliver all data submitted by the local TSU to the destination TSU.

Labeled: SCPS-TP shall associate with each connection the security and precedence levels supplied by the TSUs during connection establishment. When this information is not provided by the pair of TSUs, SCPS-TP shall assume default levels. SCPS-TP shall convey a user's security requests and responses to the security provider and shall report indications and confirmations as required. For connection-oriented communication (i.e., the fully reliable and partially reliable types of service), if the precedence levels do not match during connection, the higher precedence level is associated with the connection.

Flow controlled: SCPS-TP shall regulate the flow of data across the connection to prevent congestion within the network or the remote end system from leading to service degradation and failure.

Error checked: SCPS-TP shall deliver data that is free of errors within the probabilities supported by a simple 16-bit checksum.

D2.4.2 Specific Data Transfer Capabilities

Table D-3 identifies three specific data transfer capabilities that may be available to TSUs. A check mark in the cell at the intersection of a particular data transfer capability and a SCPS-TP type of service indicates that the data transport capability is available to users of that type of service. The term 'optional' indicates that the service may not be supported by all implementations of SCPS-TP, and that TSUs should consult the Protocol Implementation Conformance Statement to ensure that the service is available. The paragraphs that follow describe each of the characteristics in more detail.

Table D-3: Specific SCPS-TP Data Transfer Capabilities

Data Transfer Capabilities	Types of Service		
	Fully Reliable	Partially Reliable	Unacknowledged
Data Stream Push	√	√	
Urgent Data Signaling	optional	optional	
Record Delimiting	optional	optional	√

Data Stream Push: SCPS-TP shall transmit any waiting data up to and including the indicated data portions to the receiving SCPS-TP without waiting for additional data from the local TSU. The receiving SCPS-TP shall deliver the data to the receiving TSU in the same manner. (When the data stream push capability is not invoked by the local TSU, SCPS-TP may attempt to aggregate user data into maximum-sized packets for transmission, in order to optimize bit-efficiency. Note that this aggregation may be affected by record delimiting by the user as described below.)

Urgent Data Signaling: SCPS-TP shall provide a means for the sending TSU to inform the receiving TSU of the presence of significant, or 'urgent' data in the upcoming data stream.

Record Delimiting: SCPS-TP shall provide a means for the sending TSU to delimit records of data, and to have that delimiter communicated to the receiving TSU. In the case of the fully reliable and partially reliable types of service, there is no restriction on the lengths of records. In the case of the unacknowledged service, the record length is limited by the maximum-length packet supported by the network layer.

D2.5 ERROR REPORTING SERVICE

SCPS-TP shall report service failure stemming from catastrophic conditions in the network environment for which SCPS-TP cannot compensate.

Note that for the partially reliable type of service, a loss of data does not constitute service failure. However, receiving TSUs shall be notified of the location and amount of data that has been lost. Additionally, sending TSUs shall be provided with the means to request such information but will not be informed without explicit request.

For the Unacknowledged type of service, data loss does not constitute a service failure, and SCPS-TP shall not report that data has been lost.

D3 SERVICES PROVIDED BY THE TRANSPORT LAYER

This subsection describes the primitives and parameters associated with the SCPS Transport service. These primitives *do not* constitute an Application Programming Interface; the primitives and parameters are specified in an abstract manner, and a conforming implementation is not constrained in the method of making these services available.

The service primitives are grouped into two classes based on the direction of information flow. Primitives to support information passed from the TSU to the SCPS-TP (that is, information passed 'downward') are called *service request primitives*. Primitives to support information passed from the SCPS-TP to the TSU (that is, information passed 'upward') are called *service response primitives.* Service request primitives and service response primitives do not necessarily occur in pairs. Not every service request necessarily results in a service response, and a service response may occur independently of a service request.

Not all service primitives are required by all types of SCPS-TP service. For example, the Unacknowledged type of SCPS-TP service does not require the service primitives associated with connection establishment and connection termination, since the Unacknowledged type of service does not support connection management.

D3.1 SERVICE REQUEST PRIMITIVES

Table D-4 identifies the service request primitives and identifies the types of service for which the service request primitive is applicable.

A check mark in the cell at the intersection of a particular service request primitive and a SCPS-TP type of service indicates that the primitive is available to users of that type of service. Some primitives accept security requirements and network requirements. These are passed to the appropriate protocol layer, possibly with modifications to reflect local policy enforcement or to reflect the additional requirements of intervening protocol layers. Refer to the service specifications of the security protocol and network protocol for further information on these requirements.

Table D-4: SCPS-TP Service Request Primitives

Service request primitive	Types of Service		
	Fully Reliable	Partially Reliable	Unacknowledged
T-PASSIVE_OPEN	√	√	
T-ACTIVE_OPEN	√	√	
T-ACTIVE_OPEN_WITH_DATA	√	√	
T-SEND_DATA_OVER_CONNECTION	√	√	
T-SEND_CONNECTIONLESS_DATA			√
T-RECEIVE_DATA_FROM_CONNECTION	√	√	
T-RECEIVE_CONNECTIONLESS_DATA			√
T-CLOSE	√	√	
T-ABORT	√	√	
T-CONNECTION_STATUS	√	√	
T-CONNECTIONLESS_STATUS			√

D3.1.1 T-PASSIVE_OPEN.request

The T-PASSIVE_OPEN.request primitive allows a TSU to listen for and respond to connection attempts from an unnamed TSU at a specified security and precedence level. Security level and precedence levels are specified within the S-Quality_of_Service and N-Basic_Quality_of_Service parameters, respectively.

The service primitive shall provide the following parameters:

```
T-PASSIVE_OPEN.request    (T-Source_Port,
                           T-Enable_Partial_Reliability_Service,
                           T-Timeout,
                           S-Quality_of_Service,
                           N-Basic_Quality_of_Service,
                           N-Extended_Quality_of_Service)
```

The T-Source_Port parameter shall contain the local port number associated with the TSU. Valid values are integers ranging from 0 through 65535.

The T-Enable_Partial_Reliability_Service parameter shall contain an integer value of zero or one. If equal to zero, the connection shall be opened in fully reliable mode. If equal to one, the connection shall be opened in Partial Reliability mode if the remote TSU requests the partial reliability service in its T-ACTIVE_OPEN.request. Note that if the remote TSU does not also request the partially reliable service via the 'enable partial-reliability service' parameter, the connection will be established in fully reliable mode.

The T-Timeout parameter shall specify the amount of time, in seconds, to wait for a connection. If the time specified in the T-Timeout parameter elapses before a connection attempt is received, an error is returned to the TSU indicating that the open attempt has timed out. The valid range of the T-Timeout parameter is a local issue.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services being requested by the TSU.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services being requested by the TSU.

D3.1.2 T-ACTIVE_OPEN.request

The T-ACTIVE_OPEN.request primitive allows a TSU to initiate a connection attempt to a named TSU at a particular security and precedence level.

The service primitive shall provide the following parameters:

```
T-ACTIVE_OPEN.request     (T-Source_Port,
                           T-Destination_Port,
                           N-Destination_Address,
```

T-Enable_Partial_Reliability_Service,
T-Timeout,
T-MFX_Transmissions,
S-Quality_of_Service,
N-Basic_Quality_of_Service,
N-Extended_Quality_of_Service)

The T-Source_Port parameter shall specify the local identifier of the TSU. A TSU may supply an implementation-specific value that indicates 'unspecified' for the T-Source_Port, in which case the port number will be selected by the transport service provider. Valid values of the T-Source_Port parameter are 'unspecified' or an integer between 0 and 65535.

The T-Destination_Port parameter shall identify the TSU at the remote network entity. Valid values of the T-Destination_Port are integers within the range between 0 and 65535.

The N-Destination_Address parameter shall contain a valid, non-multicast Internet address.

The T-Enable_Partial_Reliability_Service parameter shall contain an integer value of zero or one. If equal to zero, the connection shall be opened in fully reliable mode. If equal to one, the connection shall be opened in Partial Reliability mode if the remote TSU requests the partial reliability service.

The T-Timeout parameter shall specify the amount of time, in seconds, to wait for a connection. If the time specified in the T-Timeout parameter elapses before a connection attempt is received, an error is returned to the TSU indicating that the open attempt has timed out. The valid range of the T-Timeout parameter is a local issue.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services being requested by the TSU.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services being requested by the TSU.

D3.1.3 T-ACTIVE_OPEN_WITH_DATA.request

The T-ACTIVE_OPEN_WITH_DATA.request primitive is available only if the TCP for Transactions (RFC 1644, reference [13]) extensions are implemented and available. It allows a TSU to initiate a connection attempt to a named TSU at a particular security and precedence level, and allows the TSU to supply user data to accompany the request.

The service primitive shall provide the following parameters:

```
T-ACTIVE_OPEN_WITH_DATA.request   (T-Source_Port,
                                   T-Destination_Port,
                                   N-Destination_Address,
                                   T-Enable_Partial_Reliability_Service,
                                   T-Timeout,
                                   T-MFX_Transmissions,
                                   S-Quality_of_Service,
                                   N-Basic_Quality_of_Service,
                                   N-Extended_Quality_of_Service,
                                   T-End_Of_Record,
                                   T-SDU_Length,
                                   T-SDU)
```

The T-Source_Port parameter shall specify the local identifier of the TSU. A TSU may supply an implementation-specific value that indicates 'unspecified' for the T-Source_Port, in which case the port number will be selected by the transport service provider. Valid values of the T-Source_Port parameter are 'unspecified' or an integer between 0 and 65535.

The T-Destination_Port parameter shall identify the TSU at the remote network entity. Valid values of the T-Destination_Port are integers within the range between 0 and 65535.

The N-Destination_Address parameter shall contain a valid, non-multicast Internet address.

The T-Enable_Partial_Reliability_Service parameter shall contain an integer value of zero or one. If equal to zero, the connection shall be opened in fully reliable mode. If equal to one, the connection shall be opened in Partial Reliability mode if the remote TSU requests the partial reliability service.

The T-Timeout parameter shall specify the amount of time, in seconds, to wait for a connection. If the time specified in the T-Timeout parameter elapses before a connection attempt is received, an error is returned to the TSU indicating that the open attempt has timed out. The valid range of the T-Timeout parameter is a local issue.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services being requested by the TSU.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services being requested by the TSU.

The T-SDU_Length parameter shall contain the length, in octets, of the user-supplied transport service data unit. The range of this parameter is an implementation issue.

The T-SDU parameter is a Transport Service Data Unit that contains user data to be transmitted by the transport service. The maximum size of the T-SDU is an implementation issue.

D3.1.4 Send Data

D3.1.4.1 General

There are two service request primitives that cause data to be queued for transmission, one for use with the connection-oriented types of service (fully reliable and partially reliable), and one for use with the connectionless type of service (Unacknowledged).

D3.1.4.2 T-SEND_DATA_OVER_CONNECTION.request

The T-SEND_DATA_OVER_CONNECTION.request primitive causes data to be queued for transmission across the named connection.

The service request primitive shall provide the following parameters:

T-SEND_DATA_OVER_CONNECTION.request (T-Local_Connection_Name,
T-Push,
T-Urgent,
T-End_Of_Record,
T-SDU_Length,
T-SDU)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Push parameter shall contain an integer value of zero or one. When equal to one, it requests that the T-SDU be sent without delay, and that the Push flag be set to '1' in the protocol header. When equal to zero, the TSU makes no requests regarding the value of the Push flag.

The T-Urgent parameter shall contain an integer value of zero or one. When equal to one, it indicates that the first octet of data in the T-SDU specifies 'urgent' data, and that the protocol should treat the urgent data as specified in Internet Standard 7 (reference [4]).

The T-End_Of_Record parameter shall contain an integer value of zero or one. When equal to one, it indicates that the last octet of the T-SDU constitutes the end of a TSU-defined record.

The T-SDU_Length parameter shall contain the length, in octets, of the user-supplied transport service data unit. The range of this parameter is an implementation issue.

The T-SDU parameter is a Transport Service Data Unit that contains user data to be transmitted by the transport service. The maximum size of the T-SDU is an implementation issue.

D3.1.4.3 T-SEND_CONNECTIONLESS_DATA.request

The T-SEND_CONNECTIONLESS_DATA.request primitive causes data to be queued for transmission to the specified remote TSU, without previously establishing a connection. The service request primitive shall provide the following parameters:

T-SEND_CONNECTIONLESS_DATA.request (T-Source_Port,
T-Destination_Port,
N-Destination_Address,
S-Quality_of_Service,
N-Basic_Quality_of_Service,
N-Expanded_Quality_of_Service,
T-SDU_Length,
T-SDU)

The T-Source_Port parameter shall specify the local identifier of the TSU. A TSU may supply an implementation-specific value that indicates 'unspecified' for the T-Source_Port, in which case the port number will be selected by the transport service provider. Valid values of the T-Source_Port parameter are 'unspecified' or an integer between 0 and 65535.

The T-Destination_Port parameter shall identify the TSU at the remote network entity. Valid values of the T-Destination_Port are integers within the range between 0 and 65535.

The N-Destination_Address parameter shall contain a valid Internet address.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services being requested by the TSU.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services being requested by the TSU.

The T-SDU_Length parameter shall contain the length, in octets, of the user-supplied transport service data unit. The range of this parameter is an implementation issue.

The T-SDU parameter is a Transport Service Data Unit that contains user data to be transmitted by the transport service. The maximum size of the T-SDU is an implementation issue.

D3.1.5 Receive Data

D3.1.5.1 General

There are two service request primitives that indicate to SCPS-TP the willingness of the TSU to receive incoming data, one for use with the connection-oriented types of service (fully reliable and partially reliable), and one for use with the connectionless type of service (Unacknowledged).

D3.1.5.2 T-RECEIVE_DATA_FROM_CONNECTION.request

The T-RECEIVE_DATA_FROM_CONNECTION.request primitive allows a TSU to authorize SCPS-TP to deliver up to the specified amount of data (in octets) from the named connection to the TSU.

The service request primitive shall provide the following parameters:

T-RECEIVE_DATA_FROM_CONNECTION.request
(T-Local_Connection_Name,
T-SDU_Length)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-SDU_Length parameter shall specify the maximum amount of data to deliver, in octets. The range of the T-SDU_Length parameter is an implementation issue.

D3.1.5.3 T-RECEIVE_CONNECTIONLESS_DATA.request

The T-RECEIVE_CONNECTIONLESS_DATA.request primitive allows a TSU to authorize SCPS-TP to deliver up to the specified amount of data (in octets) of connectionless data (that is, data transmitted via the Unacknowledged type of service). The service request primitive shall provide the following parameters:

T-RECEIVE_CONNECTIONLESS_DATA.request
(T-Destination_Port,
T-SDU_Length)

The T-Destination_Port parameter shall identify the TSU at the local network entity. Valid values of the T-Destination_Port are integers within the range between 0 and 65535.

The T-SDU_Length parameter shall specify the maximum amount of data to deliver, in octets. The range of the T-SDU_Length parameter is an implementation issue.

D3.1.6 T-CLOSE.request

The T-CLOSE.request primitive allows a TSU to indicate that it has completed data transfer across the named connection.

The service request primitive shall provide the following parameter:

T-CLOSE.request (T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.1.7 T-ABORT.request

The T-ABORT.request primitive allows a TSU to indicate the named connection is to be immediately terminated.

The service request primitive shall provide the following parameters:

T-ABORT.request(T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.1.8 Status

D3.1.8.1 General

The Status service request primitives allow a TSU to query SCPS-TP regarding the current status of the named connection or of a local port for Unacknowledged service.

D3.1.8.2 T-CONNECTION_STATUS.request

The T-CONNECTION_STATUS.request primitive shall provide the following parameter:

T-CONNECTION_STATUS.request (T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.1.8.3 T-CONNECTIONLESS_STATUS.request

The T-CONNECTIONLESS_STATUS.request primitive shall provide the following parameter:

T-CONNECTIONLESS_STATUS.request (T-Local_Port_Number)

The T-Local_Port_Number parameter shall specify the local identifier of the TSU; valid values are between 0 and 65535.

D3.2 SERVICE CONFIRM PRIMITIVES

D3.2.1 General

Table D-5 identifies the service confirm primitives and identifies the types of service for which the service primitives are applicable. (Note that a service confirm primitive is issued by the service provider to complete, at a particular service-access-point, some procedure previously invoked by a request at that service-access-point. A service indication primitive is issued by the service provider either to invoke some procedure or to indicate that a procedure has been invoked by the service user at a peer service-access-point.)

Table D-5: SCPS-TP Service Confirm and Indication Primitives

Service confirm (or indication) primitive	Types of Service		
	Fully Reliable	Partially Reliable	Unacknowledged
T-OPEN_ID	√	√	
T-OPEN_FAILURE	√	√	
T-OPEN_SUCCESS	√	√	
T-DELIVER_CONNECTION_DATA	√	√	
T-DELIVER_CONNECTIONLESS_DATA			√
T-LOST_DATA_NOTIFICATION		√	
T-CLOSING (Indication)	√	√	
T-TERMINATE (Indication)	√	√	
T-CONNECTION_STATUS	√	√	
T-CONNECTIONLESS_STATUS			√
T-CONNECTION_ERROR (Indication)	√	√	
T-CONNECTIONLESS_ERROR (Indication)			√

A check mark in the cell at the intersection of a particular service primitive and a SCPS-TP type of service indicates that the primitive may be issued to users of that type of service.

D3.2.2 T-OPEN_ID.confirm

The T-OPEN_ID.confirm primitive informs a TSU of the local connection name assigned by SCPS-TP to the connection requested in the T-PASSIVE_OPEN, T-ACTIVE_OPEN, or T-ACTIVE_OPEN_WITH_DATA service request.

The service confirmation primitive shall provide the following parameters:

```
T-OPEN_ID.confirm        (T-Local_Connection_Name,
                          T-Source_Port,
                          T-Destination_Port,
                          T-Destination_Address)
```

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Source_Port parameter shall specify the local identifier of the TSU. Valid values range from 0 to 65535.

The T-Destination_Port parameter shall identify the remote TSU, if known at the time that the service primitive is issued. Valid values include ‘unknown’ and values in the range of 0 to 65535.

The T-Destination_Address parameter shall identify the Internet address of the remote transport entity, if known. Valid values include ‘unknown’ and valid Internet addresses.

D3.2.3 T-OPEN_FAILURE.confirm

The T-OPEN_FAILURE.confirm primitive informs a TSU of the failure of an Active Open service request.

The service primitive shall provide the following parameter:

T-OPEN_FAILURE.confirm (T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.2.4 T-OPEN_SUCCESS.confirm

The T-OPEN_SUCCESS.confirm primitive informs a TSU of the completion of one of the Open service requests (Active or Passive).

The service primitive shall provide the following parameter:

T-OPEN_SUCCESS.confirm (T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.2.5 Deliver

D3.2.5.1 General

There are two service confirmation primitives that inform the TSU of the arrival of data, one for use with the connection-oriented types of service (fully reliable and partially reliable), and one for use with the connectionless type of service (Unacknowledged).

D3.2.5.2 T-DELIVER_CONNECTION_DATA.confirm

The T-DELIVER_CONNECTION_DATA.confirm primitive informs a TSU of the arrival of data across the named connection.

The service confirmation primitive shall provide the following parameters:

T-DELIVER_CONNECTION_DATA.confirm (T-Local_Connection_Name,
T-SDU_Length,
T-Urgent_Flag,
T-End_of_Record,
T-SDU)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-SDU_Length parameter shall specify the length of the T-SDU, in octets.

The T-Urgent_Flag parameter shall indicate the presence of urgent data in the receive buffer, according to the semantics specified in Internet Standard 7 (reference [4]).

The T-End_of_Record parameter shall indicate that the last octet in the T-SDU constitutes the end of a record as marked by the peer TSU.

The T-SDU parameter is a Transport Service Data Unit that contains user data that has been transmitted by the peer TSU.

D3.2.5.3 T-DELIVER_CONNECTIONLESS_DATA.confirm

The T-DELIVER_CONNECTIONLESS_DATA.confirm primitive informs a TSU of the arrival of connectionless data addressed to the port specified in a preceding Receive Connectionless Data service request.

The service response primitive shall provide the following parameters:

T-DELIVER_CONNECTIONLESS_DATA.confirm
(T-Destination_Port,
T-Source_Port,
N-Source_Address,
T-SDU_Length,
T-SDU)

The T-Destination_Port parameter shall identify the TSU at the local network entity. Valid values of the T-Destination_Port are integers within the range between 0 and 65535.

The T-Source_Port parameter shall identify the TSU at the remote network entity.

The N-Source_Address parameter shall contain a valid Internet address that corresponds to the remote network entity sending the data.

The T-SDU_Length parameter shall specify the length of the T-SDU, in octets.

The T-SDU parameter is a Transport Service Data Unit that contains user data that has been transmitted by the peer TSU.

D3.2.6 T-LOST_DATA_NOTIFICATION.confirm

The T-LOST_DATA_NOTIFICATION.confirm primitive informs a TSU that data from the named transport connection has been lost. The missing data follows the last data delivered on the connection via the T-DELIVER_CONNECTION_DATA service confirmation.

The service confirmation primitive shall provide the following parameters:

T-LOST_DATA_NOTIFICATION.confirm (T-Local_Connection_Name,
T-Lost_Data_Length)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Lost_Data_Length parameter shall specify the length of the missing data in octets.

D3.2.7 T-CLOSING.indication

The T-CLOSING.indication primitive informs a TSU that the peer TSU has issued a T-CLOSE service request and that SCPS-TP has delivered or accounted for, via T-LOST_DATA_NOTIFICATION.confirm, all data sent by the remote TSU.

The service indication primitive shall provide the following parameter:

T-CLOSING.indication (T-Local_Connection_Name)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

D3.2.8 T-TERMINATE.indication

The T-TERMINATE.indication primitive informs a TSU that the named connection has been terminated and no longer exists. SCPS-TP generates this response as a result of a remote connection reset, service failure, and connection closing by the local TSU.

The service indication primitive shall provide the following parameters:

T-TERMINATE.indication (T-Local_Connection_Name,
T-Terminate_Reason)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Terminate_Reason parameter is an implementation-dependent parameter that provides information regarding the reason that the connection is terminated.

D3.2.9 STATUS RESPONSE

D3.2.9.1 T-CONNECTION_STATUS.confirm

The T-CONNECTION_STATUS.confirm primitive returns to a TSU the current status information associated with a connection named in a previous T-CONNECTION_STATUS.request primitive.

The service confirmation primitive shall provide the following parameters:

T-CONNECTION_STATUS.confirm (T-Local_Connection_Name,
T-Local_Port,
N-Local_Address,
T-Remote_Port,
N-Remote_Address,
T-Connection_State,
T-Send_Buf_Available,
T-Send_Window,
T-Unacknowledged,
T-Receive_Buf_Octets,
T-Partial_Reliability_Transmitted_Block_Count,
T-Partial_Reliability_Transmitted_Block_List,
T-Urgent_State,
S-Quality_of_Service,
N-Basic_Quality_of_Service,
N-Expanded_Quality_of_Service)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Local_Port parameter shall identify the TSU at the local network entity. Valid values of T-Destination_Port are integers within the range between 0 and 65535.

The N-Local_Address parameter shall contain a valid Internet address of the local network entity. In cases in which a single host has multiple addresses, it specifies the local address associated with the connection.

The T-Remote_Port parameter shall identify the TSU at the remote network entity.

The N-Remote_Address parameter shall contain a valid Internet address that corresponds to the remote network entity.

The T-Connection_State parameter shall identify one of the connection states specified in Internet Standard 7 (reference [4]).

The T-Send_Buf_Available parameter shall identify the amount of data, in octets, that the local transport service provider is willing to accept for transmission.

The T-Send_Window parameter shall identify the amount of data, in octets, that the local transport service provider is allowed to send to the remote transport service provider.

The T-Unacknowledged parameter shall identify the amount of data, in octets, awaiting acknowledgment by the remote transport service provider.

The T-Receive_Buf_Octets parameter shall identify the amount of data pending receipt by the local TSU.

The T-Partial_Reliability_Transmitted_Block_Count parameter shall specify the number of blocks of data transmitted by the partial-reliability service but not acknowledged by the remote transport entity before the limit on retransmissions expired.

The T-Partial_Reliability_Transmitted_Block_List parameter is a list that shall contain, for each partial-reliability transmitted block identified by the T-Partial_Reliability_Transmitted_ Block_Count, the following information: record number, octet offset from start of connection, number of lost records, number of octets of lost data.

The T-Urgent_State parameter shall contain an integer value of zero or one and shall equal one if urgent data has been received by the local transport service provider but has not been read by the local TSU. It shall equal zero otherwise.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services associated with the connection.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services associated with the connection.

D3.2.9.2 T-CONNECTIONLESS_STATUS.confirm

The T-CONNECTIONLESS_STATUS.confirm primitive returns to a TSU the current status information associated with a connection named in a previous T-CONNECTIONLESS_STATUS.request primitive.

The service confirmation primitive shall provide the following parameters:

```
T-CONNECTIONLESS_STATUS.confirm  (T-Local_Port,
                                  T-Octets_Sent,
                                  T-Octets_Received,
                                  S-Quality_of_Service,
                                  N-Basic_Quality_of_Service,
                                  N-Extended_Quality_of_Service)
```

The T-Local_Port parameter shall identify the TSU at the local network entity. Valid values of the T-Local_Port are integers within the range between 0 and 65535.

The T-Octets_Sent parameter shall identify the number of octets sent by the port identified by the T-Local_Port parameter. (The duration of this parameter is an implementation issue.)

The T-Octets_Received parameter shall identify the number of octets received by the port identified by the T-Local_Port parameter.

The S-Quality_of_Service parameter, defined in reference [11], shall specify the security services to be requested for outgoing datagrams.

The N-Basic_Quality_of_Service parameter and N-Extended_Quality_of_Service parameter, defined in reference [10], shall specify the network services to be requested for outgoing datagrams.

D3.2.10 Error

D3.2.10.1 General

The Error service indication primitives inform a TSU of illegal service requests or of errors relating to the environment.

D3.2.10.2 T-CONNECTION_ERROR.indication

The T-CONNECTION_ERROR.indication primitive shall provide the following parameters:

T-CONNECTION_ERROR.indication (T-Local_Connection_Name,
T-Error)

The T-Local_Connection_Name parameter shall contain the value returned by the T-OPEN_ID.confirm primitive.

The T-Error parameter shall contain an implementation-dependent value indicating the nature of the error.

D3.2.10.3 T-CONNECTIONLESS_ERROR.indication

The T-CONNECTIONLESS_ERROR.indication primitive shall provide the following parameters:

T-CONNECTIONLESS_ERROR.indication (T-Local_Port,
T-Error)

The T-Local_Port parameter shall identify the TSU at the local network entity. Valid values of the T-Local_Port are integers within the range between 0 and 65535.

The T-Error parameter shall contain an implementation-specific value indicating the nature of the error.

D3.3 SERVICE STATE DIAGRAMS

D3.3.1 General

This subsection presents the service state diagrams for the three types of service. The state diagrams for the fully reliable and partially reliable types of service are very similar, and are presented together in the same subsection.

D3.3.2 State Diagrams for Fully Reliable and Partially Reliable Service

The fully reliable and partially reliable types of Transport service are both provided by the same underlying protocol, which is based on the Transmission Control Protocol (TCP). As a result of this, the state diagrams describing the two types of service are presented together, with differences noted.

SCPS-TP operates in a distributed environment. TSUs interact with their local Transport Service Provider (TSP) in order to accomplish an end-to-end data transfer. The interaction between each TSU and its local TSP can be modeled with a state diagram that uses the Service Request, Indication, and Confirmation primitives described in the previous subsection. The overall Transport service can be modeled by combining the two local state diagrams into a 'split-state' model of the Transport service.

In figure D-1, each TSP maintains an independent, local perspective of the state of the communication between the two TSUs. Protocol exchanges between the two TSPs synchronize these local perspectives.

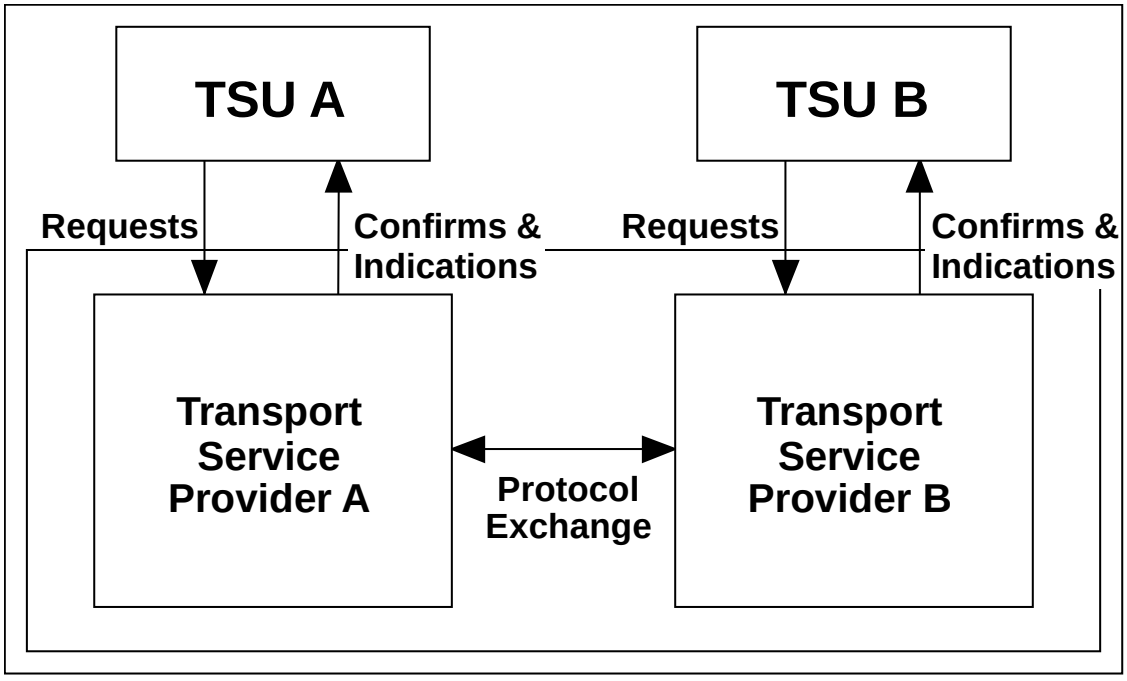

Figure D-1: SCPS-TP Composite Service Diagram for Connection-Oriented Services

Figure D-2 presents the service state diagram for the TSP. This diagram presents the sequence of state changes from the point of view of a single TSU accessing SCPS-TP's services. States are shown in boxes, and the events that cause transitions between states are shown as labeled arrows. The label on each arrow indicates the service request or SCPS-TP internal event that results in the state transition. Service requests are shown as solid arrows, while internal events are shown as dashed arrows.

In figure D-2, the initial state is 'Closed'. The Status service request primitive, not shown in the diagram, is permitted in all states except the initial 'Closed' state and causes no state transition. The Abort service request primitive, also not shown in the diagram, is permitted from all states except the initial 'Closed' state and causes a transition to the 'Closed' state. Note that the transition from 'Established' to 'Closed' may occur as a result of a protocol exchange or the detection of a service failure.

The 'Lost Data Notification' is issued to a TSU in response to a Read when the partially reliable type of service is in use and the TSP has determined that data will not be received (refer to 3.3.1 for the details of when this determination is made).

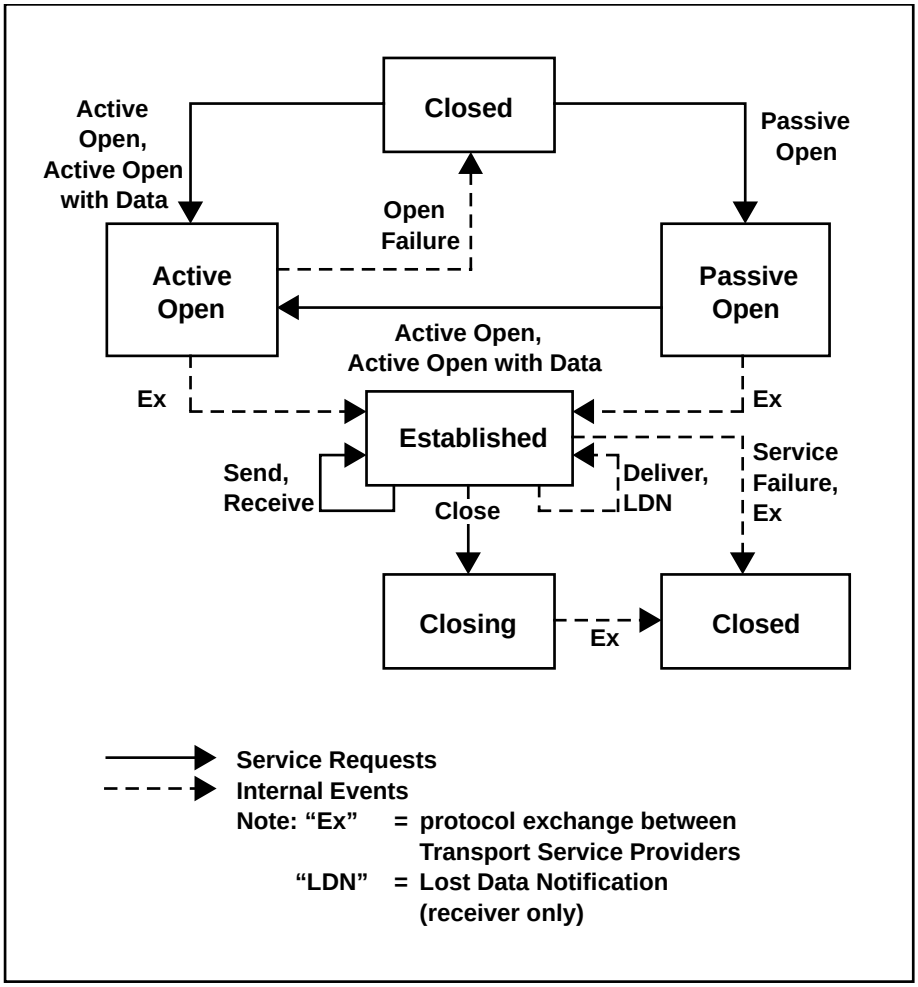

Figure D-2: Local Service Provider State Diagram

Figure D-3 shows the composite state diagram that results when two corresponding local service provider state diagrams are combined to show their interactions. Two TSUs are represented, 'A' and 'B'. The boxes represent the combined state of the two Transport services, with the state of 'A' being presented above the state of 'B'. Note that figure D-3 uses some abbreviations for state names that were not used in figure D-2. For example, the

'Active Open' state from figure D-2 is abbreviated 'Active' in figure D-3. Similarly, the 'Passive Open' state is abbreviated 'Passive', and the 'Established' state is abbreviated 'Estab'.

The events shown in figure D-3 have been abbreviated, and where appropriate, identified by the TSU or TSP causing the event. The event abbreviations are noted in the Legend appearing in figure D-3. The event initiator (A or B or both), appears in parentheses immediately following the event abbreviation. When no event initiator is listed, the event may be initiated by either side of the connection.

In addition to the state transitions shown in the figure, Status Request events can occur under the same conditions identified in the description of figure D-2, and Error Confirm events can occur in any state. Neither of these events causes a state transition.

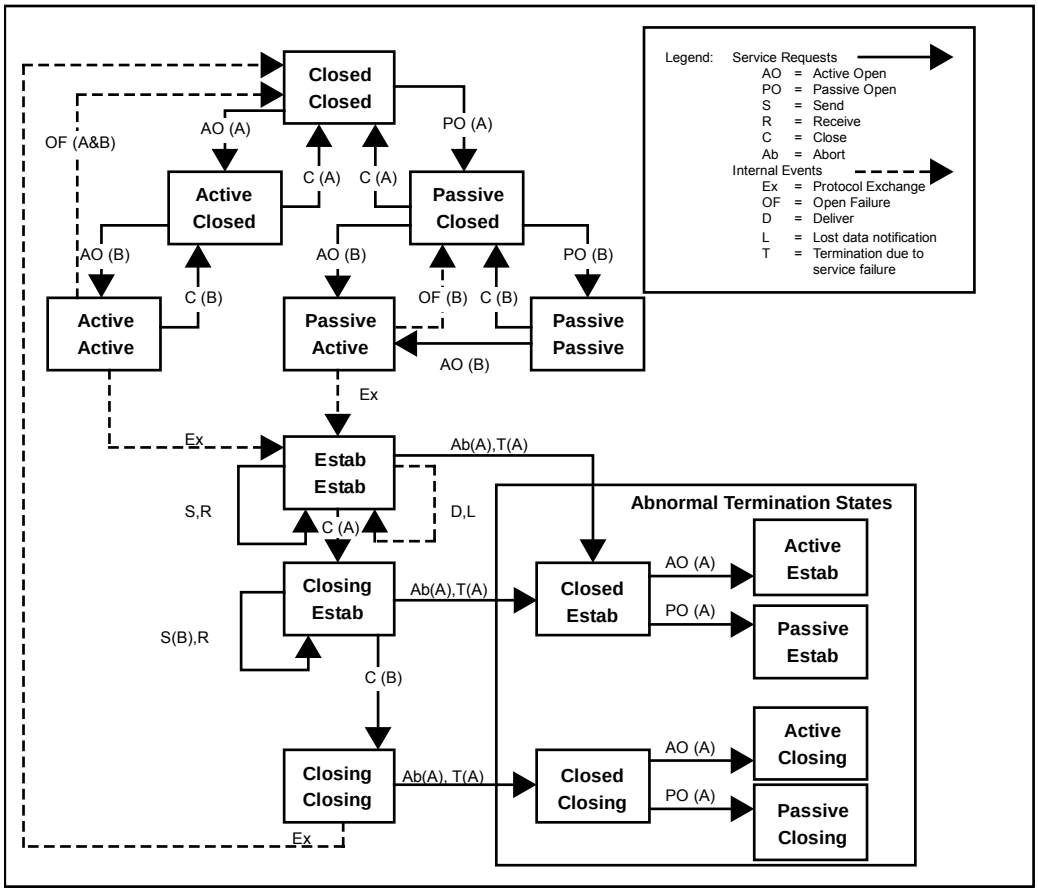

Figure D-3: Composite SCPS-TP Service State Diagram for Connection-Oriented Types of Service

The abnormal termination states shown in the lower right portion of figure D-3 identify states in which the Transport service supporting 'A' has experienced a failure of which service 'B' is unaware. The underlying protocol supporting service A attempts to inform side B of the failure, using the Reset flag in the TCP header (refer to 6.2.4.2.10). However, this

information is not sent reliably. If the packet containing the Reset is lost or damaged, side B will not be aware that side A has reset the connection. This can lead to the 'Closed/Estab' and 'Closed/Closing' states. Subsequent reuse of side A's port can lead to the remainder of the Abnormal Termination states. Note that any attempt by side B to send data to A will result in B's identifying the service failure, whereupon side B will recover its connection resources.

NOTE – Some connections may be configured with one side sending but never receiving over the connection and the other side receiving but never sending. This is not uncommon in bulk-data transfer operations, and may be typical in a telemetry data connection (which may be operated using the partial-reliability type of service). In these cases, a failed connection may not be identified by the receiver.

D3.3.3 State Diagram for Unacknowledged Service

Figure D-4 illustrates the state diagram for the unacknowledged service. The protocol providing the Unacknowledged type of service, the User Datagram Protocol (UDP), is stateless. As a result, all requests and responses result in the protocol's returning to its 'idle' state. Error responses may be issued in response to any request.

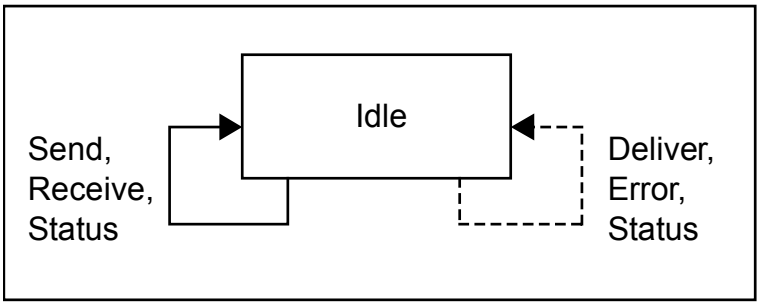

Figure D-4: State Diagram for Unacknowledged Service

D4 SERVICES ASSUMED FROM LOWER LAYERS

D4.1 SERVICE PRIMITIVES

The SCPS Transport Protocol assumes the availability of a unit data service from the lower layers. This service may be provided by a network-layer service or by a security-layer service. The primitives of the unit data service (when provided by the security layer) are

- S-UNITDATA.request;
- S-UNITDATA.indication.

The S-UNITDATA.request primitive is the service request generated by SCPS-TP to request lower-layer data transmission services. The S-UNITDATA.indication primitive is the service primitive generated by the lower layer service to deliver a Transport Protocol Data Unit (T-PDU) to the SCPS-TP.

D4.2 SERVICE PRIMITIVE PARAMETERS

Both primitives have identical parameters:

- N-Destination_Address;
- N-Source_Address;
- T-Internet_Protocol_Number;
- N-Source_Timestamp;
- S-Quality_of_Service;
- N-Basic_Quality_of_Service;
- N-Extended_Quality_of_Service;
- T-PDU.

When the underlying network service is provided by the network layer instead of the security layer, the S-Quality_of_Service is ignored. (Note that local policy may dictate that a request for security services that cannot be met shall generate an error. This is a local issue.)

The parameters to the S-UNITDATA service are as follows:

- the N-Destination_Address consists of a valid Internet address;
- the N-Source_Address consists of a valid Internet address.

The T-Internet_Protocol_Number parameter is an 8-bit field that contains, for SCPS-TP, one of the following three values as specified by the Internet Assigned Numbers Authority:

SCPS-TP Protocol	T-Internet_Protocol_Number
TCP	6
UDP	17
SCPS Compressed TCP	105

The N-Source_Timestamp parameter consists of a valid underlying network timestamp, a request that the underlying service provide a network timestamp, or a null value.

The S-Quality_of_Service parameter is a request from the TSU for security-related services such as end-to-end confidentiality and integrity.

The N-Basic_Quality_of_Service parameter and N-Expanded_Quality_of_Service parameter are requests from the TSU for network-related quality of service, such as precedence and specification of routing treatments.

D4.3 OPERATION OVER CCSDS LOWER LAYERS

The SCPS-TP assumes that it will be operating over a lower layer protocol such as the SCPS Security Protocol (SCPS-SP), the SCPS Network Protocol (SCPS-NP), or the Internet Protocol (IP). However, if it is desired for SCPS-TP to operate over the CCSDS Conventional Telemetry and Telecommand Protocols, or over the CCSDS AOS Path Service,

it is assumed that a convergence layer will be implemented in order to provide the above-defined service interface.

Neither the CCSDS Conventional nor AOS path protocols directly support the notion of individual identification of source address, destination address, or transport protocol. Rather, a single address, the Application Process Identifier (APID) captures all of this information. When operating over CCSDS Conventional or AOS path protocols, a convergence function is required to translate between IP addresses and protocol identifiers and an APID.

The N-Basic_Quality_of_Service parameter, specified in reference [10], can support a potential mapping of priority information onto prioritized Virtual Circuits, if such a capability is provided by particular implementations of the CCSDS link protocols. Security services may be provided either by the SCPS Security Protocol or by other security mechanisms operating below the transport layer. The format of the Security Quality of Service parameter is specified in reference [11].

D5 SERVICES ASSUMED FROM THE OPERATING ENVIRONMENT

SCPS-TP assumes that the operating environment provides the ability to interrogate a clock for internal protocol timing purposes.

The SCPS-TP also assumes that the operating environment provides storage to maintain transmission and possibly retransmission buffers. This storage may be ‘on line’, such as in Random Access Memory (RAM), or in sequential storage, such as might be available with a digital recorder. The amount of buffer memory required is mission dependent.