The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

SPACE LINK EXTENSION—INTERNET PROTOCOL FOR TRANSFER SERVICES

DRAFT RECOMMENDED STANDARD

CCSDS 913.1-R-0

RED BOOK

September 2005

AUTHORITY

Issue:	Red Book, Issue 0
Date:	September 2005
Location:	N/A

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

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 Recommended Standards 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 Recommended Standard 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

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING 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 **Recommended Standard** 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

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

This document is a Recommended Standard for use in developing ground systems for space missions and has been prepared by the **Consultative Committee for Space Data Systems** (CCSDS). The Application Program Interface described herein is intended for missions that are cross-supported between Agencies of the CCSDS.

This **Recommended Standard** specifies a protocol based on TCP (reference [7]) and ASN.1 (references [9] and [10]) for transfer of protocol data units defined in the Recommended Standards for Space Link Extension (SLE) Transfer Services (references [2], [3], [4], [5], and [6]) between an SLE user and an SLE provider. It allows implementing organizations within each Agency to proceed with the development of compatible, derived Standards for the ground systems that are within their cognizance. Derived Agency Standards may implement only a subset of the optional features allowed by the **Recommended Standard** and may incorporate features not addressed by the **Recommended Standard**.

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This **Recommended Standard** is therefore subject to CCSDS document management and change control procedures, as defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. 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 Program Office (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 913.1-R-0	Space Link Extension—Internet Protocol for Transfer Services, Draft Recommended Standard, Issue 0	September 2005	Pre-approval Red Book

CONTENTS

CONTENTS (continued)

Figure

Table

1 INTRODUCTION

1.1 PURPOSE

The Space Link Extension (SLE) Reference Model (reference [1]) identifies a set of SLE Transfer Services that enable missions to send forward space link data units to a spacecraft and to receive return space link data units from a spacecraft. A subset of these services is specified by the SLE Transfer Service Recommended Standards (references [2], [3], [4], [5], and [6]). The SLE Transfer Service Recommended Standards specify

a) the operations necessary to provide the transfer service;

b) the parameter data associated with each operation;

c) the behaviors that result from the invocation of each operation; and

d) the relationship between, and the valid sequence of, the operations and resulting behaviors.

However, they deliberately do not specify the methods or technologies required for communications.

The purpose of this Recommended Standard is to define a protocol for transfer of SLE Protocol Data Units (PDU) defined in the SLE Transfer Service Recommended Standards using the Internet protocols TCP (Transmission Control Protocol, reference [7]) and IP (Internet Protocol, reference [8]) for data transfer and the Abstract Syntax Notation One (ASN.1, references [9] and [10]) for data encoding. This protocol is referred to as the Internet SLE Protocol One (ISP1).

1.2 SCOPE

This Recommended Standard defines a protocol for transfer of SLE PDUs between an SLE user and an SLE provider system in terms of:

a) the procedures used to establish and release associations;

b) the messages exchanged on an established association;

c) the procedures used to monitor the status of data communication connections; and

d) the methods used to ensure that data are converted between different formats and representations on different platforms.

It does not specify:

a) individual designs, implementations, or products;

b) the configuration of the data communications infrastructure, including configuration of the TCP and IP protocols;

c) the means by which addresses (IP addresses and TCP port numbers) are agreed, assigned, and communicated.

This Recommended Standard responds to the requirements imposed by the Recommended Standards for SLE transfer services that were available when this Recommended Standard was released. The protocol specified in this Recommended Standard conforms to the requirements on data communication services set forth in those Recommended Standards.

1.3 APPLICABILITY

1.3.1 APPLICABLITY OF THIS RECOMMENDED STANDARD

This Recommended Standard provides a basis for the development of real systems that implement the Internet SLE Protocol. It is applicable for systems acting as an SLE service user or SLE service provider

1.3.2 LIMITS OF APPLICABLITY

This Recommended Standard specifies the Internet SLE Protocol that may be applied by an SLE System for inter-Agency cross support. It is neither a specification of, nor a design for, real systems that may be implemented for the control and monitoring of existing or future missions.

1.4 RATIONALE

The goal of this Recommended Standard is to create a standard for interoperability between the tracking stations and/or ground data handling systems of various agencies and the users of SLE transfer services based on the technologies TCP/IP and ASN.1.

1.5 DOCUMENT STRUCTURE

1.5.1 ORGANIZATION

This document is organized as follows:

a) section 1 presents the purpose, scope, applicability and rationale of this Recommended Standard and lists the definitions, conventions, and references used throughout the Recommended Standard;

b) section 2 describes the Internet SLE Protocol by means of an architectural model identifying individual protocol layers and the interfaces to higher layers;

c) section 3 specifies the messages exchanged via ISP1 and the procedures to be applied for connection establishment and release and for data transfer;

d) section 4 specifies the state table for the protocol;

e) annex A provides ISP1 specific diagnostic codes for the SLE PEER-ABORT operation;

f) annex B describes differences with earlier implementations of ISP1;

g) annex C lists all terms used in this document and identifies where they are defined;

h) annex D lists all acronyms used within this document;

i) annex E contains a list of informative reference documents.

1.5.2 SLE SERVICES DOCUMENTATION TREE

This Recommended Standard is part of a suite of documents specifying the SLE services. The SLE services constitute one of the three types of Cross Support Services:

a) Part 1: SLE Services;

b) Part 2: Ground Domain Services;

c) Part 3: Ground Communications Services.[1]

The basic organization of the SLE services documentation is shown in figure 1-1. The various documents are described in the following subsections.

[1] Editor's Note: the allocation of this specification to SLE services rather than to Ground Communication Services does not seem to be obvious. Is this allocation correct, and if so, does it need to be justified?

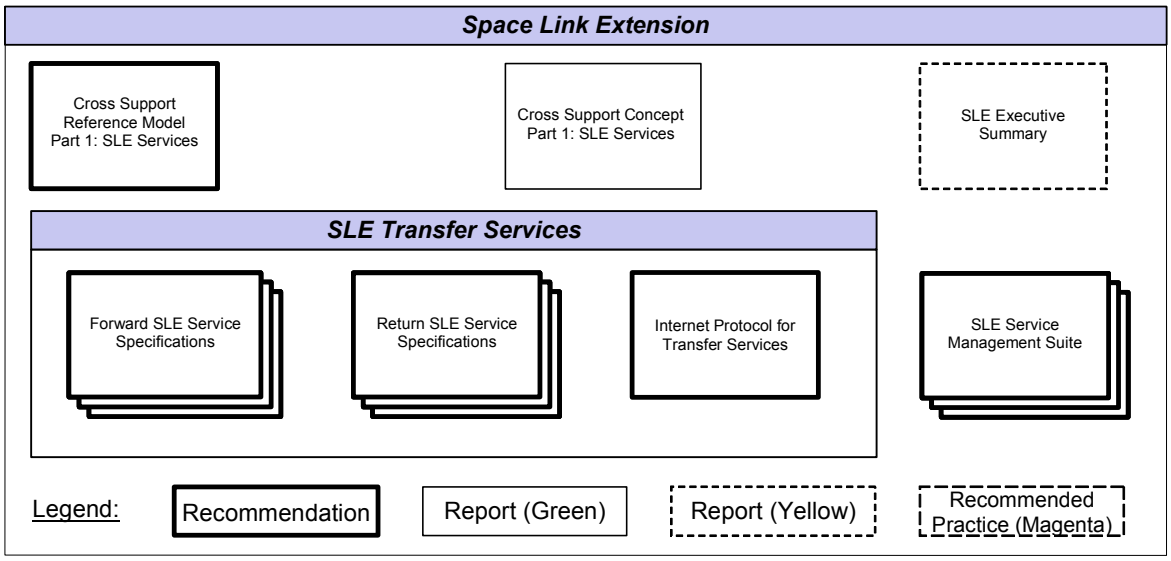

Figure 1-1: SLE Services and SLE API Documentation

a) *Cross Support Concept—Part 1: Space Link Extension Services* (reference [E2]) a Report introducing the concepts of cross support and the SLE services;

b) *Cross Support Reference Model—Part 1: Space Link Extension Services* (reference [1]): a Recommended Standard that defines the framework and terminology for the specification of SLE services;

c) *Forward SLE Service Specifications*: a set of Recommended Standards that will provide specification of all forward link SLE services;

d) *Return SLE Service Specifications*: a set of Recommended Standards that will provide specification of all return link SLE services;

e) *Internet Protocol for Transfer Services*: this Recommended Standard;

f) *SLE Service Management Specifications*: a set of Recommended Standards that establish the basis of SLE service management.

1.6 DEFINITIONS, NOMENCLATURE, AND CONVENTIONS

1.6.1 DEFINITIONS

1.6.1.1 Definitions from the SLE Reference Model

This Recommended Standard makes use of the following terms defined in reference [1]:

a) initiator;

b) operation;

c) responder;

d) service user (user);

e) service provider (provider);

f) SLE protocol data unit (SLE-PDU);

g) SLE transfer service instance (service instance).

1.6.1.2 Definitions from SLE Transfer Service Specifications

This Recommended Standard makes use of the following terms defined in references [2], [3], [4], [5], and [6]:

a) association;

b) communications service;

c) confirmed operation;

d) invocation;

e) parameter (of an operation);

f) port identifier;

g) return;

h) unconfirmed operation.

1.6.1.3 Definitions from TCP/IP

This Recommended Standard makes use of the following terms defined in references [7] and [8]:

a) Internet Protocol (IP);

b) IP address;

c) port (of TCP);

d) port number;

e) Transmission Control Protocol (TCP);

f) segment (of TCP);

g) socket.

1.6.1.4 Definitions from Abstract Syntax Notation One

This Recommended Standard makes use of the following terms defined in references [9] and [10]:

a) Abstract Syntax Notation One (ASN.1);

b) Basic Encoding Rules (BER);

c) Distinguished Encoding Rules (DER);

d) encoding rules (of ASN.1);

e) encoding;

f) module (of ASN.1).

1.6.1.5 Definitions from OSI Basic Reference Model

This Recommended Standard makes use of the following terms defined in reference [13]:

a) abstract syntax;

b) primitive;

c) (protocol) layer;

d) transfer syntax.

1.6.1.6 Additional Definitions

For the purposes of this Recommended Standard, the following definitions also apply.

1.6.1.6.1 (SLE) Application

An application is a software entity in an SLE user system or an SLE provider system that makes use of the ISP1 protocol, as distinguished from the software implementing the protocol layers defined in this Recommended Standard. The application is considered to implement the 'higher layers' defined in the architectural model in section 2.

1.6.1.6.2 Local Application

The local application is the application implementing the higher layers interfacing with a given instance of an ISP1 implementation.

1.6.1.6.3 Peer Application

The peer application is the application located on a remote network and communicating with the local application via the ISP1 protocol.

1.6.1.6.4 Application Identifier

The application identifier is the authority identifier of the application by which the application is identified in the BIND invocation and the BIND return.

NOTE – For an initiating SLE application, the application identifier is the initiator-identifier and for a responding SLE application the application identifier is the responder-identifier.

1.6.2 NOMENCLATURE

The following nomenclature applies throughout this Recommended Standard:

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.

1.6.3 CONVENTIONS

The Internet SLE Protocol is specified by a layered architecture model in which the interfaces between the layers are defined using abstract service primitives, roughly following the concepts in the OSI Basic Reference Model (reference [13]).

A service primitive is a signal optionally associated with a set of parameters that is generated by one layer and consumed by another (adjacent) layer. The direction in which the information is passed is defined by one of the following:

request	a primitive that is passed from the higher layer to the lower layer;
indication	a primitive that is passed from the lower layer to the higher layer;
response	a primitive generated by the higher layer in response to an indication of the same type;
confirmation	a primitive generated by the lower layer in response to a request of the same type.

For every service primitive, the following specifications are provided:

a) the name of the primitive;

b) the uses of the primitive (request, indication, response, confirmation);

c) the parameters associated with each of these uses.

In addition, the conditions under which the source emits a primitive and the tasks that the receiver shall perform are described.

In this Recommended Standard, primitive names are capitalized and are followed by the specific use in lower case letters. The primitive name is prefixed by the identifier of the layer that defines the interface, e.g., 'TML-CONNECT-indication'. In the state table in section 4, the prefix is omitted in order to avoid clashes with the notation used in these tables.

1.7 REFERENCES

The following documents contain provisions that, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard 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 Recommended Standards.

NOTE – A list of informative references is provided in annex E.

[1] *Cross Support Reference Model – Part 1: Space Link Extension Services*. Recommendation for Space Data System Standards, CCSDS 910.4-B-1, Blue Book. Issue 1, Washington, D.C.: CCSDS, May 1996.

[2] Space Link Extension—Return All Frames Service Specification. Recommendation for Space Data System Standards, CCSDS 911.1-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, December 2004.

[3] *Space Link Extension—Return Channel Frames Service Specification*. Recommendation for Space Data System Standards, CCSDS 911.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, December 2004.

[4] *Space Link Extension—Return Operational Control Fields Service Specification*. Recommendation for Space Data System Standards, CCSDS 911.5-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, December 2004.

[5] *Space Link Extension—Forward CLTU Service Specification*. Recommendation for Space Data System Standards, CCSDS 912.1-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, December 2004.

[6] *Space Link Extension—Forward Space Packet Service Specification*. Recommendation for Space Data System Standards, CCSDS 912.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, December 2004.

[7] *Transmission Control Protocol* – DARPA Internet Program Protocol Specification, Postel, J., ed., STD 7, RFC 793, September 1981

[8] *Internet Protocol* – DARPA Internet Program Protocol Specification, Postel, J., ed., STD 5, RFC 791, September 1981

[9] *Information technology - Open Systems Interconnection - Abstract Syntax Notation One (ASN.1) – Part 1: Specification of Basic Notation*. International Standard, ISO/IEC 8824: 1998 2nd ed. Geneva: ISO, 1998

[10] *Information technology - Open Systems Interconnection - ASN.1 Encoding Rules – Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).* International Standard, ISO/IEC 8825-1: 1998 2nd ed. Geneva: ISO, 1998

[11] *Information Technology - Open Systems Interconnection - The Directory - Part 8: Authentication framework*. International Standard, ISO/IEC 9594-8, 1993.

[12] *Secure Hash Standard*, Federal Information Processing Standards Publication 180-1, FIPS PUB 180-1.

[13] *Information Technology—Open Systems Interconnection—Basic Reference Model: The Basic Model*. International Standard, ISO/IEC 7498-1:1994. 2nd ed. Geneva: ISO, 1994.

2 DESCRIPTION OF THE INTERNET SLE PROTOCOL

2.1 INTRODUCTION

This section describes a layered model of a system supporting the Internet SLE Protocol One and introduces the main concepts by describing each layer and the interfaces between the layers. Detailed specifications are provided in section 3, which references the concepts, layers, and interfaces described by this model. It is stressed that this model has the sole purpose of supporting the specifications in this Recommended Standard and is not intended to suggest any specific design of a real implementation.

The discussion in this section assumes that the reader is familiar with the CCSDS Recommended Standards for Space Link Extension transfer services provided by references [2], [3], [4], [5], and [6], and assumes a general background on the Internet protocols, especially on TCP (reference [7]).

2.2 ARCHITECTURAL MODEL

The architectural model used to specify the Internet SLE Protocol is shown in figure 2-1.

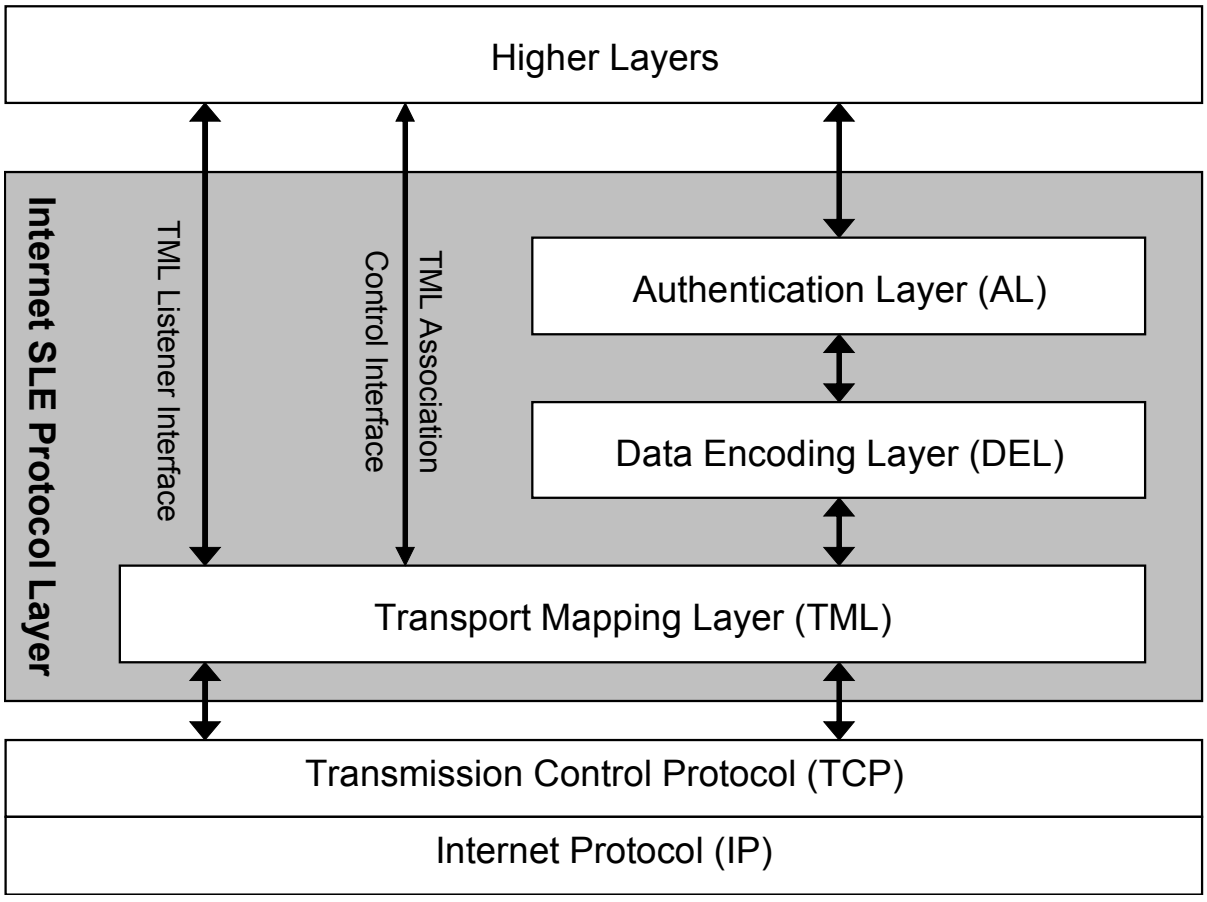

Figure 2-1: ISP1 Architecture Model

The Higher Layers represent the functionality specified by the Recommended Standards for SLE transfer services (references [2], [3], [4], [5], and [6]), including in particular:

a) preparation of SLE protocol data units to be sent to the peer application;

b) analysis and processing of the SLE protocol data units received from the peer application;

c) implementation of the state tables defined in the applicable Recommended Standard for the SLE transfer service being provided or used.

The Internet SLE Protocol Layer is further decomposed into the following sub-layers:

a) The Authentication Layer (AL) is responsible for generating and analyzing the credentials specified in the Recommended Standards for SLE transfer services. For this purpose this Recommended Standard specifies use of the simple authentication scheme defined in reference [11] and the Secure Hash Function (SHA-1) defined in reference [12].

b) The Data Encoding Layer (DEL) is responsible for encoding of SLE protocol data units received from higher layers and decoding of protocol data units received from the peer application. For this purpose, this Recommended Standard specifies use of the ASN.1 syntax defined by reference [9] and of the ASN.1 Basic Encoding Rules (BER) defined by reference [10].

c) The Transport Mapping Layer (TML) handles the interface to the Transmission Control Protocol (TCP) specified by reference [7].

ISP1 maps one TCP connection to one ISP1 association, which is used by the higher layers for one SLE association as specified by the Recommended Standards for SLE transfer services. ISP1 provides data encoding and decoding services but does not process SLE protocol data units in any other respect.

For implementations assuming the responder role, the Transport Mapping Layer provides a service to listen for incoming TCP connection requests. It therefore provides an interface by which higher layers can request to start and stop listening for a specified SLE responder port identifier, which the TML maps to an IP address and a TCP port number.

Except for the interface to control listening, all interfaces refer to one association only. ISP1 association establishment and release involves only the Transport Mapping Layer. Once the association has been established, all SLE protocol data units are passed through the Data Encoding Layer and the Authentication Layer.

ISP1 requires a number of configuration parameters for its operation, which are identified in the following sections. This Recommended Standard does not specify how these parameters are defined, agreed, stored, and made available to the implementation of the ISP1.

2.3 AUTHENTICATION LAYER

The ISP1 Authentication Layer (AL) receives SLE protocol data units from the higher layers and adds the credentials parameter if required. Likewise, it receives decoded protocol data units from the Data Encoding Layer and verifies that the credentials match the security attributes of the peer application. If authentication is not used for a given service instance or for a given PDU, the Authentication Layer passes the protocol data unit to the next layer without modification or analysis.

For generation and analysis of the credentials, the Authentication Layer uses the simple authentication scheme specified by reference [11], which is based on a secret password and a message digest generated from the that password, the time at which the credentials were generated, and a random number. For the secure one-way hash function, required for this scheme, the Secure Hash Function (SHA-1) specified by reference [12] is used.

If authentication fails for a PDU received from the peer application, the Authentication Layer informs the higher layers, which are expected to handle this event as specified by the Recommended Standards for SLE transfer services.

The Authentication Layer requires the following service instance configuration parameter for its operation:

> authentication-level, defined by the Recommended Standards for SLE transfer services, which can have the values
>
> a) 'all': all invocations and returns, except the invocation of PEER-ABORT, shall be authenticated;
>
> b) 'bind': only the BIND invocation and return shall be authenticated;
>
> c) 'none': no invocation or return shall be authenticated.

If the authentication-level is set to 'all' or 'bind' the Authentication Layer requires the following additional configuration parameters:

a) the identifier of the local application;

b) the password of the local application;

c) the identifier of the peer application;

d) the password of the peer application;

e) the maximum time allowed between generation of the credentials by the invoker of an SLE operation and verification of the credentials by the performer.

NOTE – If the local application assumes the initiator role for the BIND operation, the identifier of the local application is the initiator-identifier and the identifier of the peer application is the responder-identifier defined by the Recommended Standards for SLE transfer services. Otherwise, the identifier of the local application is the responder-identifier and the identifier of the peer application is the initiator-identifier.

2.4 DATA ENCODING LAYER

The ISP1 Data Encoding Layer (DEL) is responsible for the encoding and decoding of all SLE protocol data units, except for the PEER-ABORT invocation PDU. For this purpose it uses the ASN.1 types defined by the Recommended Standards for SLE transfer services and applies the Basic Encoding Rules (BER) defined by reference [10].

NOTE – This Recommended Standard assumes that the PEER-ABORT operation is implemented by direct interaction between the TML and higher layers. Handling of the PEER-ABORT operation by the TML is described in 2.5.5.

The DEL receives SLE PDUs from the Authentication Layer, encodes the PDU and passes the encoded PDU as a data buffer to the Transport Mapping Layer. Likewise, it receives encoded PDUs from the Transport Mapping Layer, decodes them, and passes the decoded PDU to the Authentication Layer.

When receiving a BIND invocation PDU from either the AL or the TML, DEL uses the parameters service-type and version-number to identify the ASN.1 modules to use for all PDUs subsequently exchanged on the same association. If the service type or version number is not supported, the DEL informs the higher layers, which are expected to apply the procedures specified in by the Recommended Standards for SLE transfer services.

NOTE – This Recommended Standard assumes that the DEL uses the parameters of the BIND operation to identify the ASN.1 modules to use for encoding and decoding, because no additional configuration parameters are then needed. However, the intention is not to prevent an implementation from applying any other suitable approach as long as it is ensured that the ASN.1 specification used is the one specified in the Recommended Standard that is identified by the service-type and version-number parameters in the BIND invocation.

In case of errors (e.g., decoding failure), the DEL informs the higher layers, which are expected to abort the association using the PEER-ABORT operation.

2.5 TRANSPORT MAPPING LAYER

2.5.1 INTRODUCTION

The ISP1 Transport Mapping Layer handles the interface to TCP/IP. It provides the following services:

a) mapping of logical port identifiers to TCP sockets (IP addresses and TCP port numbers);

b) establishment and release of TCP connections;

c) transfer of SLE PDUs;

d) execution of the PEER-ABORT using features available with TCP;

e) monitoring of the status of TCP connections.

In addition, this Recommended Standard defines a special connection establishment procedure, which is intended to support fast recovery from failures, provided that a responder uses redundant hosts.

2.5.2 TML MESSAGES

As the TCP user interface is based on the concept of a byte stream, this Recommended Standard defines a simple message format for transmission of SLE PDUs and exchange of protocol control information between two communicating TMLs. The following types of messages are defined:

a) an SLE PDU message for transfer of SLE PDUs except for the PEER-ABORT PDU, for which the procedure explained in 2.5.5 is used;

b) a TML context message, used to initialize a connection between two TMLs;

c) a TML heartbeat message used to monitor the status of a TCP connection.

A TML message consists of a standard message header of eight octets and a message body. The message header contains an identifier of the message-type and the number of octets in the message body. The message body is defined as follows.

2.5.2.1 SLE PDU Message

The body of the SLE PDU message consists of the encoded SLE PDU.

2.5.2.2 TML Context Message

The body of the TML context message contains the following information:

a) the identification of the protocol, i.e., the characters ‘ISP1’ in ASCII encoding;

b) the version of this Recommended Standard (the number 1);

c) the heartbeat-interval and dead-factor, used for monitoring of the status of the TCP connection (see 2.5.3).

2.5.2.3 TML Heartbeat Message

The body of the TML heartbeat message is empty. A TML heartbeat message consists of the message header only, where the length field is set to zero.

2.5.3 MONITORING OF THE STATUS OF THE TCP CONNECTION

The Recommended Standards for SLE transfer services require that SLE PDUs be transferred

a) in sequence;

b) completely and with integrity;

c) without duplication;

d) with flow control that notifies backpressure to the application layer in the event of congestion; and

e) with notification to the application layer in the event that communications between the SLE service user and the SLE service provider are disrupted, possibly resulting in a loss of data.

While the TCP fully supports requirements a) to d), it does not intrinsically provide a means of distinguishing between an idle connection and a dead connection. Therefore this Recommended Standard specifies the special procedure describes in this section.

NOTE – Some TCP implementations provide a configurable ‘keep alive’ mechanism to periodically probe idle connections. However, [E10] specifically requires that the default interval between such probes must be at least two hours.

The procedure makes use of two parameters, a heartbeat-interval and a dead-factor. The heartbeat-interval defines the maximum duration in seconds in which no message might be transmitted by a TML. A value of zero this indicates that the heartbeat mechanism shall not be used. The dead-factor defines the number of heartbeat intervals after which a TCP connection is considered dead when no message has been received from the peer TML.

The TML makes use of two timers for a heartbeat mechanism, a heartbeat-transmit-timer (HBT timer) and a heartbeat-receive-timer (HBR timer). It sets the timeout value of the HBT timer to the number of seconds defined by the heartbeat-interval and the timeout value of the HBR timer to the number of seconds defined by the heartbeat-interval multiplied by the dead-factor.

Whenever the TML transmits a message, it restarts the HBT timer. If the HBT timer expires, the TML transmits a TML heartbeat message and restarts the HBT timer.

Whenever the TML receives a message, it restarts the HBR timer. When the HBR timer expires, the TML assumes that the connection has failed, aborts using TCP-ABORT and issues the TML-PROTOCOL-ABORT-indication primitive.

Following successful establishment of the TCP connection, the TML initiating the association transmits the proposed values for the heartbeat-interval and the dead-factor as part of the TML context message, and then starts the HBT timer and the HBR timer.

When receiving the TML context message, the responding TML verifies that the values of the heartbeat-interval and the dead-factor are in the acceptable range. If the values are acceptable, the responding TML starts the HBT timer and the HBR timer. If the values are not acceptable, the TML aborts the connection using PEER-ABORT with the TML diagnostics ‘heartbeat unacceptable’.

The heartbeat mechanism remains in effect for the complete lifetime of the TCP connection. As explained in 2.5.4.2, the HBR timer is finally used to ensure that the TCP connection is released in an orderly manner.

2.5.4 CONNECTION ESTABLISHMENT AND RELEASE

2.5.4.1 Initiating TML

When receiving the primitive TML-CONNECT-request, the TML uses the port identifier to derive the IP address and the TCP port number used by the responder. It then requests the TCP to establish a connection, specifying the foreign socket. It is recommended to leave the local socket unspecified and let TCP derive the local IP address and select an ephemeral port.

If the TCP connection succeeds, the TML transmits the TML context message, issues the primitive TML-CONNECT-confirmation, and is ready to transfer SLE PDUs. If the TCP connection fails, the TML issues the primitive TML-PROTOCOL-ABORT with the appropriate diagnostics.

NOTE – This Recommended Standard assumes that the TCP implementation takes care of temporary network problems and retransmissions and that the TCP implementation can be configured as needed. Therefore, any failure reported by the TCP is assumed to be a hard error. However, this Recommended Standard does not exclude that an implementation retries connection establishment for a configurable period, or until it receives TML-RESET because the return timer has expired.

When receiving the primitive TML-DISCONNECT-request, the TML closes the TCP connection.

2.5.4.2 Responding TML

When receiving the primitive TML-START-LISTEN-request, the TML derives the TCP socket using the logical port identifier passed as argument. It then performs a 'passive open' (see 2.6.6.2), specifying the local socket.

NOTE – It is recommended not to specify the foreign socket, with the effect that the TCP accepts connections from any address. Although reference [7] specifies that the foreign socket can be defined by a call to passive open and that the TCP will then accept connections only from that socket, TCP APIs generally do not provide this option. A responder might nevertheless filter the IP addresses from which it is prepared to accept a connection, e.g., by means of a firewall.

When the TML is notified that a new connection has been accepted, it starts a configurable TML start-up timer and waits for the first message to arrive. If the timer expires before data arrive, the TML aborts the connection using TCP-ABORT. The TML checks that the first message received on a connection is a valid TML context message. If that is not the case, it aborts the connection with TCP-ABORT.

If the first message received is a valid TML context message, the TML performs the following checks:

a) The protocol identifier must be 'ISP1'; otherwise, the TML aborts using TCP-ABORT.

b) The version number must be supported by the responding TML. If the version is not supported, the TML aborts using TCP-ABORT.

c) The heartbeat parameters must be in an acceptable range; otherwise, the TML aborts using TCP-ABORT.

When these checks are passed, the TML issues the primitive TML-CONNECT-indication and is ready to receive and transmit SLE PDUs on the new connection.

When receiving TML-DISCONNECT-request, the TML terminates the HBT timer, restarts the HBR timer, and waits for an indication that the peer has closed its side of the connection.

When the peer has closed the connection, the TML closes its side of the TCP connection. If the HBR timer expires before the peer has closed the connection, the TML aborts the connection using TCP-ABORT. The TML also aborts the connection if it receives any data after receiving TML-DISCONNECT-request.

2.5.5 ABORTING CONNECTIONS

The TML provides two methods to abort a TCP connection:

a) the PEER-ABORT operation by which the diagnostic code is transmitted to the peer TML; and

b) a 'silent abort' by which no further information is made available to the peer.

The latter method is requested by the primitive TML-RESET and is mapped directly to TCP-ABORT.

A 'Close after Peer Abort' timer, CPA timer, is used to check that the peer TML reacts to the PEER-ABORT in due time.

When receiving the primitive TML-PEER-ABORT-request, the TML transmits the diagnostic parameter as one byte of TCP urgent data. It then stops the HBT and HBR timers, starts the CPA timer, and waits for the peer to close its side of the connection. The TML silently discards any data arriving after peer abort. When the peer closes the connection, the TML closes its side of the connection. If the peer does not close the connection before the CPA timer expires, the TML aborts the connection with TCP-ABORT.

When the TML is notified that urgent data are present, it reads and discards all data from the TCP receive-buffer up to but excluding the byte identified by the TCP urgent pointer. It then reads the PEER-ABORT diagnostics, closes the connection, and issues the primitive TML-PEER-ABORT-indication.

2.5.6 ERROR REPORTING

Except for the initial start-up phase on the responding side (see 2.5.4.2), the TML uses the peer abort procedure to report errors on the TML level such a badly formatted TML message or reception of a TML context message by an initiating TML. In these cases it uses one of the TML diagnostic codes defined in annex A. When a TML receives urgent data with a TML diagnostic code (i.e., a code in the range 128 to 199), it reports the problem using the primitive TML-PROTOCOL-ABORT-indication instead of TML-PEER-ABORT-indication.

NOTE – The Recommended Standards for SLE transfer services reserve PEER-ABORT diagnostic codes 0 to 127 for reporting of errors defined by these Recommended Standards. This Recommended Standard reserves the codes 128 to 199 for errors reported by the TML, leaving the codes 200 to 255 for use by implementations.

2.5.7 REDUNDANT HOSTS

If redundant hosts are used on the responding side, it is desirable to allow connection establishment, without the need to know which host is operational. Such a feature is considered particularly useful for fast recovery after failure of a connection. This version of the specification defines a very basic procedure for this purpose, which avoids the need for additional infrastructure (e.g., a directory service). More reliable and flexible procedures are left for further study and might be included in future versions.

The procedure requires an initiator to associate a set of TCP sockets with one logical responder port identifier. If the TML is requested to connect to a responder port that maps to more than one socket, it attempts to establish a TCP connection to all sockets in the set. It is expected that connection establishment will succeed for only a single socket and that all other connections will fail. If one of the connections succeeds, that TML requests TCP-ABORT on all other sockets in the set.

The procedure requires that only a single responding host listen for incoming connection requests at any time. The means by which this is achieved are outside the scope of this Recommended Standard.

2.5.8 TML CONFIGURATION

2.5.8.1 Initiating TML

An initiating TML requires the following configuration parameter for its operation:

– mapping from responder-port-identifiers to one or more TCP sockets each identified by IP address and TCP port number.

The following configuration parameters are needed for every association:

a) proposed heart-beat interval;

b) proposed dead-factor;

c) close-after-peer-abort timeout.

2.5.8.2 Responding TML

A responding TML requires the following configuration parameter for its operation:

– mapping from responder-port-identifiers to local TCP sockets.

The following configuration parameters are needed for every association:

a) acceptable range of the heartbeat-interval;

b) acceptable range of the dead-factor;

c) TML-startup-timeout;

d) close-after-peer-abort timeout.

2.6 INTERFACES

2.6.1 AUTHENTICATION LAYER INTERFACES

2.6.1.1 Overview

The primitives defined for the interface between the higher layers and the AL are listed in table 2-1.

Table 2-1: Primitives of the AL Interface

Primitive	Request	Indication	Confirmation
AL-SLE-PDU	X	X	

2.6.1.2 AL-SLE-PDU

The primitive is used to pass SLE operation invocations and returns between the higher layers and the AL. Its parameters are shown in table 2-2.

Table 2-2: Parameters of the Primitive DEL-SLE-PDU

Parameter	Request	Indication
SLE PDU	X	X
authentication-result		X
decoding-result		X

Request The request is used by the higher layers to pass PDUs to the AL for processing and transmission. If applicable, the AL generates and adds the credentials and forwards the PDU to the DEL using DEL-SLE-PDU-request.

Indication The indication is issued by the AL when a PDU received from the DEL via DEL-SLE-PDU-indication has been authenticated. The result of the authentication process is reported in the parameter authentication-result. If authentication of a PDU is not required, the effect is as if the PDU were successfully authenticated. The decoding result is copied from the DEL-SLE-PDU-indication.

2.6.2 DATA ENCODING LAYER INTERFACE

2.6.2.1 Overview

The primitives defined for the interface between the AL and the DEL are listed in table 2-3.

Table 2-3: Primitives of the DEL Interface

Primitive	Request	Indication	Confirmation
DEL-SLE-PDU	X	X	

2.6.2.2 DEL-SLE-PDU

The primitive is used to pass SLE operation invocations and returns between the AL and the DEL. Its parameters are shown in table 2-4.

Table 2-4: Parameters of the Primitive DEL-SLE-PDU

Parameter	Request	Indication
SLE PDU	X	X
decoding-result		X

Request The request is used by the AL to forward PDUs received from higher layers to the DEL. The DEL encodes the PDU and forwards the encoded PDU to the TML using TML-SLE-PDU-request.

Indication The indication is issued by the DEL when a PDU received from the TML via TML-SLE-PDU-indication has been decoded or when decoding has failed. The result of the decoding process is reported in the parameter decoding-result.

2.6.3 TML DATA TRANSFER INTERFACE

2.6.3.1 Overview

The primitives defined for the interface between the DEL and the TML are listed in table 2-5

Table 2-5: Primitives of the TML Data Transfer Interface

Primitive	Request	Indication	Confirmation
TML-SLE-PDU	X	X	

2.6.3.2 TML-SLE-PDU

The primitive is used to pass encoded SLE PDUs between the TML and the DEL. The parameters of the primitive are shown in table 2-6.

Table 2-6: Parameters of the Primitive TML-SLE-PDU

Parameter	Request	Indication
Encoded PDU	X	X

Request The request is issued by the DEL when receiving DEL-SLE-PDU-request. The TML adds the TML message header and transmits the PDU as a TML SLE PDU message.

Indication The indication is issued by the TML when receiving a TML SLE PDU message.

2.6.4 TML ASSOCIATION CONTROL INTERFACE

2.6.4.1 Overview

The primitives defined for the interface between the TML and the higher layers for establishment and release of associations are listed in table 2-7.

Table 2-7: Primitives of the TML Association Control Interface

Primitive	Request	Indication	Confirmation
TML-CONNECT	X	X	X
TML-DISCONNECT	X		
TML-PEER-ABORT	X	X	
TML-PROTOCOL-ABORT		X	
TML-RESET	X		

2.6.4.2 TML-CONNECT

The primitive is used to request establishment of a TCP connection, report an incoming connection request, or confirm successful connection establishment. The parameters of the primitive are shown in table 2-8.

Table 2-8: Parameters of the Primitive TML-CONNECT

Parameter	Request	Indication	Confirmation
Responder Port Identifier	X		

Request The request is issued by the higher layers on the initiating side to request a connection to the peer application. The TML attempts to establish a connection to the destination defined by the responder port identifier, as described in 2.5.4 and 2.5.7.

Indication The indication is issued by a responding TML when a new TCP connection has been accepted and the initial TML context message has been received and accepted.

Confirmation The confirmation is issued by an initiating TML when establishment of a TCP connection succeeds.

2.6.4.3 TML-DISCONNECT

The primitive is issued by the higher layers to request that a TCP connection be disconnected. An initiating TML initializes the TCP connection release procedure as described in 2.5.4. A responding TML prepares for connection release as described in 2.5.4.2. The primitive does not have any parameters.

2.6.4.4 TML-PEER-ABORT

The primitive is used to request abortion of the association via PEER-ABORT or to report that the association has been aborted. Its parameters are listed in table 2-9.

Table 2-9: Parameters of the Primitive TML-PEER-ABORT

Parameter	Request	Indication
Diagnostic (one octet)	X	X
Originator (Local TML or Peer)		X

Request The primitive is used by the higher layers to request that the association be aborted using PEER-ABORT. The TML performs the peer abort procedure as described in 2.5.5.

Indication The indication is issued by the TML when it receives urgent data containing diagnostics in the range reserved by the Recommended Standards for SLE transfer services.

2.6.4.5 TML-PROTOCOL-ABORT

The primitive is issued by the TML if the connection fails and an error is reported by TCP or if the connection has been aborted due to an error reported on the level of the TML.

Table 2-10: Parameters of the Primitive TML-PROTOCOL-ABORT

Parameter	Indication
Diagnostics	X

2.6.4.6 TML-RESET

The primitive is issued by the higher layers to request abortion of the TCP connection without transmitting diagnostics to the peer. It does not carry any parameter. The TML terminates the connection using TCP-ABORT.

2.6.5 TML LISTENER INTERFACE

2.6.5.1 Overview

The TML listener interface supports the two primitives displayed in table 2-11.

Table 2-11: Primitives of the TML Listener Interface

Primitive	Request
TML-START-LISTEN	X
TML-STOP-LISTEN	X

2.6.5.2 TML-START-LISTEN

The primitive is used by the SPL to request the TML to accept TCP connections on the socket associated with the responder port parameter passed as argument. For this interface the responder port identifier must map to a single socket and the IP address must be valid for the local host.

Table 2-12: Parameters of the Primitive TML-START-LISTEN

Parameter	Request
Responder Port Identifier	X

2.6.5.3 TML-STOP-LISTEN

The primitive is used by the SPL to instruct the TML that no more connections should be accepted on the socket associated with the responder port parameter passed as argument.

Table 2-13: Parameters of the Primitive TML-STOP-LISTEN

Parameter	Request
Responder Port Identifier	X

2.6.6 TCP INTERFACE

2.6.6.1 Introduction

The TCP specification in reference [7] does not define an application program interface. It only provides a functional description of user commands that all TCP implementations must provide. In practice, two different APIs have evolved:

a) The 'socket interface' was originally developed for BSD UNIX. It is the most popular interface and is available on most platforms. Unfortunately there is no commonly agreed specification of the socket interface and implementations vary in a number of subtle details. POSIX has developed a draft standard P3001.1g for the socket interface (reference [E11]), but this standard has not yet been widely supported at the time of writing.

b) The Transport Layer Interface (TLI) was developed for UNIX System V as a general-purpose transport interface supporting Internet and OSI protocols. X/Open has adopted and extended TLI as the X/Open Transport Interface (XTI). The POSIX standard also includes XTI.

This Recommended Standard does not assume use of any specific TCP API. In order to specify how TCP shall be used, it defines a set of abstract service primitives based on the functional description of user commands in reference [7]. This section first provides a summary discussion of these user commands and then a definition of the service primitives used for the specification.

2.6.6.2 TCP User Commands Defined in RFC 793

Reference [7] defines the following user commands for TCP:

a) OPEN to establish a new connection or listen for incoming connect requests;

b) SEND to send data;

c) RECEIVE to read data that have been received by the TCP;

d) CLOSE to terminate a TCP connection;

e) ABORT to abort a TCP connection.

In addition, reference [7] requires that TCP provide 'TCP-to-User messages' to inform the user of certain events. These messages are not further detailed.

2.6.6.2.1 OPEN

Input: local port number
foreign socket (IP address and port number)
active/passive flag
timeout - optional

Output: local connection name

Reference [7] additionally defines arguments related to security and precedence. These have been omitted, as they are not generally supported by TCP APIs.

If the active/passive flag is set to 'active', the TCP attempts to establish a connection to the foreign socket (active open). If the flag is set to 'passive', the TCP accepts incoming connection requests (passive open). According to reference [7], a user can only specify the local port number, but TCP APIs allow specification of the IP address as well (the IP address must be a valid address on the host). However, TCP APIs generally do not allow passive open to specify the foreign socket. Therefore, this specification does not consider that option. Whether the timeout option is supported depends on the specific API.

2.6.6.2.2 SEND

Input: local connection name
buffer address
byte count
PUSH flag
URGENT flag
timeout - optional

Output: none

The SEND command copies the number of bytes defined by 'byte count' from 'buffer address' to the TCP transmit buffer. The description in reference [7] refers to a blocking interface where the command would return only when all data have been copied. APIs supporting a non-blocking interface return the number of bytes that have been copied. To send the remaining data the command must be reissued when the TCP reports it has sufficient room in its transmit buffer.

The PUSH flag instructs the TCP actually to transmit the data as soon as possible. TCP APIs generally do not support specification of the flag by the user. Therefore, it is not further considered in this Recommended Standard.

When the URGENT flag is set, TCP sets the urgent flag in the next segment transmitted indicating to the receiver that urgent data are present. In addition it sets an urgent pointer in the segment header that points to the end of the urgent data.

Whether the timeout option is supported depends on the specific API.

2.6.6.2.3 RECEIVE

Input: local connection name
buffer address
byte count

Output: byte count
PUSH flag
URGENT flag

The RECEIVE command copies a maximum of 'byte count' bytes from the TCP receive-buffer to the 'buffer address' and returns the actual number of bytes copied. If present, it also returns the PUSH flag and the URGENT flag.

2.6.6.2.4 CLOSE

Input: local connection name

Output: none

The CLOSE command terminates transmission of data on a connection and signals to the peer TCP that the connection is closing. Data can still be received on this connection. This is called a 'half-close' by reference [7]. The TCP connection terminates when both users have closed the connection.

NOTE – The CLOSE command must not be confused with the close function of the socket interface, which actually performs a 'full close'.

2.6.6.2.5 ABORT

Input: local connection name

Output: none

When ABORT is issued, the TCP sends a special segment with the flag RST set (RESET segment) and immediately releases all resources allocated to the connection. Reception of a RESET segment causes a hard TCP error.

2.6.6.3 Service Primitives Used in this Recommended Standard

2.6.6.3.1 Overview

This Recommended Standard uses the abstract primitives listed in table 2-14 to describe how the TCP interface is used. These primitives are defined in terms of the TCP user commands in reference [7] as far as possible. Additional specifications are added where necessary.

Table 2-14: Primitives Used for the TCP-Interface

Primitive	Request	Indication	Confirmation
TCP-PASSIVE-OPEN	X		
TCP-CONNECT	X	X	X
TCP-DISCONNECT	X	X	
TCP-DATA	X	X	
TCP-URGENT-DATA	X	X	
TCP-ABORT	X	X	

2.6.6.3.2 TCP-PASSIVE-OPEN

The TCP-PASSIVE-OPEN primitive is used to request TCP to accept connections on a specified local port. The parameters are listed in table 2-15.

Table 2-15: Parameters of the Primitive TCP-PASSIVE-OPEN

Parameter	Request
Local IP address	X
Local port number	X

Request Call to the command OPEN with the active/passive flag set to 'passive'. The local IP address can remain unspecified. In this case, TCP will accept a connection arriving on any interface of the host.

2.6.6.3.3 TCP-CONNECT

The TCP-CONNECT primitive request is used to request establishment of a TCP connection to a specified socket. The indication refers to an indication of TCP that a connection has been accepted.

Table 2-16: Parameters of the Primitive TCP-CONNECT

Parameter	Request	Indication	Confirmation
Foreign IP address	X	X	
Foreign port number	X	X	
Local IP address	X		
Local port number	X		

Request Request of the OPEN command with the active/passive flag set to 'active'. The foreign IP address and port number are specified. The local port number remains unspecified such that TCP can select an ephemeral port. It is recommended to leave the local IP address and port number unspecified, but supply of these parameters is not excluded.

Indication TCP-to-User message, indicating that a new connection has been accepted by the TCP on a socket for which TCP-PASSIVE-OPEN-request had been issued. TCP APIs generally require the application program to accept the new connection before data can be exchanged.

Confirmation Indication by the TCP that a non-blocking connect request succeeded.

2.6.6.3.4 TCP-DISCONNECT

The TCP-DISCONNECT service request is used to request a half-close of the connection. The indication refers to an indication by TCP that the peer has closed its side of the connection.

Request Request of the command CLOSE.

Indication TCP-to-User message, indicating that the peer has closed its side of the connection. (In the socket API this is indicated by returning a zero byte count from a request to read data from the TCP receive buffer.)

As mentioned earlier in 2.6.6.2, the TCP user command CLOSE is supposed to perform a 'half close', which indicates that no more data will be transmitted but still allows receiving data. Standard interfaces provided by TCP APIs to disconnect a TCP connection terminate both transmission and reception of data and perform the TCP connection release procedure autonomously without user interaction. This Recommended Standard has been specifically designed to allow use of such interfaces as well as interfaces that support the original concepts expressed in reference [7] to implement the primitive TCP-DISCONNECT.

2.6.6.3.5 TCP-DATA

The primitive is used to describe data transfer across a TCP connection.

Table 2-17: Parameters of the Primitive TCP-DATA

Parameter	Request	Indication
Data buffer	X	X

Request Call of the command SEND with the URGENT flag not set, repeating the call if necessary to send a complete message.

Indication TCP-to-User message, indicating that data are present in the TCP receive buffer and subsequent call of the command RECEIVE. For the purpose of this Recommended Standard it is assumed that RECEIVE is called once to read the TML message header followed by one or more calls to read the message body.

2.6.6.3.6 TCP-URGENT-DATA

The primitive is used to describe urgent data transfer across a TCP connection.

Request Call of the command SEND with the URGENT flag set.

Indication TCP-to-User message, indicating that urgent data are present.

2.6.6.3.7 TCP-ABORT

The service request is used abort a connection. The indication refers to a TCP error report indicating that the connection can no longer be used.

Request Call of the command ABORT.

Indication TCP-to-User message, indicating a hard TCP error.

This Recommended Standard recognizes that some implementations of specific TCP APIs do not support the user command ABORT or do not allow to issue this command in certain states, e.g., after closing of the connection has been requested. In such cases, the primitive TCP-ABORT should be implemented using the most efficient method for termination of the connection that is supported by the TCP API. Such alternative methods might invoke the standard orderly connection release procedure and not transmit a RESET segment, which is considered a conforming behavior.

3 ISP1 MESSAGES AND PROCEDURES

3.1 AUTHENTICATION LAYER

3.1.1 Authentication shall be based on the following parameters:

a) User name. The following rules shall apply for this parameter:

1) the user name shall be a string of 3 to 16 characters;

2) the user name shall be identical to the authority identifier of the application by which the application is identified in the BIND invocation and the BIND return.

NOTE – For an initiating SLE application, the user name is the initiator-identifier and for a responding SLE application the user name is the responder-identifier.

b) Password. The password shall be an octet string of 6 to 16 octets.

3.1.2 When the Authentication Layer receives an SLE PDU requiring authentication from the higher layers it shall generate the credentials and insert them into the PDU passed to the Data Encoding Layer.

NOTE – Whether a PDU requires authentication is determined by the authentication level agreed upon for a service instance, it can be

– 'all' all PDUs except PEER-ABORT-invocation shall be authenticated;

– 'bind' only the BIND invocation and return shall be authenticated;

– 'none' no PDU shall be authenticated.

3.1.2.1 Generation of credentials shall be performed according to the protected simple authentication procedure (Protected 1) defined in reference [11] and detailed by the following specifications.

3.1.2.1.1 The following information shall be encoded using the ASN.1 syntax defined in reference [9] and the Distinguished Encoding Rules (DER) specified in reference [10]:

a) the current time, using the CCSDS day segmented time code without the P-field;

b) a random number;

c) the user name of the local SLE application;

d) the password of the local SLE application.

NOTE – Encoding the information with DER provides a platform-independent bit pattern from which a hash code can be generated. Use of ASN.1 and DER for generation of credentials does not imply that ASN.1 or DER is used for encoding of data exchanged between the service user and the service provider.

3.1.2.1.2 The ASN.1 type used for encoding shall be the one defined in figure 3-1.

```
HashInput ::= SEQUENCE
{ time          OCTET STRING (SIZE(8))
, randomNumber  INTEGER
, userName      VisibleString
, passWord      OCTET STRING
}
```

Figure 3-1: ASN.1 Type for Generation of 'the Protected'

3.1.2.1.3 The output of the encoder shall be passed through a one-way hash function to obtain a message digest, also referred to as 'the protected'.

3.1.2.1.4 The following information shall be inserted into the parameter credentials of the PDU before passing it to the Data Encoding Layer:

a) the time parameter used for generation of the message digest;

b) the random number used for generation of the message digest;

c) the message digest itself.

NOTE – The user name used for authentication is transmitted as part of the BIND invocation or return. Therefore it is not necessary to include it into the credentials parameter.

3.1.2.2 Authentication of credentials received from the peer application shall be performed according to the protected simple authentication procedure (Protected 1) defined in reference [11] and detailed by the following specifications.

3.1.2.2.1 The time in the credentials shall be checked against the current time. If the time difference is larger than acceptable, authentication shall fail.

NOTE – This Recommended Standard assumes that the acceptable delay is provided to the Authentication Layer as a configuration parameter.

3.1.2.2.2 The following information shall be encoded using the ASN.1 type defined in figure 3-1 and the Distinguished Encoding Rules:

a) the time obtained from the credentials, in the CCSDS day segment time format without P-Field;

b) the random number obtained from the credentials;

c) the user name of the peer application;

d) the password of the peer application.

3.1.2.2.3 The encoded data shall be passed through a one-way hash function to obtain a message digest.

3.1.2.2.4 The message digest shall be compared with the message digest in the credentials. If these match, authentication shall be regarded as successful. Otherwise, authentication shall fail.

3.1.2.3 The one-way hash function used shall be SHA-1 defined by reference [12].

3.2 DATA ENCODING LAYER

3.2.1 The DEL shall encode and decode SLE PDUs using the ASN.1 types defined in the Recommended Standard for the applicable transfer service type and version number and the Basic Encoding Rules defined in reference [10].

NOTE – The service type and version number can be extracted from the BIND invocation PDU, which must be the first PDU passed to the DEL after establishment of an ISP1 association.

3.2.2 If decoding of a PDU fails, the DEL shall inform the higher layers, which shall apply the procedures foreseen by the Recommended Standards for SLE transfer services.

3.2.3 The DEL shall encode the credentials parameter defined for SLE PDUs using the ASN.1 type shown in figure 3-2. It shall encode the resulting bytes as an OCTET STRING according to the definitions in the Recommended Standards for SLE transfer services.

```
ISP1Credentials ::= SEQUENCE
{  time           OCTET STRING (SIZE (8)) -- CCSDS CDS time code
,  randomNumber   INTEGER (0 .. 4294967295)
,  theProtected   OCTET STRING (SIZE (20))
}
```

Figure 3-2 : ASN.1 Type for the Credentials Parameter

NOTE – For the purpose of authentication, the constituents of the credentials must be distinguishable for the receiver. This implies the need for a refined ASN.1 type.

3.3 TRANSPORT MAPPING LAYER

3.3.1 TECHNOLOGY

The TML shall use the Transmission Control Protocol (TCP) specified by reference [7] for communication with a remote TML. It shall map one SLE association to exactly one TCP connection.

3.3.2 TML MESSAGES

3.3.2.1 The TML shall use three types of messages for exchange of data over an established TCP connection:

a) an SLE PDU message for transfer of SLE PDUs;

b) a TML context message to transmit TML initialization parameters;

c) a TML heartbeat message to periodically probe an idle TCP connection.

3.3.2.2 TML messages shall be formatted as defined by the specifications in 3.3.2.2.1 to 3.3.2.2.6.

NOTES

1 The layout of a TML message is shown in figure 3-3.

2 The message body contains any number of octets, not necessarily a multiple of four.

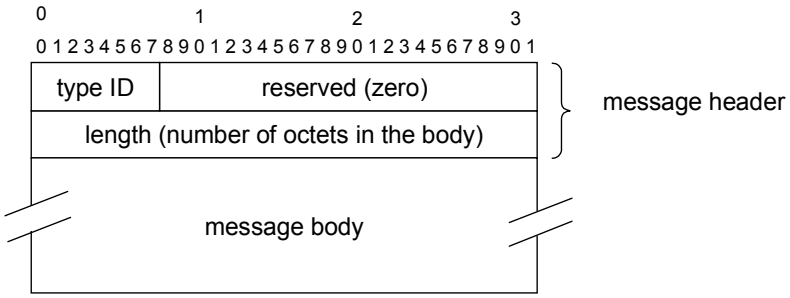

Figure 3-3: Layout of a TML Message

3.3.2.2.1 A TML message consists of an eight-octet message header and a message body.

3.3.2.2.2 The message header consists of the following fields in the sequence indicated:

a) a message type field of one octet containing the message type identifier as defined by table 3-1;

Table 3-1: TML-Message Type Identifiers

Type ID	Message type
1	SLE PDU Message
2	TML Context Message
3	TML Heartbeat Message

b) a reserved field of 3 octets set to 'zero';

c) a length field of four octets, which holds the number of octets in the message body represented as a binary unsigned integer value.

3.3.2.2.3 The body of an SLE PDU Message consists of the encoded SLE PDU.

3.3.2.2.4 The body of a TML context message consists of 12 octets and contains the following fields in the sequence indicated:

a) the protocol identification field (four octets) containing the characters 'I' 'S' 'P' '1' in ASCII encoding (hexadecimal 49 53 50 31);

b) a reserved field of 3 octets, set to 'zero';

c) the protocol version (1 octet) set to the binary value 1;

d) the heartbeat-interval (2 octets) represented as a binary unsigned integer value;

e) the dead-factor (2 octets) represented as a binary unsigned integer value.

NOTE – The parameters heartbeat interval and dead-factor are defined in 3.3.3. The layout of the TML context message is shown in figure 3-4.

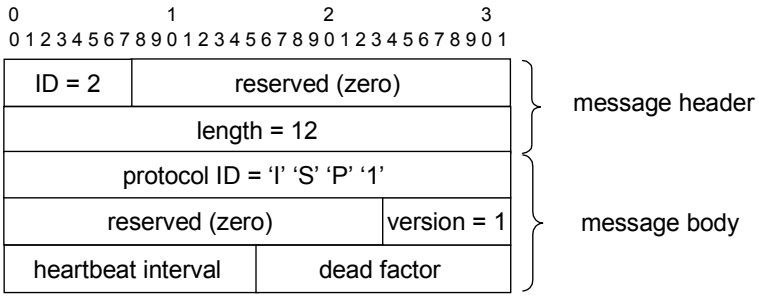

Figure 3-4: Layout of a TML Context Message

3.3.2.2.5 The body of a TML heartbeat message is empty.

NOTE – A TML heartbeat message consists of a message header only, where the length field is set to 'zero'.

3.3.2.2.6 All integer values in TML messages shall be in network octet order ('big endian').

NOTE – Because the interface to TCP is defined in terms of bytes, the TML must ensure that the byte order of integer values is correct. The order in which the bits of individual bytes are transmitted depends on lower level protocols and hardware interfaces and cannot be influenced by the application software. If there is a need to reverse the bit order within a byte, it is usually performed either by the hardware or low-level device drivers.

3.3.3 HEARTBEAT MECHANISM

NOTE – The TML uses the procedure specified in this section to detect failure of a TCP connection during periods in which no messages are transmitted. This procedure is described first, because the parameters and timers defined by the procedure are frequently referred to by the specification of connection establishment and release.

3.3.3.1 Detection of failed connections shall be based on the following parameters:

a) the heartbeat-interval specifies the maximum period, measured in seconds, in which a TML might not transmit any message;

b) the dead-factor defines the time, measured in units of heartbeat-intervals, after which the TML shall assume that a connection has failed, if it does not receive any message from the peer TML;

3.3.3.2 If the heartbeat-interval is set to zero the procedure defined in this section shall not be applied.

3.3.3.3 The TML shall use a heartbeat transmit-timer (HBT timer), set to the value of the heartbeat-interval according to the following specifications:

3.3.3.3.1 Whenever the TML transmits a message, it shall restart the HBT timer.

3.3.3.3.2 When the HBT timer expires, the TML shall transmit a TML heartbeat message and restart the HBT timer.

3.3.3.4 The TML shall use a heartbeat receive-timer (HBR timer), set to the value of the heartbeat-interval multiplied by the value of the dead-factor, according to the following specifications:

3.3.3.4.1 Whenever the TML receives a message, it shall restart the HBR timer.

3.3.3.4.2 When the HBR timer expires, the TML shall assume that the connection has failed. It shall issue TCP-ABORT and inform the higher layers by TML-PROTOCOL-ABORT-indication with the TML diagnostic set to 'heartbeat timeout'.

3.3.4 CONNECTION ESTABLISHMENT

3.3.4.1 Initiator Side

NOTE – This section specifies the connection establishment procedure used by the TML when a logical port identifier maps to a single TCP socket. The procedure for connecting to multiple hosts concurrently is defined in 3.3.8.

3.3.4.1.1 When receiving TML-CONNECT-request, the TML shall establish a TCP connection as defined by the specifications in 3.3.4.1.1.1 to 3.3.4.1.1.3.

3.3.4.1.1.1 The TML shall issue TCP-CONNECT-request using the IP address and port number derived from the logical port identifier to specify the foreign IP address and port number.

NOTE – TCP APIs generally allow the local address and port number to remain unspecified. They will then automatically use an IP address by which the socket can be reached and assign an ephemeral port number.

3.3.4.1.1.2 If TCP-CONNECT-request succeeds, the TML shall:

a) start the HBT timer and the HBR timer; and

b) issue TML-CONNECT-confirmation.

NOTE – The TML context message is defined in 3.3.2. The heartbeat mechanism is defined in 3.3.3. If the heartbeat interval is zero, the heartbeat timers are not started.

3.3.4.1.1.3 If TCP-CONNECT-request fails, or any other error occurs, the TML shall issue TML-PROTOCOL-ABORT-indication with the appropriate diagnostics.

NOTE – This Recommended Standard assumes that the TCP implementation takes care of temporary network problems and re-transmissions and that the TCP implementation can be configured as needed. Therefore, failure of TCP-CONNECT request is assumed to be a hard error. This Recommended Standard does not exclude that an implementation retries connection establishment for a configurable period or until it receives TML-RESET-request because the return timer expires.

3.3.4.2 Responder Side

3.3.4.2.1.1 When receiving TML-START-LISTEN-request, the TML shall request TCP-PASSIVE-OPEN using the IP address and port number derived from the logical responder port identifier to specify the local address and port number.

3.3.4.2.2 When receiving TML-STOP-LISTEN-request, the TML shall instruct TCP to stop accepting connection requests for the IP address and port number identified by the argument 'responder port identifier'.

NOTE – This effect can generally be achieved by closing the listen socket (Socket API) or transport endpoint (XTI/TLI).

3.3.4.2.3 When receiving TCP-CONNECT-indication, the TML shall create a new association in the state 'TML starting' and perform the following procedure:

3.3.4.2.3.1 It shall start a TML start-up timer (TMS timer).

3.3.4.2.3.2 If the TML receives at least the amount of data specified for a TML message header before the TMS timer expires, it shall read the first 8 octets of the data and perform the following checks:

3.3.4.2.3.3 If the first 8 octets of the data do not present a valid TML message header, the TML shall stop the TMS timer and abort the connection using TCP-ABORT.

NOTE – As explained in 2.6.6.3.7, this specification allows TCP-ABORT to have the same effect as TCP-DISCONNECT-request, if the TCP user command ABORT is not supported by the TCP API.

3.3.4.2.3.4 If the message received is a TML context message, the TML shall retrieve the complete message and perform the checks in 3.3.4.2.3.4.1 to 3.3.4.2.3.4.3:

NOTE – The TML context message is defined in 3.3.2.

3.3.4.2.3.4.1 If the protocol identifier is not correct, the TML shall stop the TMS timer and abort the connection using TCP-ABORT.

3.3.4.2.3.4.2 If the version number does not match the version of this specification supported by the TML, the TML shall stop the TMS timer and abort the connection using TCP-ABORT.

3.3.4.2.3.4.3 If the heartbeat interval and dead-factor values are acceptable, the TML shall start the HBT timer.

NOTES

1 The heartbeat interval and dead-factor are explained in 3.3.3.

2 If the heartbeat interval is zero, the heartbeat transmit timer shall not be not started.

3.3.4.2.3.5 When all checks have been passed, the TML shall change the state of the association to 'data transfer' and issue TML-CONNECT-indication.

3.3.4.2.3.6 If the first message received is a valid TML message other than the TML context message, the TML shall stop the TMS timer and abort the connection using TCP-ABORT.

3.3.4.2.3.7 When receiving the first valid SLE PDU message in the state 'data transfer', the TML shall stop the TMS timer and start the HBR timer.

NOTE – If the heartbeat interval is zero, the heartbeat receive timer shall not be started.

3.3.4.2.3.8 If the TMS timer expires before a valid context message and the first valid SLE PDU message arrive, the TML shall terminate the connection using TCP-ABORT.

3.3.4.2.3.9 If the TMS timer expires in the state 'data transfer', i.e., after reception of the context message, but before reception of a valid SLE PDU message, the TML shall additionally inform the higher layers using TML-PROTOCOL-ABORT indication.

3.3.5 ORDERLY CONNECTION RELEASE

3.3.5.1 When receiving TML-DISCONNECT-request, an initiating TML shall close the TCP connection.

NOTE – If the peer TML does not close its side of the connection, actual release of resources allocated to the TCP socket depends on the policy applied by the TCP implementation. Implementations can ensure release of resources using the following procedure if that is supported by the TCP API and the TCP implementation:

a) perform a half-close for the sending side and start a timer;

b) if any data arrive, abort the connection using TCP-ABORT;

c) if the peer does not close the connection within the timeout, abort the connection using TCP-ABORT.

3.3.5.2 When receiving TML-DISCONNECT-request, a responding TML shall perform the steps described in 3.3.5.2.1 to 3.3.5.2.4:

3.3.5.2.1 The TML shall stop the HBT timer, restart the HBR timer, and wait for TCP-DISCONNECT-indication.

NOTE – If the heartbeat interval is zero, the heartbeat mechanism shall be disabled. In this case, no timers shall be used for the connection release procedure.

3.3.5.2.2 The TML shall abort the TCP connection using TCP-ABORT-request if it receives further data.

NOTE – No data can arrive if both sides correctly implement the SLE operations as defined by the applicable Recommended Standard.

3.3.5.2.3 When receiving TCP-DISCONNECT-indication, the TML shall close its side of the connection using TCP-DISCONNECT-request.

3.3.5.2.4 If the HBR timer expires before the TML receives TCP-DISCONNECT-indication, it shall abort the connection with TCP-ABORT-request.

3.3.6 ABORTING CONNECTIONS

3.3.6.1 Peer Abort

3.3.6.1.1 When receiving TML-PEER-ABORT-request, the TML shall perform the procedure specified by 3.3.6.1.3.

3.3.6.1.2 Except when explicitly defined otherwise in this Recommended Standard, the TML shall use the peer abort procedure defined by 3.3.6.1.3 to report errors to the peer TML, using one of the TML diagnostic codes defined in annex A. In this case, it shall issue TML-PROTOCOL-ABORT-indication with the appropriate diagnostic to inform the higher layers that it has aborted the connection.

3.3.6.1.3 The TML shall implement the PEER-ABORT operation defined by the Recommended Standards for SLE transfer services as defined by the specifications in 3.3.6.1.3.1 to 3.3.6.1.3.3:

3.3.6.1.3.1 The aborting TML shall perform the following steps:

a) the TML shall send the PEER-ABORT diagnostics as one byte of urgent data using TCP-URGENT-DATA-request;

b) the TML shall stop the HBT and HBR timers, and start the CPA timer;

c) the TML shall wait for TCP-DISCONNECT-indication;

d) the TML shall silently discard all data that arrive after it has sent the urgent data;

e) when receiving TCP-DISCONNECT-indication, the TML shall close its side of the connection using TCP-DISCONNECT-request;

f) if the CPA timer expires before the TML receives TCP-DISCONNECT-indication, the TML shall abort the connection with TCP-ABORT-request.

NOTE – As explained in 2.6.6.3.7, this specification allows TCP-ABORT to have the same effect as TCP-DISCONNECT-request, if the TCP user command ABORT is not supported by the TCP API.

3.3.6.1.3.2 When the TML is notified by TCP-URGENT-DATA-indication that urgent data are pending, it shall perform the following steps:

a) the TML shall read and discard all data from the TCP receive buffer up to but excluding the byte referred to by the TCP urgent pointer;

b) the TML shall read and analyze the byte referenced by the TCP urgent pointer;

c) if the diagnostic code is in the range reserved by the Recommended Standards for SLE transfer services, the TML shall issue the primitive TML-PEER-ABORT-indication passing the diagnostic as a parameter;

d) otherwise the TML shall issue TML-PROTOCOL-ABORT-indication primitive with the diagnostics retrieved from the urgent byte;

e) the TML shall close the TCP connection.

NOTES

1 When the receiving TML has been notified of the presence of urgent data, that TML might receive requests to transfer data from the DEL. It is expected that such requests will either be rejected with an indication that the connection is aborting or will be silently ignored. Requests to disconnect or abort the connection must be accepted, but should be ignored. The only exception is a TML-RESET request, which should cause an immediate TCP-ABORT.

2 If the peer TML does not close its side of the connection, actual release of resources allocated to the TCP socket depends on the policy applied by the TCP implementation. Implementations can ensure release of resources using the following procedure if that is supported by the TCP API and the TCP implementation:

   a) perform a half-close for the sending side and start a timer;

   b) if any data arrive, abort the connection using TCP-ABORT;

   c) if the peer does not close the connection within the timeout, abort the connection using TCP-ABORT.

3.3.6.1.3.3 If the aborting TML is notified by TCP-URGENT-DATA-indication that urgent data are pending after transmission of the PEER-ABORT diagnostics, it shall read and discard all data up to and including the byte referred to by the TCP urgent pointer and then close the TCP connection.

NOTE – This situation can occur if both sides request PEER-ABORT nearly simultaneously. Both sides must close the TCP connection in order to avoid a deadlock situation where each side is waiting for the other side to close the connection. In this situation, neither side will be transmitting data following the PEER-ABORT diagnostic, such that a 'full close' can be issued without invoking TCP ABORT. Because the PEER-ABORT procedure was already invoked locally the PEER-ABORT diagnostic received will not be passed to the higher layers.

3.3.6.2 Silent Abort

When receiving TML-RESET-request, the TML shall abort the connection with TCP-ABORT-request.

NOTE – As explained in 2.6.6.3.7, this Recommended Standard allows TCP-ABORT to have the same effect as TCP-DISCONNECT-request, if the TCP user command ABORT is not supported by the TCP API.

3.3.6.3 Protocol Abort

3.3.6.3.1 The TML shall issue TML-PROTOCOL-ABORT-indication in the following cases

a) TCP reports an unrecoverable error;

b) TCP reports a send timeout;

c) TCP reports that the peer has closed its side of the connection by TCP-DISCONNECT-indication before TML-DISCONNECT-request or TCP-URGENT-DATA-indication has been received.

NOTE – As indicated in 3.3.6.1 the TML additionally issues TML-PROTOCOL-ABORT-indication when receiving peer abort with a diagnostic code in the range defined by annex A.

3.3.7 SLE PDU TRANSFER

3.3.7.1 When receiving TML-SLE-PDU-request, the TML shall add the message header for the SLE-PDU message and transmit the message using TCP-DATA-request.

3.3.7.2 When receiving TCP-DATA-indication, the TML shall check the first eight octets of the data and proceed as defined in 3.3.7.2.1 to 3.3.7.2.4.

NOTE – The very first message received on a TCP connection by a responding TML shall be handled differently, as specified in 3.3.4.2.

3.3.7.2.1 If the first eight octets of the data do not present a valid TML message header, the TML shall abort the connection with the peer abort procedure and the diagnostics 'badly formatted TML message'.

3.3.7.2.2 If the message is a TML heartbeat message, the TML shall restart the HBR timer.

3.3.7.2.3 If the message is an SLE PDU message, the TML shall retrieve the message body from the TCP, forward it using the primitive TML-SLE-PDU-indication, and restart the HBR timer.

3.3.7.2.4 If the message is a TML context message, the TML shall abort the connection with the peer abort procedure and the diagnostics 'TML protocol error'.

3.3.8 CONNECTION ESTABLISHMENT TO REDUNDANT HOSTS

3.3.8.1 If an initiating TML receives TML-CONNECT-request with a port identifier that maps to a set of sockets, it shall perform the modified connection establishment procedure defined in 3.3.8.1.1 to 3.3.8.1.3.

3.3.8.1.1 The TML shall call TCP-CONNECT-request for each of the sockets in the set.

3.3.8.1.2 When the TCP connection establishment succeeds for one of the sockets, the TML shall immediately call TCP-ABORT on all other sockets in the set. It shall then proceed with the connection establishment as specified by 3.3.4.1.1.2, using the socket for which TCP connection establishment succeeded.

3.3.8.1.3 When TCP connection fails for all sockets in the set, the TML shall issue TML-PROTOCOL-ABORT-indication with the appropriate diagnostics.

4 TML STATE TABLE

4.1 INTRODUCTION

This section defines state tables for the Transport Mapping Layer. The state tables for the Data Encoding Layer and the Authentication Layer are trivial and are therefore not detailed in this document.

The state table reflects the baseline requirements specified in section 3 and does not cover implementation considerations discussed in additional comments on these requirements. It specifies connection establishment with a single provider and does not cover the procedure for concurrent connection establishment with redundant provider systems described in 3.3.8.

4.2 NOTATION

NOTE – The notation used for the state tables is the one specified by UML for state diagrams (see reference [E12]). This notation has been slightly extended to adapt it to state tables. It is summarized below in EBNF notation together with the extensions. Extensions are highlighted by underlining. For formulation of conditions, the Object Constraint Language (OCL) specified by UML is used.

An incoming event in the event column is defined by

<origin> ':' <event-name> ['('<arguments>')']

Processing of the event is described by the following sequence

[<guard-condition>] [<action-expression>]* [<send-clause>]* [<state-transition>]
<guard-condition> : := '[' <condition> ']'
<condition> ::= conditional expression formulated in OCL
<action-expression> ::= '/' <action-name> ['(' <arguments> ')']
<send-clause> ::= '^' <target> '.' <event-name> ['(' <arguments> ')']
<state-transition> ::= '→' <new-state>

Transition to self is not shown in the tables.

Actions can be simple actions or compound actions. Compound actions are displayed in capital letters and are expanded using simple pseudo-code (IF THEN ELSE END IF) together with the notational elements shown above.

For events passed between the layers defined in this Recommended Standard, the names of the service primitives as defined in 2.6 are used. Because of the limited space, the use of a primitive is abbreviated as shown below:

- request req
- response rsp
- indication ind

– confirmation cnf

State / event combinations that cannot occur when the TML is correctly implemented and the higher layers correctly implement the SLE transfer service are marked by 'N/A'.

4.3 STATES

The state table uses the following states for an ISP1 association:

S0	CLOSED	The initial and final state, in which no TCP connection exists.
S1	TML STARTING	Initialization state for TML; depending on the role (initiator or responder) the TML establishes the TCP connection, or waits for the initial message on an accepted TCP connection.
S2	DATA TRANSFER	SLE PDUs can be exchanged.
S3	PEER ABORTING	The TML has received an indication that urgent data are present and waits for the PEER-ABORT diagnostic to arrive.
S4	TML CLOSING	The TML releases the TCP connection in response to a PEER-ABORT or a DISCONNECT request from the higher layers.

4.4 EVENTS

4.4.1 EVENTS RECEIVED FROM THE HIGHER LAYERS

The events received from the higher layers are the service primitives defined in 2.6.4 and 2.6.5, namely:

– CONNECT-request
– DISCONNECT-request
– PEER-ABORT-request
– RESET-request

4.4.2 EVENTS SENT TO THE HIGHER LAYERS

The events sent to the higher layers are the service primitives defined in 2.6.4 and 2.6.5, namely:

– CONNECT-indication
– CONNECT-confirmation
– PEER-ABORT-indication
– PROTOCOL-ABORT-indication

4.4.3 EVENTS RECEIVED FROM THE DATA ENCODING LAYER

The event received from the DEL is the service primitive defined in 2.6.3, namely

– SLE-PDU-request

4.4.4 EVENTS SENT TO THE DATA ENCODING LAYER

The event sent to the DEL is the service primitive defined in 2.6.3, namely

– SLE-PDU-indication

4.4.5 EVENTS RECEIVED FROM THE TCP

The events received from the TCP are the service primitives defined in 2.6.6.3, namely

– CONNECT-indication
– CONNECT-confirmation
– DISCONNECT-indication
– DATA-indication
– URGENT-DATA-indication (signal that urgent data are present in the incoming data stream)
– ABORT-indication

In addition, 'TCP Error' and 'Timeout' are considered events that are received from the TCP. 'Timeout' refers to either a connect timeout or send timeout indicated by the TCP.

4.4.6 EVENTS SENT TO THE TCP

The events sent to the TCP are the service primitives defined in 2.6.6.3, namely

– CONNECT-request
– DISCONNECT-request
– DATA-request
– URGENT-DATA-request
– ABORT-request

4.4.7 INTERNAL EVENTS

TMS Timeout	the TML Start-up timer expires
HBT Timeout	the heartbeat-transmit-timer expires
HBR Timeout	the heartbeat-receive-timer expires
CPA Timeout	the 'Close after Peer Abort' timer expires

4.5 PREDICATES

diagnostics = SLE diagnostics	The diagnostic code received for PEER-ABORT is in the range reserved by the Recommended Standards for SLE transfer servcies.
diagnostics = TML diagnostics	The diagnostic code received for PEER-ABORT is in the range defined in annex A of this Recommended Standard.
first PDU	The PDU is the first one received on the association.
heartbeat acceptable	The parameters received with the TML context message are acceptable to the receiver.
local peer abort	The PEER-ABORT procedure was requested by the local application.
msg = TML Context	The message received is a correctly formatted TML context message.
msg <> TML Context	The message received is not a TML context message.
msg = TML PDU	The message received is a PDU message.
msg = TML Heartbeat	The message received is a TML heartbeat message.
msg <> TML Message	The message received is not a valid TML message.
role = initiator	The TML is in the role of the initiator, i.e., it initializes the TCP connection.
role = responder	The TML is in the role of the responder, i.e., it accepts the TCP connection.
urgent byte available	The urgent byte was read from the TCP interface.
protocol supported	The protocol identifier is ‘ISP1’ and the version number is supported by the TML.

4.6 ACTIONS

4.6.1 SIMPLE ACTIONS

/build TML message	Add the TML message header to the encoded PDU.
/cleanup	Stop HBR timer and HBT timer.
/discard	Silently discard the data received.
/discard pending data	If unsent data exist discard them, even if part of a TML message has already been transmitted.
/discard normal data	Discard all received data except for the urgent byte.
/extract PDU	Remove the TML message header from the encoded PDU.
/reject	Reject the request received from higher layers.
/restart HBR timer	Stop the heartbeat-receive-timer and start it again.
/restart HBT timer	Stop the heartbeat-transmit-timer and start it again.
/start HBR timer	Start the heartbeat-receive-timer.
/start HBT timer	Start the heartbeat-transmit-timer.
/start TMS timer	Start the TML start-up timer.
/stop HBR timer	Stop the heartbeat-receive-timer.

/stop HBT timer	Stop the heartbeat-transmit-timer.
/stop TMS timer	Stop the TML start-up timer.

4.6.2 COMPOUND ACTIONS

/PEER ABORT(diagnostic) is defined as

^TCP.URGENT-DATA-req(diagnostic)
/discard pending data
/set local peer abort = true
/stop HBT timer
/stop HBR timer
/start CPA timer

4.7 STATE TABLE

	S0 - CLOSED	S1 - TML STARTING	S2 - DATA TRANSFER	S3 - PEER ABORTING	S4 - TML CLOSING
HL: CONNECTreq	^TCP.CONNECTreq → S1 *(N/A for the responder)*	N/A	N/A	N/A	N/A
TCP: CONNECTcnf	N/A	^TCP.DATAreq(TML-Context) /start HBR-Timer /start HBT-Timer ^HL.ConnectConf → S2 *(N/A for the responder)*	N/A	N/A	N/A
TCP: CONNECTind	/start TMS-timer /set first PDU = false → S1 *(N/A for the initiator)*	N/A	N/A	N/A	N/A

	S0 - CLOSED	S1 - TML STARTING	S2 - DATA TRANSFER	S3 - PEER ABORTING	S4 - TML CLOSING
TCP: DATAind(msg)	N/A	[role = initiator] ^TCP.ABORTreq ^HL.PROTOCOL-ABORTind → S0 [role = responder] [msg = TML Context] [not protocol supported] /stop TMS timer ^TCP.ABORTreq → S0 [not heartbeat acceptable] /stop TMS timer /PEER ABORT (heartbeat values not acceptable) → S4 /start HBT-Timer /set first PDU = true ^HL.CONNECTind → S2 [msg <> TML Context] /stop TMS timer ^TCP.ABORTreq → S0	[msg = TML-PDU] [role = responder] [not first PDU] /restart HBR timer [first PDU] /stop TMS timer /start HBR Timer /set first PDU = false [role = initiator] /restart HBR timer /extract PDU ^DEL.SLE-PDUind(PDU) [msg = TML-Heartbeat] restart HBR-Timer [msg = TML-Context] [role=responder and first PDU] /stop TMS timer ^HL.PROTOCOL-ABORTind /PEER ABORT(protocol err) → S4 [msg <> TML message] ^HL.PROTOCOL-ABORTind [role=responder and first PDU] /stop TMS timer /PEER ABORT (badly formatted message) → S4	/discard normal data [urgent byte available] [not local peer abort] [diagnostics = SLE diagnostics] ^HL.PEER-ABORTind [diagnostics = TML diagnostics] ^HL.PROTCOL-ABORTind ^TCP.DISCONNECT-req → S0	[not local peer abort] ^TCP.ABORTreq /stop HBR Timer → S0 [local peer abort] /discard
HL: DISCONNECT req	N/A	N/A	[role = initator] ^TCP.DISCONNECTreq /cleanup → S0 [role = responder] /set local peer abort = false /stop HBT-timer /restart HBR timer → S4	/reject(aborting)	N/A

	S0 - CLOSED	S1 - TML STARTING	S2 - DATA TRANSFER	S3 - PEER ABORTING	S4 - TML CLOSING
TCP: DISCONNECT ind	N/A	^TCP.DISCONNECTreq /stop TMS-Timer → S0 *(N/A for the initiator)*	^TCP.DISCONNECTreq ^HL.PROTOCOL-ABORTind /cleanup → S0	^TCP.DISCONNECTreq [not local peer abort] ^HL.PROTOCOL-ABORTind /stop HBR-Timer → S0	^TCP.DISCONNECTreq /stop HBR-Timer → S0
DEL: SLE-PDUreq	N/A	N/A	/build TML message ^TCP.DATAreq(message) /restart HBT-Timer	/reject(aborting)	N/A
HL: PEER-ABORTreq (diagnostics)	N/A	N/A	/PEER ABORT (diagnostics) → S4	/set local peer abort = true	N/A
TCP: URGENT-DATAind	N/A	^TCP.ABORTreq /stop TMS-Timer → S0 *(N/A for the initiator)*	/set local peer abort = false → S3	N/A	[not local peer abort] ^TCP.ABORTreq /stop HBR-Timer → S0 [local peer abort] →S3
HL: RESETreq	N/A	^TCP.ABORTreq → S0 *(N/A for the responder)*	^TCP.ABORTreq /cleanup → S0	^TCP.ABORTreq /stop HBR-Timer → S0	N/A
TCP: ABORTind	N/A	[role = initiator] ^HL.PROTOCOL-ABORTind → S0	^HL.PROTOCOL-ABORTind /cleanup → S0	[not local peer abort] ^HL.PROTOCOL-ABORTind /stop HBR-Timer → S0	/stop HBR-Timer → S0
TCP: Timeout	N/A	^TCP.ABORTreq ^HL. PROTOCOL-ABORTind → S0 *(N/A for the responder)*	^TCP.ABORTreq ^HL. PROTOCOL-ABORTind /cleanup → S0	N/A	N/A

	S0 - CLOSED	S1 - TML STARTING	S2 - DATA TRANSFER	S3 - PEER ABORTING	S4 - TML CLOSING
TCP: Error	N/A	[role = initiator] ^HL.PROTOCOL-ABORTind [role = responder] /stop TMS-Timer → S0	^HL.PROTOCOL-ABORTind /cleanup → S0	[not local peer abort] ^HL.PROTOCOL-ABORTind /stop HBR-Timer → S0	/stop HBR-Timer → S0
TMS Timeout	N/A	^TCP.ABORTreq → S0 *(N/A for the initiator)*	^TCP.ABORTreq ^HL.PROTOCOL-ABORTind → S0 *(N/A for the initiator)*	N/A	N/A
HBT Timeout	N/A	N/A	^TCP.DATAreq(TML heartbeat) /start HBT-Timer	N/A	N/A
HBR Timeout	N/A	N/A	^TCP.ABORTreq ^HL.PROTOCOL-ABORTind /cleanup →S0	^TCP.ABORTreq [not local peer abort] ^HL.PROTOCOL-ABORTind → S0	^TCP.ABORTreq →S0
CPA Timeout	N/A	N/A	N/A	N/A	^TCP.ABORTreq →S0

ANNEX A

TML DIAGNOSTIC CODES

(Normative)

The diagnostic codes listed in this annex are used to report errors detected on the level of the Transport Mapping Layer.

The error codes are allocated in the range 128 to 255, which the Recommended Standards for SLE transfer services make available for technology specific use. Errors related to this Recommended Standard use the range 128 to 199, leaving the range 200 to 255 for use by implementations.

Code	Meaning
128	TML protocol error
129	Badly formatted TML message
130	Heartbeat parameters not acceptable
131	Association establishment timeout
132	Heartbeat receive timeout
133	Unexpected disconnect by peer
134	Premature disconnect during peer abort
135	Timeout during peer abort
199	Other reason

ANNEX B

DIFFERENCES WITH EARLIER IMPLEMENTATIONS

(Informative)

B1 INTRODUCTION

The precursor of this Recommended Standard is a document called 'Specification of an SLE API Proxy for TCP/IP and ASN.1' (reference [E6]), which was published by the European Space Agency. This document adopts the specification of the authentication scheme and of the service instance identifier from another ESA document called 'SLE C++ Application Program Interface for Transfer Services' (reference [E7]).

All known implementations of SLE transfer services at the time this Recommended Standard was published were based on reference [E6]. This annex describes the differences between this Recommended Standard and reference [E6] in order to support development of implementations that can interoperate with implementations based on reference [E6].

Two differences must be considered:

a) support for earlier versions of the SLE transfer services RAF, RCF, and CLTU; and

b) structure of the service instance identifier.

B2 EARLIER VERSIONS OF THE SERVICES RAF, RCF, AND CLTU

Reference [E6] was based on earlier versions of the Recommended Standards for the services RAF, RCF, and CLTU, namely

- reference [E3] for the Return All Frames service specification;
- reference [E4] for the Return Channel Frames specification; and
- reference [E5] for the Forward CLTU service specification.

All these draft Recommended Standards specify the version number 1, whereas references [2], [3], and [5] specify version number 2. Some of the data types specified in annex A of references [E3], [E4], and [E5] differ from those defined in references [2], [3], and [5]. An implementation wishing to interoperate with earlier implementations will have to use the ASN.1 specification in references [E3], [E4], and [E5].

Implementations invoking the BIND operation will have to know whether the responder conforms to this Recommended Standard or to reference [E6]. If the responder conforms to reference [E6], the initiator must specify version 1 in the BIND operation and use the ASN.1 types specified in the applicable draft Recommended Standard.

Implementations performing the BIND operations can derive the ASN.1 specification to use from the version number parameter in the BIND invocation.

NOTE – The ASN.1 encoding of the BIND invocation has not been modified and it is therefore possible to decode the BIND invocation with either ASN.1 specification.

B3 STRUCTURE OF THE SERVICE INSTANCE IDENTIFIER

Because references [E3], [E4], and [E5] did not specify the structure of the service instance identifier, a preliminary specification was provided by reference [E7]. This specification defined a set of attribute names that could be used within a service instance identifier and specified that all attribute values should be character strings.

The specification of the service instance identifier in annex A of references [2], [3], and [5] adopts a small subset of the attribute names defined by reference [E7] and additionally specifies the sequence of attributes in a service instance identifier. It adopts the specification that all attribute values shall be character strings. Therefore, a service instance identifier constructed according to the specification in references [2], [3], and [5] generally conforms to the specification given in reference [E7].

However, there is one exception, which must be taken into account by implementations wishing to support earlier implementations. For historical reasons the service type entry in the service instance identifier originally distinguished between Virtual Channel Frames (VCF) and Master Channel Frames (MCF), whereas the new specification has been aligned with the service definitions and allows only the attribute RCF (Return Channel Frames). Therefore, a service instance identifier for an RCF service instance with timely online delivery mode constructed according to the specification in reference [E7] would contain the attribute value pair ‘vcf=onlt1’ or ‘mcf=onlt1’, whereas the same identifier constructed according to reference [3] must contain the attribute-value pair ‘rcf=onlt1’.

ANNEX C

INDEX TO DEFINITIONS

(Informative)

This annex lists terms used in this Recommendation and, for each term, provides a reference to the definition of that term.

Term	Reference
abstract syntax	reference [13]
Abstract Syntax Notation One (ASN.1)	reference [9]
application (of SLE)	subsection 1.6.1.6.1
application identifier	subsection 1.6.1.6.4
association	references [2] to [6]
Basic Encoding Rules (BER)	reference [10]
communications service	references [2] to [6]
confirmed operation	references [2] to [6]
Distinguished Encoding Rules (DER)	reference [10]
encoding	reference [10]
encoding rules (of ASN.1)	reference [10]
initiator	reference [1]
Internet Protocol (IP)	reference [8]
invocation	references [2] to [6]
IP address	reference [8]
layer (of a protocol)	reference [13]
local application	subsection 1.6.1.6.2
module (of ASN.1)	reference [9]
operation	reference [1]
parameter (of an operation)	references [2] to [6]
peer application	subsection 1.6.1.6.3
port (of TCP)	reference [7]
port identifier	references [2] to [6]
port number	reference [7]

Term	Reference
primitive	reference [13]
responder	reference [1]
return	references [2] to [6]
segment (of TCP)	reference [7]
service provider (provider)	reference [1]
service user (user)	reference [1]
SLE protocol data unit (SLE-PDU)	reference [1]
SLE transfer service instance (service instance)	reference [1]
socket	reference [7]
transfer syntax	reference [13]
Transmission Control Protocol (TCP)	reference [7]
unconfirmed operation	references [2] to [6]

ANNEX D

ACRONYMS

(Informative)

This annex lists the acronyms used in this Recommendation.

AL	Authentication Layer
API	Application Program Interface
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCSDS	Consultative Committee for Space Data Systems
CPA	Close after Peer Abort (Timer)
DEL	Data Encoding Layer
DER	Distinguished Encoding Rules
EBNF	Extended Backus-Naur Form
HBR	Heartbeat Receive (Timer)
HBT	Heartbeat Transmit (Timer)
ICMP	Internet Control Message Protocol
IP	Internet Protocol
ISO	International Standardization Organization
ISP1	Internet SLE Protocol One
MSL	Maximum Segment Lifetime
MTU	Maximum Transmission Unit
OCL	Object Constraint Language
OSI	Open Systems Interconnection
PDU	Protocol Data Unit

RFC	Request For Comments
SLE	Space Link Extension
TCP	Transmission Control Protocol
TLI	Transport Layer Interface
TML	Transport Mapping Layer
TMS	TML Startup (Timer)
UML	Unified Modeling Language
XTI	X/Open Transport Interface

ANNEX E

INFORMATIVE REFERENCES

(Informative)

[E1] *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.

[E2] *Cross Support Concept—Part 1: Space Link Extension Services*. Report Concerning Space Data System Standards, CCSDS 910.3-G-2. Green Book. Issue 2. Washington, D.C.: CCSDS, April 2002.

[E3] *Space Link Extension – Return All Frames Service Specification,* Draft Recommendation for Space Data System Standards, CCSDS 911.1-R1.7, Draft 1.7 of Red Book Issue 2, September 1999.

[E4] *Space Link Extension – Return Virtual Channel Frames Service Specification,* Draft Recommendation for Space Data System Standards, CCSDS 911.2-R1.7, Draft 1.7 of Red Book Issue 2, September 1999.

[E5] *Space Link Extension – Forward CLTU Service Specification,* Draft Recommendation for Space Data System Standards, CCSDS 912.1-R1.99h, Draft 1.99h of Red Book Issue 2, February 2000.

[E6] *Specification of a SLE API Proxy for TCP/IP and ASN.1*, SLES-SW-API-0002-TOS-GCI, Issue 1.1, European Space Agency, February 2001.

[E7] *SLE C++ Application Program Interface for Transfer Services*, SLES SW API 0001 TOS GCI, Issue 1.1, December 2000.

[E8] *Internet Official Protocol Standards*, Postel, J. and J. Reynolds, eds., STD 1, RFC 2400, September 1998.

[E9] *Internet Control Message Protocol* – DARPA Internet Program Protocol Specification, Postel, J., STD 5, RFC 792, September 1981.

[E10] *Requirements for Internet Hosts - Communication Layers*, Braden, R. Ed., STD-3, RFC 1122, October 1989.

[E11] *Information technology – Portable Operating System Interface (POSIX) – Part 1g: Protocol Independent Interfaces (PII),* P1003.1g/D6.6, IEEE, 1998.

[E12] *OMG Unified Modeling Language Specification,* Version 1.3, OMG Document formal/2000-03-01 Object Management Group, March 2000.