Network Working Group Brian Bidulock
INTERNET-DRAFT OpenSS7 Corporation

Expires in six months January 10, 2002

SS7 TCAP-User Adaptation Layer
TUA
<draft-bidulock-sigtran-tua-00.ps>

Status of this Memo

Abstract

This document defines a protocol for the transport of any SS7 TCAP-User signalling (e.g, INAP, MAP, etc.) over IP using the Stream Control Transport Protocol[RFC 2960]. The protocol should be modular and symmetric, to allow it to work in diverse architectures, such as a Signalling Gateway and IP Signalling End-point architecture. Protocol elements are added to allow seamless operation between peers in the SS7 and IP domains.

1. Introduction

This draft defines a protocol for the transport of SS7 TCAP[Q.771, T1.114] Users (i.e, MAP, INAP, etc.) signalling messages over IP using the Stream Control Transmission Protocol (SCTP)[RFC 2960]. This protocol would be used between a Signalling Gateway (SG) and Signalling End-point located in an IP network. Additionally, the protocol can be used to transport SS7 TCAP users between two signalling end-points located within an IP network.

1.1. Scope

There is on-going integration of SCN networks and IP networks. Network service providers are designing all IP architectures that include support for SS7 signalling protocols. IP provides an effective way to transport user data and for operators to expand their networks and build new services. In these networks, there is a need for interworking between the SS7 and IP domains[RFC 2719].

This document details the delivery of TC-user messages (MAP, CAP, INAP, etc.) over IP between two signalling end-points. Consideration is given for the transport from an SS7 Signalling Gateway (SG) to an IP signalling node (such as an IP-resident Database) as described in the Framework Architecture for Signalling Transport[RFC 2719] This protocol can also support transport of TC-user messages between two end-points wholly contained within and IP network.

The delivery mechanism addresses the following criteria:

- Support for transfer of TCAP messages (INAP, MAP, etc.)
- Support for TCAP operation class 1, 2, 3 and 4 operation.
- Support for the seamless operation of TC-User protocol peers.
- Support for the management of SCTP transport associations between an SG and one ore more IP-based signalling nodes.

- Support for distributed IP-based signalling nodes.
- Support for the asynchronous reporting of status changes to management.

1.2. Terminology

Application Server (AS) – a logical entity serving a specific Routing Key. An example of an Application Server is a virtual database element handling all HLR or SCP transactions for a particular SS7 Signalling Point. The *AS* contains a set of one or more unique *Application Server Processes*, of which one or more is normally actively processing traffic. There is a 1:1 relationship between an Application Server and a Routing Key.

Application Server Process (ASP) – a process instance of an Application Server. An *Application Server Process* serves as an active, backup, load-share or broadcast process of an Application Server (e.g, part of a distributed signalling node or database element). Examples of ASPs are MGCs, IP SCPs, or IP HLRs. An ASP contains an SCTP end-point and may be configured to process traffic within more that one *Application Server*.

Association – refers to an SCTP association[RFC 2960]. The association provides the transport for the delivery of TCAP protocol data units and TUA layer peer messages.

Component Sub-layer (TC)
The Component Sub-layer of TCAP[Q.771].

Fail-over – the capability to reroute signalling traffic as required to an alternate Application Server Process, or group of ASPs, within an Application Server in the event of failure or unavailability of a currently used Application Server Process. *Fail-over* may apply upon the return to service of a previously unavailable Application Server Process.

Host – the computing platform that the process (SGP, ASP or IPSP) is running on.

IP Server Process (IPSP) – a process instance of an IP-based application. An IPSP is essentially the same as an ASP, except that it uses TUA in a point-to-point fashion.

Layer Management (LM) – a nodal function that handles the inputs and outputs between the TUA layer and a local management entity.

Message Transfer Part (MTP)
The Message Transfer Part[Q.701, T1.111] of the SS7 protocol.

Nodal Interworking Function (NIF) – an implementation dependent interworking function present at a Signalling Gateway that interworks primitives and procedures between the TCAP and TUA layers in the SG.

Network Appearance (NA) – a value that identifies the SS7 network context of a Routing Key. The *Network Appearance* value is of significance only within an administrative domain; it is coordinated between the SG and ASP.

Network Byte Order – the ordering of bytes most-significant-byte first, also referred to as Big Endian.

Routing Context (RC) – a value that uniquely identifies a Routing Key and an Application Server. *Routing Context* values are either configured using a configuration management interface, or by using the Routing Key Management (RKM) messages and procedures defined for TUA.

Routing Key (RK) – describes a set of SS7 parameters and parameter values that uniquely define the range of signalling traffic to be handled by a particular Application Server.

Signalling Connection Control Part (SCCP) – The Signalling Connection Control Part[Q.711] of the SS7 protocol.

Signalling Gateway (SG) – a signalling agent that exchanges SCN native signalling at the edge of the IP network[RFC 2719]. An SG appears to the SS7 network as an SS7 Signalling Point. An SG contains a set of one or more Signalling Gateway Processes, of which one or more is normally actively processing traffic. When an SG contains more than one SGP, the SG is a logical entity and the contained SGPs are assumed to be coordinated into a single management view both toward the SS7 network and toward the supported Application Servers.

Signalling Gateway Process (SGP) – a process instance of a Signalling Gateway. It serves as an active, backup, load-sharing or broadcast process of a Signalling Gateway.

Stream – an SCTP stream; a unidirectional logical channel established from one SCTP endpoint to another associated SCTP endpoint, within which all user messages are delivered in sequence, except for those submitted to the unordered delivery service.

Transaction Capabilities Application Part (TCAP) – The Transaction Capabilities Application Part[Q.771, T1.114] of the SS7 protocol.

Transaction Mapping Function (TMF) – an implementation dependent function that is responsible for resolving the address and application context presented in the incoming TUA message to the correct SCTP association and Routing Context for the desired application. The TMF **MAY** use routing context or routing key information as selection criteria for the appropriate SCTP association.

Transaction Sublayer (TR) – The Transaction Sublayer of TCAP[Q.771].

Transport Address – an address that serves as a source or destination for the unreliable packet transport service used by SCTP. In IP networks, a transport address is defined by the combination of IP address and an SCTP port number [1].

1.3. TUA Overview

1.3.1. Signalling Transport Architecture

The framework architecture that has been defined for SCN signalling transport over IP[RFC 2719] uses multiple components, including an IP transport protocol, a signalling common transport protocol and an adaptation module to support the services expected by a particular SCN signalling protocol from its underlying protocol layer.

In general terms, the TUA architecture can be modeled as a peer-to-peer architecture. The first section considers the SS7-to-IP interworking architectures for TCAP class 1, 2, 3, and 4 operations. For this case, it is assumed that the ASP initiates the establishment of the SCTP association with the SG.

1.3.2. Protocol Architecture for Classes 1, 2, 3 and 4

In this architecture (illustrated in *Figure 1*), the TCAP and TUA layers interface in the SG. A Nodal Interworking Function (NIF) provides for interworking between the TCAP and TUA layers and provides for the transfer of the user messages as well as management messages.

1.3.3. All IP Architecture

This architecture, illustrated in *Figure 2*, can be used to carry a protocol which uses the transport services of TCAP, but is contained within an IP network. This allows extra flexibility in developing networks, especially when interaction between legacy signalling is not needed. The architecture removes the need for a signalling gateway function.

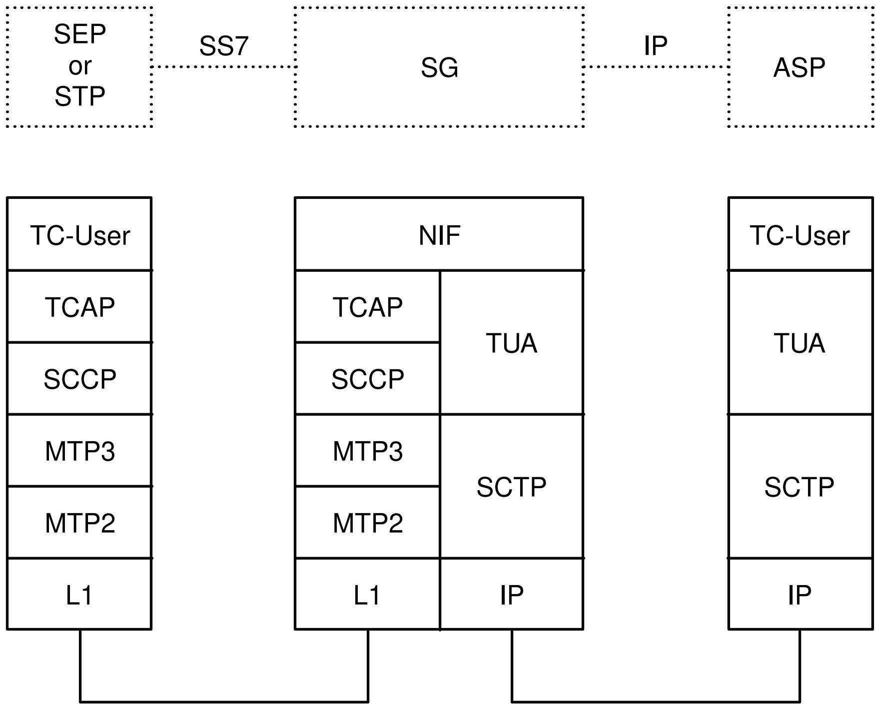


Figure 1. Protocol Architecture

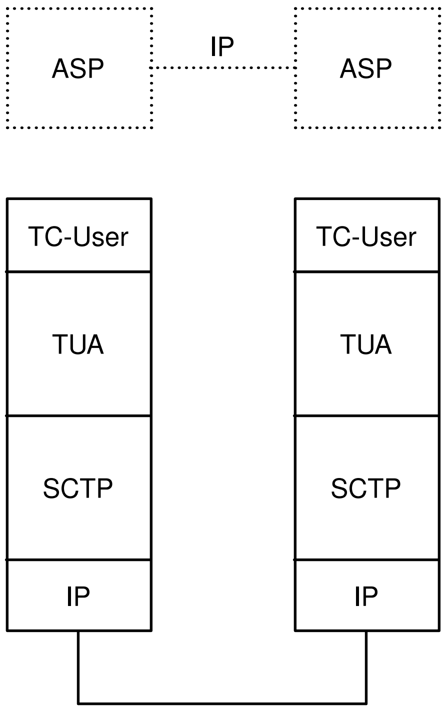


Figure 2. All IP Architecture

1.3.4. ASP Fail-over Model and Terminology

The TUA protocol supports ASP fail-over functions to support a high availability of transaction processing capability.

An Application Server can be considered as a list of all ASPs configured or registered to handled TC-user messages within a certain range of routing information, or within a certain set of transaction dialogues, known as a `Routing Key.´ One or more ASPs in the list may normally be active to handle traffic, while others may be inactive but available in the event of failure or unavailability of the active ASPs.

For operational considerations, see Appendix A.

1.3.5. Services Provided by the TUA Layer

1.3.5.1. Support for the transport of TCAP-User Messages

The TUA supports the transfer of TC-user messages. The TUA layer at the SG and the ASP support the seamless transport of user messages between the SG and the ASP.

1.3.5.1.1. TCAP Operation Class Support

Depending on the TC-users supported, the TUA shall support the 4 possible TCAP operation classes transparently. The TCAP operation classes are defined as follows:

Operation Class 1 – provides for transactions reporting both success and failure.
Operation Class 2 – provides for transactions reporting failure.
Operation Class 3 – provides for transactions reporting success.
Operation Class 4 – provides for transactions reporting neither success nor failure.

1.3.5.2. Native Management Functions

The TUA layer provides the capability to indicate errors associated with the TUA-protocol messages and to provide notification to local management and the remote peer as necessary.

1.3.5.3. Interworking with TCAP Management Functions

The TUA layer provides interworking with TCAP management functions at the SG for seamless interoperation between the SCN network and the IP network. TUA provides the following management functions:

(1) Provides an indication to the TC-user at an ASP that an SS7 subsystem, SCCP User Part or MTP Destination is unavailable.

(2) Provides an indication to the TC-user at an ASP that an SS7 subsystem, SCCP User Part or MTP Destination is available.

(3) Provides an indication to the TC-user at an ASP that an SS7 subsystem or MTP Destination is congested (flow controlled).

(4) Provides the initiation of an audit of SS7 subsystems or MTP Destinations status at the SG.

The interworking with TCAP management messages consists of DUNA, DAVA, DAUD, DRST, DUPU or SCON messages on receipt of management events to the appropriate ASPs. The primitives in *Table 1* are sent between the TCAP and TUA management functions in the SG to trigger events in the IP and SS7 domain.

The TUA layer provides transparent passing of SCCP availability, unavailability and congestion status indication primitives (N-STATE, N-PCSTATE and N-COORD) as provided for in *ITU-T Q.771 2.2.3*[Q.771].

1.3.5.4. Support for the Management of SCTP Associations

The TUA layer at the SGP maintains the availability state of all configured remote ASPs, to manage the SCTP Associations and the traffic between TUA peers. As well, the active/inactive and congestion state of remote ASPs is maintained.

The TUA layer **MAY** be instructed by local management to establish an SCTP association to a peer TUA node. This can be achieved using the M-SCTP_ESTABLISH primitives to request, indicate and confirm the establishment of an SCTP association with a peer TUA node. To avoid redundant SCTP associations between two TUA peers, one side (client) **SHOULD** be designated to establish the SCTP association, or TUA configuration information maintained to detect redundant associations (e.g, via knowledge of the expected local and remote SCTP endpoint addresses).

Local management **MAY** request from the TUA layer the status of the underlying SCTP associations using the M-SCTP_STATUS request and confirm primitives. Also, the TUA **MAY** autonomously inform local management of the reason for the release of an SCTP association, determined either locally within the TUA layer or by a primitive from the SCTP.

Also, the TUA layer **MAY** inform the local management of the change in status of an ASP or AS. This **MAY** be achieved using the M-ASP_STATUS request or M-AS_STATUS request primitives.

1.4. Functional Areas

Table 1. Mapping of Management Primitives

Name		**Reference**		**TUA Management Message**
Generic	**Specific**	**ITU-T Q.711**	**ANSI T1.112**	
N-STATE	Request Indication	6.3.2.3.2	2.3.2.3.2	DUNA DAVA SCON
N-PCSTATE	Indication	6.3.2.3.3	2.3.2.3.4	DUNA DAVA SCON DUPU
N-COORD	Request Indication Response Confirm	6.3.2.3.1	2.3.2.3.3	DRST

1.4.1. Dialogue Identifiers, Routing Contexts and Routing Keys

1.4.1.1. Overview

The mapping of TCAP messages into dialogues between the SGP and the Application Servers is determined by Dialogue Identifiers, Routing Keys and their associated Routing Contexts.

A Routing Key is essentially a set of TCAP parameters used to direct TCAP messages; whereas, the Routing Context parameter is a 4-byte value (unsigned integer) that is associated to that Routing Key in a 1:1 relationship. The Routing Context therefore can be viewed as an index into a sending node's Transaction Mapping Function tables containing the Routing Key entries.

Possible TCAP address/routing information that comprise a Routing Key entry includes, for example, a local and remote Point Code, Subsystem Number, Global Title Address, Application Context, local and remote Transaction Id pairs, or TC-User specific information such as User Information, IMSI, SLP. The particular information used to define a TUA Routing Key is application and network dependent, and none of the above examples are requirements for TUA.

An Application Server Process (ASP) may be configured to process signalling traffic related to more than one Application Server (AS), over a single SCTP Association. ASP Active (ASPAC) and ASP Inactive (ASPIA) management messages (see Section 3) use the Routing Context to discriminate signalling traffic to be started or stopped. At an ASP, the Routing Context parameter uniquely identifies the range of signalling traffic associated with each Application Server that the ASP is configured to receive.

1.4.1.2. Routing Key Limitations

Routing Keys **SHOULD** be unique in the sense that each received TCAP message **SHOULD** have a full or partial match to a single routing result. It is not necessary for the parameter range values within a particular Routing Key to be continuous. For example, an AS could be configured to support transaction processing for multiple ranges of subscribers that are not represented by contiguous Global Title Addresses.

1.4.1.3. Managing Routing Context and Routing Keys

There are two ways to provision a Routing Key at an SGP. A Routing Key may be configured statically using an implementation dependent management interface, or dynamically managed using the the TUA Routing Key registration procedures.

When using a management interface to configure Routing Keys, the Transaction Mapping Function within the SGP is not limited to the set of parameters defined in this document. Other implementation dependent distribution algorithms may be used.

1.4.1.4. Transaction Mapping Function

To perform its addressing and relaying capabilities, the TUA makes use of an Transaction Mapping Function (TMF). This function is considered part of TUA, but the way it is realized is left implementation or deployment dependent (local tables, SCCP GTT database, DNS[RFC 2916], LDAP, etc.)

The TMF is invoked when a message is received at the incoming interface. The TMF is responsible for resolving the application context, address and transaction ids presented in the incoming TCAP message to SCTP associations and destinations within the IP network. The TMF will select the key information available. The Routing Keys reference an Application Server, which will normally have one or more ASPs processing transactions for the AS. The availability and status of the ASPs is handled by TUA ASP management messages.

Possible SS7 application context, address or routing information that comprise a Routing Key entry includes, for example, SCCP subsystem number and SCCP addresses and Global Title addresses, Transaction ID, and Application Context.

It is expected that the routing keys will be provisioned via a MIB, dynamic registration or an external process, such as a database.

1.4.1.4.1. Transaction Mapping at the SG

To direct messages received from the SS7 network to the appropriate IP destination, the SGP must perform a transaction mapping function using information from the received TCAP message.

To support this transaction mapping, the SGP might, for example, maintain the equivalent of a network address translation table, mapping incoming TCAP message information to an Application Server for a particular application and set of transactions. This could be accomplished by comparing the addressing, dialog or component portions of the incoming TCAP message to currently defined Routing Keys in the SGP. These Routing Keys could in turn map directly to an

Application Server that is enabled by one or more ASPs. These ASPs proxy dynamic status information regarding their availability, transaction handling capabilities and congestion to the SGP using various management messages defined in the TUA protocol.

The list of ASPs in the AS is assumed to be dynamic, taking into account the availability, transaction handling capability and congestion status of the individual ASPs in the list, as well as configuration changes and possible fail-over mechanisms.

Normally, one or more ASPs are active in the AS (i.e, currently processing transactions) but in certain failure and transition cases it is possible that there may not be an active ASP available. The SGP will buffer the message destined for this AS for a time T(r) or until an ASP becomes available. When no ASP becomes available before expiry of T(r), the SGP will flush the buffered messages and initiate the appropriate TCAP abort procedures.

If there is no match for an incoming message, a default treatment **MAY** be specified. Possible solutions are to provide a default Application Server to direct all unallocated transactions to a (set of) default ASP(s), or to drop the messages and provide a notification to management. The treatment of unallocated transactions is implementation dependent.

1.4.1.4.2. Transaction Mapping at the ASP

To direct messages to the SS7 network, the ASP **MAY** perform a transaction mapping to choose the proper SGP for the given message. This is accomplished by observing the Application Context, Destination Address, Destination Transaction Id, and other elements of the outgoing message, SS7 network status, SGP availability, and Routing Context configuration tables.

A Signalling Gateway may be composed of one or more SGPs [2]. There is, however, no TUA messaging to manage the status of an SGP. Whenever an SCTP association to an SGP exists, it is assumed to be available. Also, every SGP of one SG communicating with one ASP regarding one AS provides identical SS7 connectivity to this ASP.

In general, an ASP routes responses to the SGP that it received messages from; within the routing context which it is currently active and receiving transactions. The routing context itself is used by the ASP to select the SGP.

1.4.1.5. Signalling Gateway SS7 Layers

The SG is responsible for terminating up to the TC-user of the SS7 protocol, and offering an IP-based extension to its users.

From an SS7 perspective, it is expected that the Signalling Gateway transmits and receives TCAP messages to and from the SS7 Network over standard SS7 network interfaces, using the services of the SCCP[Q.711] and MTP[Q.704] to provide transport of the messages.

Note that it is also possible for the SCCP services to be provided using the services of the SCCP-User Adaptation Layer (SUA)[SUA] and the MTP3-User Adaptation Layer (M3UA)[M3UA].

The TC-SAP through which TUA at the SG obtains its services could reside at a Signalling Transfer Point (STP) or Signalling End Point (SEP)[Q.705].

1.4.1.6. SS7 and TUA Interworking at the SG

The SGP provides a functional interworking of transport functions between the SS7 network and the IP network by also supporting the TUA adaptation layer. It allows the TCAP application to exchange components in dialogues with an IP-based Application Server Process where the peer TC-User protocol layer exists.

To perform TCAP management, it is required that the TC-User protocols at ASPs receive indications of subsystem availability and congestion, as well as user part availability and signalling point availability and congestion as they would be expected by an SS7 TCAP application. To accomplish this, the N-PCSTATE, N-STATE and N-COORD primitives received at the TCAP upper layer interface at the SG need to be propagated to the remote TC-user lower layer interface at the ASP.

SCCP management messages (such as SSP, SSA) and MTP management messages (such as TFP, TFA) received from the SS7 network **MUST NOT** be encapsulated. The SG **MUST** terminate these messages and generate TUA message as appropriate.

1.4.1.7. Application Server

A cluster of Application Servers is responsible for providing the overall support for one ore more SS7 upper layers. From an TCAP standpoint, an Application Part provides complete support for the upper layer service within a given Application Context. As an example, an Application Part providing HLR capabilities could provide complete support for GSM MAP HLR (and any other, MSC or VLR application parts located at the signalling point) for a given point code.

Where an ASP is connected to more than one SG, the TUA layer must maintain the status of configured SS7 destinations and route messages according to the availability/congestion status of potentially replicated subsystem.

1.4.1.8. SCTP Stream Mapping

The TUA supports SCTP streams. The SG and AS need to maintain a list of SCTP and TC-Users for mapping purposes. TC-Users requiring sequenced message transfer need to be sent over a stream using sequenced delivery.

TUA **SHOULD NOT** use stream 0 for TUA management messages. It is **OPTIONAL** that sequence delivery be used to preserve the order of management message delivery.

All TUA Dialogue Handling (DH) messages not using the optional component handling interface (i.e, DH messages with components included) **MAY** select unordered delivery, depending on the requirements of the TC-User [3]. All TUA Component Handling (CH) messages and Dialogue Handling (DH) messages with external components **SHOULD** select ordered delivery.

The stream selected is based upon the Sequence Control field in the Quality of Service parameter, the Dialogue Id given by the TC-User over the primitive interface and other traffic information available to the SGP or ASP.

1.4.2. Redundancy Models

1.4.2.1. Application Server Redundancy

All TQRY and SSNM messages (e.g, TC-BEGIN, N-STATE) which match a provisioned Routing Key at an SGP are mapped to an Application Server.

The Application Server is the set of all ASPs associated with a specific Routing Key. Each ASP in this set may be active, inactive or unavailable. Active ASPs handle traffic; inactive ASPs might be used when active ASPs become unavailable.

The fail-over model supports an "n+k" redundancy model, where "n" ASPs is the minimum number of redundant ASPs required to handle traffic and "k" ASPs are available to take over for a failed or available ASP. A "1+1" active/backup redundancy is a subset of this model. A simplex "1+0" model is also supported as a subset, with no ASP redundancy.

1.4.3. Flow Control

Local Management at an ASP may wish to stop traffic across an SCTP association to temporarily remove the association from service or to perform testing and maintenance activity. The function could optionally be used to control the start of traffic onto a newly available SCTP association.

1.4.4. Congestion Management

The TUA layer is informed of local and IP network congestion by means of an implementation-dependent function (e.g, an implementation-dependent indication from the SCTP of IP network congestion).

At an ASP or IPSP, the TUA layer indicates congestion to local TC-users by means of an appropriate TCAP primitive (N-PCSTATE, N-STATE, TC-NOTICE), as per current TCAP procedures, to invoke appropriate upper layer responses. When an SG determines that the transport of SS7 messages is encountering congestion, the SG might trigger SS7 Congestion messages to originating SS7 nodes, per the congestion procedures of the relevant SCCP[Q.711, T1.112] or MTP[Q.704, T1.111] standard. (The triggering of SS7 Management messages from an SG is an implementation-dependent function.)

1.5. Definition of TUA Boundaries

TUA has three protocol boundaries: an upper boundary between TUA and the TC-User; a lower boundary between TUA and SCTP; and a layer management boundary between TUA and the Layer Management Function.

1.5.1. Definition of Upper Boundary

The primitives and messages listed in *Table 2* are provided between the TUA and TC-User in support of Dialogue Handling[Q.771, T1.114].

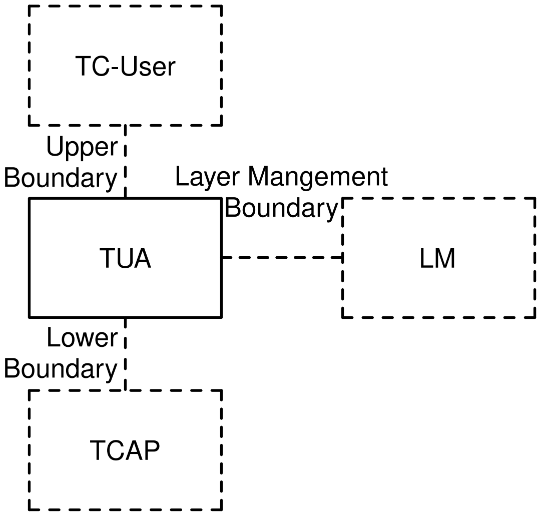


Figure 3. TUA Protocol Boundaries

Table 2. Mapping of Dialogue Handling Primitives

Generic Name	**Specific Name**	**ITU-T Q.771 Reference**	**ANSI T1.114 Message**	**TUA Msg**
TC-UNI	Request Indication	3.1.2.2.1	Unidirectional	TUNI
TC-BEGIN	Request Indication	3.1.2.2.2.1	Query w/ Perm	TQRY
———————	—————	——————	Query w/o Perm	
TC-CONTINUE (Initial)	Request Indication	3.1.2.2.2.2	Conv w/ Perm	TCNV
TC-CONTINUE (Non-initial)	Request Indication	3.1.2.2.2.3		
———————	—————	——————	Conv w/o Perm	
TC-END	Request Indication	3.1.2.2.2.4	Response	TRSP
TC-U-ABORT	Request Indication		U-Abort	TUAB
TC-P-ABORT	Indication	3.1.4.2	P-Abort	TPAB
TC-NOTICE	Indication	3.1.2.2.3	———————	TNOT

The primitives and messages listed in *Table 3* are provided between the TUA and TC-User in **OPTIONAL** support of Component Handling[Q.771, T1.114].

Table 3. Mapping of Component Handling Primitives

<table>
<tr><th>Generic Name</th><th>Specific Name</th><th>ITU-T Q.771 Reference</th><th>ANSI T1.114 Message</th><th>TUA Msg</th></tr>
<tr><td>TC-INVOKE</td><td>Request
Indication</td><td>3.1.3.2</td><td>Invoke L</td><td rowspan="2">CINV</td></tr>
<tr><td>————————</td><td>—————</td><td>——————</td><td>Invoke NL</td></tr>
<tr><td>TC-RESULT-L
TC-RESULT-NL</td><td>Request
Indication</td><td>3.1.3.3</td><td>Ret Result L
Ret Result NL</td><td>CRES</td></tr>
<tr><td>TC-U-ERROR</td><td>Request
Indication</td><td>3.1.3.4</td><td>Ret Error</td><td>CERR</td></tr>
<tr><td>TC-U-REJECT</td><td>Request
Indication</td><td>3.1.3.5</td><td rowspan="3">Reject</td><td rowspan="3">CREJ</td></tr>
<tr><td>TC-L-REJECT</td><td>Request
Indication</td><td rowspan="2">3.1.4.1</td></tr>
<tr><td>TC-R-REJECT</td><td>Request
Indication</td></tr>
<tr><td>TC-U-CANCEL
TC-L-CANCEL</td><td>Request
Indication</td><td>3.1.3.6</td><td>——————</td><td>CCAN</td></tr>
</table>

1.5.2. Definition of Boundary between TUA and Layer Management

M-SCTP_ESTABLISH request
Direction: LM->TUA
Purpose: LM request ASP to establish an SCTP association with its peer.

M-SCTP_ESTABLISH confirm
Direction: TUA -> LM
Purpose: ASP confirms to LM that it has established an SCTP association with its peer.

M-SCTP_ESTABLISH indication
Direction: TUA -> LM
Purpose: TUA informs LM that a remote ASP has established an SCTP association.

M-SCTP_RELEASE request
Direction: LM -> TUA
Purpose: LM requests ASP to release an SCTP association with its peer.

M-SCTP_RELEASE confirm
Direction: TUA -> LM
Purpose: ASP confirms to LM that it has released SCTP association with its peer.

M-SCTP_RELEASE indication
Direction: TUA -> LM
Purpose: TUA informs LM that a remote ASP has released an SCTP Association or the SCTP association has failed.

M-SCTP RESTART indication
Direction: TUA -> LM
Purpose: TUA informs LM that an SCTP restart indication has been received.

M-SCTP_STATUS request
Direction: LM -> TUA

Purpose: LM requests TUA to report the status of an SCTP association.

M-SCTP_STATUS confirm
Direction: TUA -> LM
Purpose: TUA responds with the status of an SCTP association.

M-SCTP_STATUS indication
Direction: TUA -> LM
Purpose: TUA reports the status of an SCTP association.

M-ASP_STATUS request
Direction: LM -> TUA
Purpose: LM requests TUA to report the status of a local or remote ASP.

M-ASP_STATUS confirm
Direction: TUA -> LM
Purpose: TUA reports status of local or remote ASP.

M-AS_STATUS request
Direction: LM -> TUA
Purpose: LM requests TUA to report the status of an AS.

M-AS_STATUS confirm
Direction: TUA -> LM
Purpose: TUA reports the status of an AS.

M-NOTIFY indication
Direction: TUA -> LM
Purpose: TUA reports that it has received a Notify message from its peer.

M-ERROR indication
Direction: TUA -> LM
Purpose: TUA reports that it has received an Error message from its peer or that a local operation has been unsuccessful.

M-ASP_UP request
Direction: LM -> TUA
Purpose: LM requests ASP to start its operation and send an ASP Up message to its peer.

M-ASP_UP confirm
Direction: TUA -> LM
Purpose: ASP reports that is has received an ASP UP Ack message from its peer.

M-ASP_UP indication
Direction: TUA -> LM
Purpose: TUA reports it has successfully processed an incoming ASP Up message from its peer.

M-ASP_DOWN request
Direction: LM -> TUA
Purpose: LM requests ASP to stop its operation and send an ASP Down message to its peer.

M-ASP_DOWN confirm
Direction: TUA -> LM
Purpose: ASP reports that is has received an ASP Down Ack message from its peer.

M-ASP_DOWN indication
Direction: TUA -> LM

Purpose: TUA reports it has successfully processed an incoming ASP Down message from its peer, or the SCTP association has been lost or reset.

M-ASP_ACTIVE request
Direction: LM -> TUA
Purpose: LM requests ASP to send an ASP Active message to its peer.

M-ASP_ACTIVE confirm
Direction: TUA -> LM
Purpose: ASP reports that is has received an ASP Active Ack message from its peer.

M-ASP_ACTIVE indication
Direction: TUA -> LM
Purpose: TUA reports it has successfully processed an incoming ASP Active message from its peer.

M-ASP_INACTIVE request
Direction: LM -> TUA
Purpose: LM requests ASP to send an ASP Inactive message to its peer.

M-ASP_INACTIVE confirm
Direction: LM -> TUA
Purpose: ASP reports that is has received an ASP Inactive Ack message from its peer.

M-ASP_INACTIVE indication
Direction: TUA -> LM
Purpose: TUA reports it has successfully processed an incoming ASP Inactive message from its peer.

M-AS_ACTIVE indication
Direction: TUA -> LM
Purpose: TUA reports that an AS has moved to the AS-ACTIVE state.

M-AS_INACTIVE indication
Direction: TUA -> LM
Purpose: UA reports that an AS has moved to the AS-INACTIVE state.

M-AS_DOWN indication
Direction: TUA -> LM
Purpose: UA reports that an AS has moved to the AS-DOWN state.

M-RK_REG request
Direction: LM -> TUA
Purpose: LM requests ASP to register RK(s) with its peer by sending REG REQ message

M-RK_REG confirm
Direction: TUA -> LM
Purpose: ASP reports that it has received REG RSP message with registration status as successful from its peer.

M-RK_REG indication
Direction: TUA -> LM
Purpose: TUA informs LM that it has successfully processed an incoming REG REQ message.

M-RK_DEREG request
Direction: LM -> TUA
Purpose: LM requests ASP to deregister RK(s) with its peer by sending DEREG REQ message.

M-RK_DEREG confirm
Direction: TUA -> LM

Purpose: ASP reports that it has received DEREG REQ message with deregistration status as successful from its peer.

M-RK_DEREG indication
Direction: TUA -> LM
Purpose: TUA informs LM that it has successfully processed an incoming DEREG REQ from its peer.

1.5.3. Definition of the Lower Boundary

The upper layer primitives provided by the SCTP are provided in the SCTP specification "Stream Control Transmission Protocol (SCTP)"[RFC 2960].

2. Conventions

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **NOT RECOMMENDED**, **MAY**, and **OPTIONAL**, when they appear in this document, are to be interpreted as described in[RFC 2119].

In this document, the following conventions are used to describe how a parameter is used in the message:

Mandatory The parameter **MUST** be present in the message. A message listing a parameter as *Mandatory* without containing such a parameter is is incorrectly formatted.

Conditional The parameter **SHOULD** be present in the message under the conditions specified. A message listing a parameter as *Conditional* without containing such a parameter under the conditions specified is incorrectly formatted.

Optional The parameter **MAY** be present in the message as specified. A message listing a parameter as *Optional* without containing such a parameter is correctly formatted.

3. Protocol Elements

The general message format includes a Common Message Header together with a list of zero or more parameters as defined by the Message Type.

For forward compatibility, all Message Types **MAY** have attached parameters even if none are specified in this version.

3.1. Common Message Header

The protocol messages for the TCAP-User Adaptation Protocol (TUA) require a message structure that contains a version, message type, message length and message contents:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Version    |   Reserved    | Message Class | Message Type  |
+---------------+---------------+---------------+---------------+
|                        Message Length                         |
+---------------------------------------------------------------+
|                         Message Data                          |
```

Notes:

- This message header is common among all signalling protocol adaptation layers.
- The 'data' portion of TUA messages **SHALL** contain zero or more TUA parameters, and **SHALL NOT** contain an encapsulated TCAP message.
- All fields in the TUA message **MUST** be transmitted in the network byte order, unless otherwise stated.

3.1.1. TUA Protocol Version

Version: 8-bits (unsigned integer)

The *Version* field of the Common Message Header contains the version of the TUA adaptation layer. The supported versions are:

1 – TUA Version 1.0

3.1.2. Message Classes

Message Class: 8-bits (unsigned integer)

The *Message Class* field of the Common Message Header contains the class of the message. The supported classes are as follows:

0	Management (MGMT) Message
7	Reserved for Other Signalling Adaptation Layers
2	SS7 Signalling Network Management (SSNM) Messages
3	ASP State Maintenance (ASPSM) Messages
4	ASP Traffic Maintenance (ASPTM) Messages
5	TUA Dialogue Handling (DH) Messages
6	TUA Component Handling (CH) Messages
7	Reserved for Other Signalling Adaptation Layers
8	Reserved for Other Signalling Adaptation Layers
9	Routing key Management (RKM) Messages
10 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

3.1.3. Message Types

Message Type: 8-bits (unsigned integer)

The *Message Type* field of the Common Message Header contains the type of message within a message class. The supported types of messages within the supported classes are as follows:

Management (MGMT) Messages

0	Error (ERR)
1	Notify (NTFY)
2 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

SS7 Signalling Network Management (SSNM) Messages

0	Reserved
1	Destination Unavailable (DUNA)
2	Destination Available (DAVA)
3	Destination State Audit (DAUD)
4	Destination Congestion (SCON)
5	Destination User Part Unavailable (DUPU)
6	Destination Restricted (DRST)
7 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

Application Server Process State Maintenance (ASPSM) Messages

0	Reserved
1	ASP Up (UP)
2	ASP Down (DOWN)
3	Heartbeat (BEAT)
4	ASP Up Ack (UP ACK)
5	ASP Down Ack (DOWN ACK)
6	Heartbeat Ack (BEAT ACK)
7 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

Application Server Process Traffic Maintenance (ASPTM) Messages

0	Reserved
1	ASP Active (ASPAC)
2	ASP Inactive (ASPIA)
3	ASP Active Ack (ASPAC ACK)
4	ASP Inactive Ack (ASPIA ACK)
5 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

Routing Key Management (RKM) Messages

0	Reserved
1	Registration Request (REG REQ)
2	Registration Response (REG RSP)
3	Deregistration Request (DEREG REQ)
4	Deregistration Response (DEREG RSP)
5 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

TUA Dialogue Handling (DH) Messages

0	Unidirectional(TUNI)
1	Query (TQRY)
2	Conversation (TCNV)
3	Response (TRSP)
4	U-Abort (TUAB)
5	P-Abort (TPAB)
6	Notice (TNOT)
7 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

TUA Component Handling (CH) Messages

1	Invoke (CINV)
2	Result (CRES)
3	Error (CERR)
4	Reject (CREJ)
5	Cancel (CCAN)
6 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

3.1.4. Message Length

Message Length: 32-bits (unsigned integer)

The *Message Length* field of the Common Message Header defines the length of the message in octets, including the header.

3.1.5. Tag-Length-Value Format

TUA messages consist of a Common Message Header followed by zero or more parameters, as defined by the message type. The Tag-Length-Value (TLV) parameters contained in a message are defined in a Tag-Length-Value format as shown below [4].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Parameter Tag        |       Parameter Length        |
+-------------------------------+-------------------------------+
\                                                               \
/                       Parameter Value                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Parameter Tag: 16-bits (unsigned integer)
The Parameter Tag field is a 16-bit identifier of the type of parameter. It takes a value of 0 to 65534.

Parameter Length: 16-bits (unsigned integer)
The Parameter Length field contains the size of the parameter in bytes, including the Parameter Tag, Parameter Length, and Parameter Value fields. The Parameter Length does not include any padding bytes.

Parameter Value: variable-length
The Parameter Value field contains the actual information to be transferred in the parameter. The total length of a parameter (including Tag, Parameter Length and Value fields) **MUST** be a multiple of 4 bytes. If the length of the parameter is not a multiple of 4 bytes, the sender **MUST** pad the Parameter at the end (i.e., after the Parameter Value field) with all zero bytes. The length of the padding **MUST NOT** be included in the parameter length field. A sender **SHOULD NOT** pad with more than 3 bytes. The receiver **MUST** ignore the padding bytes.

3.2. TUA Message Header

In addition to the Common Message Header, a specific message header is included for TUA messages. The TUA message header will immediately follow the Common Message Header in TUA Dialogue Handling (DH) and Component Handling (CH) messages.

The *TUA Message Header* is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |             Length = 8        |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Routing Context                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0013          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Correlation Id                         |
+-------------------------------+-------------------------------+
|         Tag = 0x0401          |             Length = 8        |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Dialogue Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The TUA Message header can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
Correlation Id	Conditional	*2
Dialogue Id	Conditional	*3

Note 1: When an ASP is registered or configured for multiple AS with an SG, the Routing Context **MUST** be present in the TUA Message Header. The Routing Context **SHOULD** always be placed in the TUA Message Header. When the Routing Context is present in the TUA Message Header it **SHOULD** be placed first in the header because the context of the Dialogue Id depends on the Routing Context.

Note 2: Under some circumstances, the Correlation Id parameter **MUST** be included in the TUA Message Header. See sections "Correlation Id" and "ASP Active Procedures".

Note 3: When an AS is handling multiple Dialogues, the Dialogue Id parameter **MUST** be placed in the TUA Message Header. The Dialogue Id parameter **SHOULD** always be placed in the TUA Message Header. The Dialogue Id parameter **MAY** be excluded from the TUA header for *TUNI* and *TPAB* DH messages, or may be included but then **MUST** contain a value of zero.

3.3. TUA Dialogue Handling (DH) Messages

The following section describes the TUA Dialogue Handling (DH) messages and parameter contents. The general message format includes a Common Message Header, the TUA Message Header and the DH Message Header, together with a list of zero or more parameters as defined by the Message Type. For forward compatibility, all Message Types **MAY** have optional attached parameters in addition to the message headers.

3.3.1. DH Message Header

In addition to the Common Message Header and TUA Message Header, a specific message header is included for TUA Dialogue Handling (DH) messages. The DH Message Header will immediately follow the TUA Message header in these messages.

The *DH Message Header* is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0402          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Dialogue Flags                         |
+-------------------------------+-------------------------------+
|         Tag = 0x0403          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                      Quality of Service                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DH Message header* contains the following parameters:

Parameters	
Dialogue Flags	Mandatory
Quality of Service	Mandatory

3.3.2. Unidirectional (TUNI)

The *Unidirectional (TUNI)* Request message is sent from an ASP to an SG or IPSP to invoke a TCAP class 4 operation. The *TUNI* Indication message is sent from an SGP to an ASP to indicate the TCAP class 4 operation.

The *TUNI* message corresponds to the ITU-T ‘TC-UNI’ primitive[Q.771], and the ITU-T and ANSI ‘Unidirectional’ message[Q.773, T1.114].

The *TUNI* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0404          |           Length              |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Destination Address                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0405          |           Length              |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
```

```
\                                                               \
/                      Originating Address                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0406          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Context Name                   /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0407          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        User Information                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0408          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Security Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0409          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Confidentiality                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x040E          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Components                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TUNI* message can contain the following parameters:

Parameters		
Destination Address	Conditional	*1
Originating Address	Conditional	*1
Application Context Name	Optional	
User Information	Optional	
Security Context	Optional	
Confidentiality	Optional	
Components	Optional	*2

Note 1: The *Destination Address* or *Originating Address* parameter **MUST** be present in the *TUNI* message when either parameter is not implied by the Routing Context in the TUA Message Header.

Note 2: Any components **SHOULD** be included in the *TUNI* messages but **MAY** be formatted in separate TUA Component Handling (CH) messages.

3.3.3. Query (TQRY)

The *Query (TQRY)* message is sent to a TUA peer to begin a new dialogue between TC-Users.

The *TQRY* message corresponds to the ITU-T 'TC-BEGIN' primitive[Q.771], the ITU-T 'Begin' message[Q.773] and the ANSI 'Query' message[T1.114].

The *TQRY* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0410          |             Length            |
```

```
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Transaction Id                         |
+-------------------------------+-------------------------------+
|         Tag = 0x0404          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Destination Address                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0405          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Originating Address                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0406          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Context Name                   /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0407          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        User Information                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0408          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Security Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0409          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Confidentiality                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x040E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Components                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TQRY* message can contain the following parameters:

Parameters		
Transaction Id	Mandatory	
Destination Address	Conditional	*1
Originating Address	Conditional	*1
Application Context Name	Optional	
User Information	Optional	
Security Context	Optional	
Confidentiality	Optional	
Components	Optional	*2

Note 1: The *Destination Address* or *Originating Address* parameter **MUST** be present in the *TQRY* message when the parameter is not implied by the Routing Context in the TUA Message Header.

Note 2: Any components **SHOULD** be included in the *TQRY* messages but **MAY** be formatted in separate Component Handling (CH) messages.

3.3.4. Conversation (TCNV)

The *Conversation (TCNV)* message is used in response to a *TQRY* message or another *TCNV* message.

When sent in response to a *TQRY* message, the *TCNV* message confirms and continues a dialogue; when in response to a received *TCNV* message, it only continues a dialogue. The Dialogue Flags in the DH Message Header indicate whether the initiator of the *TCNV* message give permission to the peer to terminate the dialogue.

The *TCNV* message corresponds to the ITU-T ‘TC-CONTINUE’ primitive[Q.771], ITU-T ‘Continue’ message[Q.773] and the ANSI ‘Conversation’ message[T1.114].

The *TCNV* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0410          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Transaction Id                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0405          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Originating Address                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0406          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Context Name                   /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0407          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        User Information                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0408          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Security Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0409          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Confidentiality                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x040E          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Components                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TCNV* message can contain the following parameters:

Parameters		
Transaction Id	Conditional	*1
Originating Address	Conditional	*2
Application Context Name	Conditional	*3
User Information	Conditional	*3
Security Context	Conditional	*3

Confidentiality	Conditional	*3
Components	Optional	*4

Note 1: The *Transaction Id* parameter **MUST** be present in the **TCNV** message when the message is sent in response to a **TQUR** message. The *Transaction Id* parameter contains the Transaction Identifier assigned by the remote TC-User.

Note 2: The *Originating Address parameter* **MUST** be present in the *TCNV* message when the message is used in response to a *TQRY* message and the parameter is not implied by the Routing Context in the TUA Message Header.

Note 3: These dialogue portion parameters **SHOULD** only be optionally included in the *TCNV* message when the message is used in response to a *TQRY* message. When the *TCNV* message is sent in response to a received *TCNV* message, these parameters **SHOULD NOT** be included in the responding *TCNV* message.

Note 4: Any components **SHOULD** be included in the *TCNV* messages but **MAY** be formatted in separate Component Handling (CH) messages.

3.3.5. Response (TRSP)

The *Response (TRSP)* message is used in response to a *TQRY* message or *TCNV* message to complete and existing dialogue.

When sent in response to a *TQRY* message, the *TRSP* message confirms and completes a dialogue; when in response to a received *TCNV* message, it only terminates a dialogue.

The *TRSP* message corresponds to the ITU-T 'TC-END' primitive[Q.771], ITU-T 'End' message[Q.773] and the ANSI 'Response' message[T1.114].

The *TRSP* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040A          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Termination                          |
+-------------------------------+-------------------------------+
|         Tag = 0x0406          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Context Name                   /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0407          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        User Information                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0408          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Security Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0409          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Confidentiality                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x040E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
```

```
/                          Components                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TRSP* message can contain the following parameters:

Parameters		
Termination	Mandatory	
Application Context Name	Optional	*1
User Information	Optional	*1
Security Context	Optional	*1
Confidentiality	Optional	*1
Components	Optional	*2

Note 1: These dialogue portion parameters **SHOULD** only be optionally included in the *TRSP* message when it is issued in response to an *TQRY* message. When the *TRSP* message is in response to a *TCNV* message, the dialogue portion parameters **SHOULD NOT** be included in the *TRSP* message.

Note 2: Any components **SHOULD** be included in the *TRSP* messages but **MAY** be formatted in separate TUA Component Handling (CH) messages.

3.3.6. U-Abort (TUAB)

The TUA peer sends an *U-Abort (TUAB)* message when it wishes to abort a dialogue, either under TUA-user control (TC-U-ABORT).

When sent in response to a *TQRY* message, the *TUAB* message negatively confirms and aborts a dialogue; when in response to a received *TCNV* message, it only aborts a dialogue.

The *TUAB* message corresponds to the ITU-T ‘TC-U-ABORT’ primitive[Q.771], the ITU-T ‘Abort’ message[Q.773] and the ANSI ‘Abort’ message[T1.114].

The *TUAB* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Abort Reason                          |
+-------------------------------+-------------------------------+
|         Tag = 0x0405          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Originating Address                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0406          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Context Name                   /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0407          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        User Information                       /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TUAB* message can contain the following parameters:

Parameters		
Abort Reason	Mandatory	
Application Context Name	Conditional	*1
User Information	Optional	*2

Note 1: These dialogue portion parameters **SHOULD** only be optionally included in the *TUAB* message when it is issued in response to an *TQRY* message. When the *TUAB* message is in response to a *TCNV* message, the dialogue portion parameters **SHOULD NOT** be included in the *TUAB* message.

Note 2: The User Information parameter carries any User Abort Information.

3.3.7. P-Abort (TPAB)

The TUA peer sends an *P-Abort (TPAB)* message when it wishes to abort a dialogue, either under TUA control (TC-P-ABORT).

The *TPAB* message corresponds to the ITU-T 'TC-P-ABORT' primitive[Q.771], the ITU-T 'Abort' message[Q.773] and the ANSI 'Abort' message[T1.114].

The *TPAB* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Abort Cause                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TPAB* message can contain the following parameters:

Parameters	
Abort Cause	Mandatory

3.3.8. Notice (TNOT)

An SG sends a *Notice (TNOT)* message when it wishes to inform the ASP of a network condition that concerns the transmission of TCAP or TUA messages to the remote TC-User in a dialogue[Q.775]. It is used at the SG when an SCCP message containing TC-User information from an AS has been returned in a UDTS when the "Return Option" flag was set in the Quality of Service parameters when the message was sent.

The *TNOT* message corresponds to the ITU-T[Q.771] TC-NOTICE primitive.

The *TNOT* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Report Cause                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *TNOT* message can contain the following parameters:

Parameters	
Report cause	Mandatory

3.4. TUA Component Handling (CH) Messages

The following section describes the TUA Component Handling messages and parameter contents. The general message format includes a Common Message Header, a TUA Message Header, a CH Message Header, followed by a list of zero or more parameters as defined by the Message Type. For forward compatibility, all Message Types **MAY** have attached optional parameters in addition to the message headers.

Component Handling (CH) messages are used to convey components associated with operations within a dialogue. They are issued prior to the Dialogue Handling (DH) message with which they are associated, but are received after receiving a Dialogue Handling (DH) message that has the "Components Present" bit set in the Dialogue Flags parameter within the DH message.

3.4.1. CH Message Header

In addition to the Common Message Header and TUA Message Header, a specific message header is included for TUA Component Handling (CH) messages. The CH Message Header will immediately follow the TUA Message Header in these messages.

The *CH Message Header* if formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0411          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Invoke Id                           |
+-------------------------------+-------------------------------+
|         Tag = 0x0412          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Linked Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CH Message Header* can contain the following parameters:

Parameters	
Invoke Id	Mandatory
Linked Id	Optional

3.4.2. Invoke (CINV)

The *Invoke (CINV)* message is used to invoke an operation within a dialogue.

The *CINV* message corresponds to the ITU-T ‘TC-INVOKE’ primitive[Q.771], the ITU-T ‘Invoke’ component[Q.773], and the ANSI ‘Invoke (Last)’ and ‘Invoke (Not Last)’ components[T1.114].

The *CINV* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0413          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Component Flags                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0418          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                            Timeout                            |
+-------------------------------+-------------------------------+
|         Tag = 0x0414          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Operation                           /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0415          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Parameters                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CINV* message can contain the following parameters:

Parameters		
Component Flags	Mandatory	*1
Timeout	Mandatory	
Operation	Mandatory	
Parameters	Optional	

Note 1: The Component Flags parameter **MAY** be ignored by the receiver of the *CINV* message for ITU-T protocol variants of TC-Users that do not support the concept of a "Not Last" TC-INVOKE primitive.

3.4.3. Result (CRES)

The *Result (CRES)* message is used to report the successful completion of an operation within a dialogue.

The *CRES* message corresponds to the ITU-T ‘TC-RESULT-L’ and ‘TC-RESULT-NL’ primitives[Q.771], the ITU-T ‘Return Result (Last)’ and ‘Return Result (Not Last)’ components[Q.773] and the ANSI ‘Return Result (Last)’ and ‘Return Result (Not Last)’ components.

The *CRES* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0413          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Component Flags                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0414          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Operation                           /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0415          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Parameters                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CRES* message can contain the following parameters:

Parameters		
Component Flags	Mandatory	
Operation	Conditional	*1
Parameters	Optional	

Note 1: The Operation parameter **MUST** be present in the **CRES** message when the Parameters parameter is also present.

3.4.4. Error (CERR)

The *Error (CERR)* message is used to report the failure of an operation within a dialogue.

The *CERR* message corresponds to the ITU-T 'TC-U-ERROR' primitive[Q.771], the ITU-T 'Return Error' component[Q.773] and the ANSI 'Return Error' component[T1.114].

The *CERR* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0416          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                             Error                             /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0415          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Parameters                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CERR* message can contain the following parameters:

Parameters		
Error	Mandatory	
Parameters	Conditional	*1

Note 1: The *Parameters* parameter is only included in the message for specific error codes.

3.4.5. Reject (CREJ)

The *Reject (CREJ)* message is used to reject an operation within a dialogue.

The *CREJ* message corresponds to the ITU-T 'TC-L-REJECT', 'TC-R-REJECT' and 'TC-U-REJECT' primitives[Q.771], the ITU-T 'Reject' component[Q.773] and the ANSI 'Reject' component[T1.114].

The *CREJ* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0417          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Problem Code                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CREJ* message can contain the following parameters:

Parameters	
Problem Code	Mandatory

3.4.6. Cancel (CCAN)

The *Cancel (CCAN)* message is used to cancel an operation within a dialogue.i

The *CCAN* message corresponds to the ITU-T 'TC-L-CANCEL' and 'TC-U-CANCEL' primitives[Q.771].

The *CCAN* message presently contains no Message-Type-specific parameters.

3.5. SS7 Signalling Network Management (SSNM) Messages

SS7 Signalling Network Management (SSNM) Messages are used to convey network management information to the TC-User. Theses messages correspond to specific N-STATE, N-PCSTATE and N-COORD primitives.

3.5.1. Destination Unavailable (DUNA)

The **Destination Unavailable (DUNA)** message is sent from an SGP to all concerned ASPs to indicate the unavailability of an SS7 SCCP subsystem or signalling point. The TC-User at the ASP is expected to stop traffic to TC-User peers at the affected subsystems or signalling points via the SG initiating the **DUNA** message.

When the **DUNA** message contains the *Subsystem Number* parameter, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-STATE' primitive. When the **DUNA** message does not contain the *Subsystem Number* parameter, message, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-PCSTATE' primitive.

The *DUNA* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Affected Point Code                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Subsystem Number                       |
+-------------------------------+-------------------------------+
|         Tag = 0x041A          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                Subsystem Multiplicity Indicator               |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DUNA* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	
Subsystem Number	Conditional	*1
Subsystem Multiplicity Indicator	Optional	*2
Info String	Optional	

Note 1: The *Subsystem Number* parameter **SHALL** be present in the **DUNA** message when indicating the unavailability of a subsystem, and **SHALL NOT** be present when indicating the unavailability of a signalling point.

Note 2: The *Subsystem Multiplicity Indicator* parameter **SHOULD NOT** be present in the **DUNA** message when the *Subsystem Number* parameter is not also present.

3.5.2. Destination Available (DAVA)

The **Destination Available (DAVA)** message is sent from an SGP to all concerned ASPs to indicate the availability of an SS7 SCCP Subsystem or signalling point. The TC-User at the ASP is expected to resume traffic to TC-Users peers at the affected subsystems or signalling points via the SG initiating the **DAVA** message.

When the **DAVA** message contains the *Subsystem Number* parameter, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-STATE' primitive. When the **DAVA** message does not contain the *Subsystem Number* parameter, message, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-PCSTATE' primitive.

The *DAVA* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Affected Point Code                     /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Subsystem Number                       |
+-------------------------------+-------------------------------+
|         Tag = 0x041A          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                Subsystem Multiplicity Indicator               |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DAVA* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	
Subsystem Number	Conditional	*1

Subsystem Multiplicity Indicator	Optional	*2
Info String	Optional	

Note 1: The *Subsystem Number* parameter **SHALL** be present in the **DAVA** message when indicating the availability of a subsystem, and **SHALL NOT** be present when indicating the availability of a signalling point.

Note 2: The *Subsystem Multiplicity Indicator* parameter **SHOULD NOT** be present in the **DAVA** message when the *Subsystem Number* parameter is not also present.

3.5.3. Destination State Audit (DAUD)

The **Destination State Audit (DAUD)** message is sent from an ASP to an SG to query the availability state of routes to SS7 SCCP subsystems or signalling points. A **DAUD** message **MAY** be sent periodically after the ASP has received a **DUNA** message, and until a **DAVA** is received for the affected subsystem or signalling point. The **DAUD** message can also be sent when an ASP recovers from isolation from the SG.

When the **DAVA** message contains the *Subsystem Number* parameter, the message is soliciting responses that correspond to the ITU-T[Q.711] and ANSI[T1.112] 'N-STATE' primitive. When the **DAVA** message does not contain the *Subsystem Number* parameter, message, the message soliciting responses that correspond to the ITU-T[Q.711] and ANSI[T1.112] 'N-PC-STATE' primitive.

The *DAUD* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Affected Point Code                     /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Subsystem Number                       |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DAUD* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	
Subsystem Number	Conditional	*1
Info String	Optional	

Note 1: The *Subsystem Number* parameter **SHALL** be present in the **DAVA** message when auditing the status of a subsystem, and **SHALL NOT** be present when auditing the status of a signalling point.

3.5.4. Network Congestion (SCON)

The **Network Congestion (SCON)** message is sent from an SG to all concerned ASPs to indicate that the congestion level in the SS7 network to a specified subsystem or signalling point has changed. The TC-User at the ASP is expected to stop traffic at the indicated importance level to TC-User peers at the affected subsystems or signalling points via the SG initiating the **SCON** message.

When the **SCON** message contains the *Subsystem Number* parameter, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-STATE' primitive. When the **SCON** message does not contain the *Subsystem Number* parameter, message, the message corresponds to the ITU-T[Q.711] and ANSI[T1.112] 'N-PCSTATE' primitive.

The *SCON* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Affected Point Code                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x041B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Congestion Level                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Subsystem Number                        |
+-------------------------------+-------------------------------+
|         Tag = 0x041A          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|               Subsystem Multiplicity Indicator                |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *SCON* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	
Congestion Level	Mandatory	
Subsystem Number	Optional	*1
Subsystem Multiplicity Indicator	Optional	*2
Info String	Optional	

Note 1: The *Subsystem Number* parameter **SHALL** be present in the **SCON** message when indicating the congestion of a subsystem, and **SHALL NOT** be present when indicating the congestion of a signalling point.

Note 2: The *Subsystem Multiplicity Indicator* parameter **SHOULD NOT** be present in the **SCON** message when the *Subsystem Number* parameter is not also present.

3.5.5. Destination User Part Unavailable (DUPU)

The **Destination User Part Unavailable (DUPU)** message is sent from an SG to all concerned ASPs to indicate the unavailability of an SS7 SCCP.

The **DUPU** message corresponds to the ITU[Q.711] and ANSI[T1.112] 'N-PCSTATE' primitive.

The *DUPU* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Affected Point Code                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x041C          |            Length = 8         |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           User/Cause                          |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Info String                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DUPU* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	
User/Cause	Mandatory	*1
Info String	Optional	

Note 1: The *User* field of the *)User/Cause* parameter must indicate an SCCP MTP-User part and can be ignored by the receiver of the **DUPU** message.

3.5.6. Destination Restricted (DRST)

The **Destination Restricted (DRST)** message is sent from an SG to all concerned ASPs to indicate one of the following:

(1) A replicated subsystem is requesting that the TUA layer at the ASP accept transactions for the affected subsystem. The TUA layer at the ASP is expected to determine whether it can accept the traffic of the affected subsystem and respond with a **DRST** message.

(2) An SG representing a signalling transfer point is requesting that the TUA layer at the ASP routing message traffic via an alternate SG if possible.

The **DRST** is sent from an ASP to an SG in response to a **DRST** from the SG when the TUA layer at the ASP is prepared to accept traffic for the affected subsystem.

When the **DRST** message contains the *Subsystem Number* parameter, this message corresponds to the ITU[Q.711] and ANSI[T1.112] 'N-COORD' primitive. When the **DRST** message contains the *Subsystem Multiplicity Indicator* parameter, the message corresponds to the ‘Request’ and ‘Indication’ forms of the ‘N-COORD’ primitive; when it dos not include the parameter, it corresponds to the ‘Response’ and ‘Confirm’ forms of the ‘N-COORD’ primitive.

When the **DRST** message does not contain the *Subsystem Number* parameter, the message corresponds to the ITU[Q.704] and ANSI[T1.111] ‘Transfer Restricted’ message.

The *DRST* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Routing Context                         /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Affected Point Code                      /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Subsystem Number                       |
+-------------------------------+-------------------------------+
|         Tag = 0x041A          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                Subsystem Multiplicity Indicator               |
+-------------------------------+-------------------------------+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DRST* message can contain the following parameters:

Parameters		
Routing Context	Mandatory	
Affected Point Code	Mandatory	*1
Subsystem Number	Conditional	*2
Subsystem Multiplicity Indicator	Conditional	*3
Info String	Optional	

Note 1: The Affected Point Code refers to the node which has become restricted or which has requested coordinated service outage.

Note 2: The *Subsystem Number* parameter **SHALL** be present in the **SCON** message when requesting or responding to a subsystem coordinated service outage, and **SHALL NOT** be present when indicating the restriction of a signalling point.

Note 3: The *Subsystem Multiplicity Indicator* parameter **SHOULD NOT** be present in the **SCON** message when the *Subsystem Number* parameter is not also present. The *Subsystem Multiplicity Indicator* parameter **SHALL** be present in the **SCON** message when requesting or indicating a coordinated service outage, and **SHALL NOT** be present when responding to or confirming a coordinated service outage.

3.6. Application Server Process State Maintenance (ASPSM) Messages

3.6.1. ASP Up (UP)

The *ASP Up (UP)* message is used to indicate to a remote TUA peer that the Adaptation layer is up and running.

The *ASP UP* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0011          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         ASP Identifier                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASP UP* message can contain the following parameters:

Parameters		
ASP Identifier	Conditional	*1
Info String	Optional	

Note 1: ASP Identifier **MUST** be used where the IPSP/SGP cannot identify the ASP by pre-configured address/port number information (e.g, where an ASP is resident on a Host using dynamic address/port number assignment).

3.6.2. ASP Up Ack (UP ACK)

The *ASP Up Ack (UP ACK)* message is used to acknowledge an *ASP UP* message received from a remote TUA peer.

The *ASP UP ACK* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0004          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASP UP ACK* message can contain the following parameters:

Parameters	
Info String	Optional

3.6.3. ASP Down (DOWN)

The *ASP Down (DOWN)* message is used to indicate to a remote TUA peer that the adaptation layer is not running.

The *ASP DOWN* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0004          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASP DOWN* message can contain the following parameters:

Parameters	
Info String	Optional

3.6.4. ASP Down Ack (DOWN ACK)

The *ASP Down Ack (DOWN ACK)* message is used to acknowledge an *ASP DOWN* message received from a remote TUA peer.

The *ASP DOWN ACK* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASP DOWN ACK* message can contain the following parameters:

Parameters	
Info String	Optional

Note: The *ASP DOWN ACK* message will always be sent to acknowledge an *ASP DOWN* message.

3.6.5. Heartbeat (BEAT)

The *Heartbeat (BEAT)* message is optionally used to ensure that the TUA peers are still available to each other.

The *BEAT* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0009          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Heartbeat Data                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *BEAT* message can contain the following parameters:

Parameters	
Heartbeat Data	Optional

3.6.6. Heartbeat Ack (BEAT ACK)

The *Heartbeat ACK (BEAT ACK)* message is sent in response to a *BEAT* message. A peer **MUST** send a *BEAT ACK* in response to a *BEAT* message. It includes all the parameters of the received *BEAT* message, without any change.

The *BEAT ACK* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0009          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Heartbeat Data                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *BEAT ACK* message can contain the following parameters:

Parameters	
Heartbeat Data	Optional

3.7. Application Server Process Traffic Maintenance (ASPTM) Messages

3.7.1. ASP Active (ASPAC)

The *ASP Active (ASPAC)* message is sent by an ASP to indicate to a remote TUA peer that it is Active and ready to process signalling traffic for a particular Application Server.

The *ASPAC* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Routing Context                         /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x000B          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Traffic Mode Type                       |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASPAC* message can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
Traffic Mode Type	Optional	*2
Info String	Optional	

Note 1: When an ASP is registered or configured for multiple AS with an SG, the Routing Context associated with the AS whose activation is being requested **MUST** be placed in the *ASPAC* message.

Note 2: The Traffic Mode Type parameter is not necessary in the *ASPAC* message when both peers are aware of the traffic mode of the AS by configuration or registration.

3.7.2. ASP Active Ack (ASPAC ACK)

The *ASP Active Ack (ASPAC)* Ack message is used to acknowledge an *ASPAC* message received from a remote TUA peer.

The *ASPAC ACK* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Routing Context                         /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x000B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Traffic Mode Type                       |
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASPAC ACK* message can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
Traffic Mode Type	Optional	
Info String	Optional	

Note 1: When an ASP is registered or configured for multiple AS with an SG, the Routing Context associated with the AS whose activation is being acknowledged **MUST** be placed in the *ASPAC ACK* message.

3.7.3. ASP Inactive (ASPIA)

The *ASP Inactive (ASPIA)* message is sent by an ASP to indicate to a remote TUA peer that it is no longer processing signalling traffic within a particular Application Server.

The *ASPIA* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Routing Context                         /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          INFO String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ASPIA message can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
INFO String	Optional	

Note 1: When an ASP is registered or configured for multiple AS with an SG, the Routing Context associated with the AS whose deactivation is being requested **MUST** be placed in the *ASPIA* message.

3.7.4. ASP Inactive Ack (ASPIA ACK)

The *ASP Inactive Ack (ASPIA ACK)* message is used to acknowledge an *ASPIA* message received from a remote TUA peer.

The *ASPIA* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Routing Context                         /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          INFO String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASPIA* message can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
INFO String	Optional	

Note 1: When an ASP is registered or configured for multiple AS with an SG, the Routing Context associated with the AS whose deactivation is being acknowledged **MUST** be placed in the *ASPIA ACK* message.

3.8. Management (MGMT) Messages

3.8.1. Error (ERR)

The *Error (ERR)* message is used by a TUA peer to indicate an error situation. *ERR* messages **MUST NOT** be generated in response to other *ERR* messages.

The *ERR* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x000C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Error Code                           |
+-------------------------------+-------------------------------+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Routing Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0012          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Affected Point Code                     /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0419          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Subsystem Number                       |
+-------------------------------+-------------------------------+
|         Tag = 0x041D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
```

```
|                      Network Appearance                       |
+-------------------------------+-------------------------------+
|         Tag = 0x0007          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Diagnostic Info                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ERR* message can contain the following parameters:

Parameters		
Error Code	Mandatory	
Routing Context	Conditional	*1
Affected Point Code	Conditional	*2
Subsystem Number	Conditional	*3
Network Appearance	Conditional	*4
Diagnostic Info	Conditional	*5

Note 1: When the Error Code is "Invalid Routing Context," the Routing Context parameter **MUST** contain the invalid routing context value(s).

Note 2: When the Error Code is "Destination Status Unknown" or "Subsystem Status Unknown," the Affected Point Code parameter **MUST** contain the point codes for which status is unknown or unauthorized.

Note 3: When the Error Code is "Subsystem Status Unknown," the Subsystem Number parameter **MUST** contain the subsystem for which status is unknown or unauthorized.

Note 4: When the Error Code is "Invalid Network Appearance," the Network Appearance parameter **MUST** contains the invalid network appearance value.

Note 5: The Diagnostic Info parameter **SHOULD** contain the first 40 bytes of the message that caused the ERR message to be sent.

3.8.2. Notify (NTFY)

The Notify message is used to provide an autonomous indication of TUA events at an SG or IPSP to an ASP.

The *NTFY* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x000D          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                            Status                             |
+-------------------------------+-------------------------------+
|         Tag = 0x0011          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         ASP Identifier                        |
+-------------------------------+------------------------------
|         Tag = 0x0006          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Routing Context                          /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0004          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Info String                           /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *NTFY* message can contain the following parameters:

Parameters		
Status	Mandatory	
ASP Identifier	Conditional	*1
Routing Context	Conditional	*2
Info String	Optional	

Note 1: ASP Identifier **MUST** be used where the IPSP/SGP cannot identify the ASP by pre-configured address/port number information (e.g, where an ASP is resident on a Host using dynamic address/port number assignment) and the Status parameter is set to "Alternate ASP Active" or "ASP Failure".

Note 2: When an ASP is registered or configured for multiple AS with an SG, to identify the Application Server, the Routing Context associated with the AS whose state is being notified **MUST** be placed in the *NTFY* message when the Status parameter is set to "AS_State_Change".

3.9. Routing Key Management (RKM) Messages

Routing Key Management (RKM) messages are used to manage the Routing Keys that are used by an SG to direct traffic toward an Application Server.

3.9.1. Registration Request (REG REQ)

The *Registration Request (REG REQ)* message is sent by an ASP to indicate to a remote TUA peer that it wishes to register one or more given Routing Keys with the remote peer. Typically, an ASP would send this message to an SGP, and expects to receive a REG RSP message in return with an associated Routing Context value.

The *REG REQ* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Routing Key 1                         /
\                                                               \
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x041E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Routing Key n                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *REG REQ* message can contain the following parameters:

Parameters		
Routing Key	Mandatory	*1

Note 1: One or more Routing Key parameters **MAY** be included in a single *REG REQ* message. Whereas it is **OPTIONAL** for an implementation to be able to generate a *REG REQ* message with more than one Routing Key parameter, it is **REQUIRED** that the implementation be able to receive multiple Routing Key parameters in a single *REG REQ* message.

3.9.2. Registration Response (REG RSP)

The *Registration Response (REG RSP)* message is sent by an SG to an ASP to indicate the result of a previous *REG REQ* from an ASP. When successful, the *REG RSP* message contains the Routing Context assigned to the one or more Routing Keys that were presented in the *REG REQ message.*

The *REG RSP* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Registration Result 1                     /
\                                                               \
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x041F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Registration Result n                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *REG RSP* message can contain the following parameters:

Parameters		
Registration Result	Mandatory	*1

Note 1: *REG RSP* message. Whereas it is **OPTIONAL** for an implementation to be able to generate a *REG RSP* message with more than one Routing Key parameter, it is **REQUIRED** that the implementation be able to receive multiple Routing Key parameters in a single *REG RSP* message.

3.9.3. Deregistration Request (DEREG REQ)

The *Deregistration Request (DEREG REQ)* message is sent by an ASP to indicate to a remote TUA peer that it wishes to deregister a given Routing Key as identified by the given Routing Context. Typically, an ASP would send this message to an SGP, and expects to receive a *DEREG RSP* message in return with the associated Routing Context value.

The *DEREG REQ* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Routing Context                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DEREG REQ* message contains the following parameters:

Parameters		
Routing Context	Mandatory	*1

Note 1: One or more Routing Context values **MAY** be included in the Routing Context parameter. Whereas it is **OPTIONAL** for an implementation to be able to generate a *DEREG REQ* message with multiple Routing Context values in the Routing Context parameter, it is **REQUIRED** that an implementation be able to receive multiple

Routing Context values in the Routing Context parameter of the *DEREG REQ* message.

3.9.4. Deregistration Response (DEREG RSP)

The *Deregistration Response (DEREG RSP)* message is used as a response to the *DEREG REQ* message from a remote TUA peer.

The *DEREG REQ* message is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0420          |          Length = 12          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                    Deregistration Result 1                    |
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0420          |          Length = 12          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                    Deregistration Result n                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *DEREG REQ* message contains the following parameters:

Parameters		
Deregistration Result	Mandatory	*1

Note 1: One or more Deregistration Result parameters **MAY** be included in one *DEREG RSP* message. Whereas it is **OPTIONAL** for an implementation to be able to generate a *DEREG RSP* message with multiple Deregistration Result parameters, it is **REQUIRED** that an implementation be able to receive multiple Deregistration Result parameters in a single *DEREG RSP* message.

3.10. Common Parameters

These TLV parameters are common across the different adaptation layers.

Parameter Name	Parameter ID	Section
Reserved	0x0000	–
Not used in TUA	0x0001	–
Not used in TUA	0x0002	–
Not used in TUA	0x0003	–
Info String	0x0004	3.10.1
Not used in TUA	0x0005	–
Routing Context	0x0006	3.10.2
Diagnostic Info	0x0007	3.10.3
Not used in TUA	0x0008	–
Heartbeat Data	0x0009	3.10.4
Not used in TUA	0x000A	–
Traffic Mode Type	0x000B	3.10.5
Error Code	0x000C	3.10.6
Status	0x000D	3.10.7
Not used in TUA	0x000E	–
Not used in TUA	0x000F	–
Not used in TUA	0x0010	–
ASP Identifier	0x0011	3.10.8
Affected Point Code	0x0012	3.10.9

Correlation Id 0x0013 3.10.12

3.10.1. Info String

The *Info String* parameter is optionally included in all MGMT, ASPSM and ASPTM messages to provide additional debugging or diagnostic information.

The *Info String* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0004          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Info String                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Info String* parameter contains the following fields:

Info String field: variable (ASCII string)

The *Info String* field can carry any meaningful 8-bit ASCII character string along with the message. Length of the Info String field is from 0 to 255 characters. No procedures are presently identified for its use but implementations may use the Info String for debugging purposes.

3.10.2. Routing Context

The *Routing Context* parameter is included in all TUA SSNM, DH and CH messages as well as in MGMT, ASPTM, ASPSM that reference one or more Application Servers. The *Routing Context* parameter is used to uniquely identify an Application Server and Routing Key within an association between an SGP and ASP.

The *Routing Context* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0006          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Routing Context(s)                       /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Routing Context* parameter can contain the following fields:

Routing Context field: list of 32-bit (unsigned integer)

The *Routing Context* field contains (a list of) 32-bit unsigned integers indexing the Application Server traffic that the sending ASP is configured or registered to receive. There is 1:1 relationship between a Routing Context value, an SG Routing Key and an Application Server [5].

3.10.3. Diagnostic Information

The *Diagnostic Info* parameter is used in the MGMT Error (ERR) message to provide additional information concerning the message that generated an Error message reply. The *Diagnostic Info* parameter **SHOULD** contain the first 40 bytes of the message that generated the error.

The *Diagnostic Info* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0007          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Diagnostic Info                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Diagnostic Info* parameter contains the following fields:

Diagnostic Info field: variable length (bytes)

The *Diagnostic Info* field can contain any information germane to the error condition, to assist in the identification of the error condition. The Diagnostic Info **SHOULD** be the first 40 bytes of the offending message.

3.10.4. Heartbeat Data

The **Heartbeat Data** parameter is used in the BEAT and BEAT-Ack messages and contains whatever information the sender of the BEAT message chooses to include. Some uses for the Heartbeat Data parameter are described in Section 4.

The **Heartbeat Data** parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0009          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                        Heartbeat Data                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The **Heartbeat Data** parameter contains the following fields:

Heartbeat Data field: variable length (opaque)

The sending node defines the Heartbeat Data field contents. It may include a Heartbeat Sequence Number or Time-stamp, or other implementation specific details. The receiver of a Heartbeat message does not process this field as it is only of significance to the sender. The receiver **MUST** echo the content of the Heartbeat Data in a BEAT-Ack message. The data field can be used to store information in the heartbeat message useful to the sending node (e.g. the data field can contain a time stamp, a sequence number, etc.).

3.10.5. Traffic Mode Type

The *Traffic Mode Type* parameter indicates the fail-over and traffic distribution algorithm and procedures that will be used for an Application Server Process serving an Application Server. Each Application Server has associated with it only one Traffic Mode Type.

The *Traffic Mode Type* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x000B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Traffic Mode Type                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Traffic Mode Type* parameter contains the following fields:

Traffic Mode Type field: 32-bits (unsigned integer)

The Traffic Mode Type field identifies the traffic mode of operation of an ASP within an AS. The valid values for the Traffic Mode Type field are as follows:

1	Override
2	Loadshare
3	Broadcast

Within a Routing Context, Override, Load-share Types and Broadcast cannot be mixed. The Override value indicates that the ASP is operating in Override mode, and that when the ASP becomes active for the Application Server, it will take over all traffic for the AS (i.e, primary/back-up operation), overriding any currently active ASP in the AS. In Load-share mode, when the ASP becomes active for the AS, the ASP will share in the traffic distribution with any other active ASPs. In Broadcast mode, when the ASP becomes active for the AS, the ASP will receive the same traffic as any other active APSs.

3.10.6. Error Code

The *Error Code* parameter is used in the Error (ERR) message to indicate the reason that the ERR message was generated and, along with the other parameters in the Error message, help to locate the problem that generated the error condition.

The *Error Code* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x000C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Error Code                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Error Code* parameter contains the following fields:

Error Code field: 32-bit (unsigned integer)

The *Error Code* field indicates the reason for the Error Message. The Error Code field value can be one of the following values:

1	Invalid Version
3	Unsupported Message Class
4	Unsupported Message Type
5	Unsupported Traffic Handling Mode
6	Unexpected Message
7	Protocol Error
9	Invalid Stream Identifier
13	Refused - Management Blocking
14	ASP Identifier Required
15	Invalid ASP Identifier
17	Invalid Parameter Value
18	Parameter Field Error
19	Unexpected Parameter
20	Destination Status Unknown
21	Invalid Network Appearance
22	No configured AS for ASP
23	Invalid Routing Context
24	Subsystem Status Unknown

The "Invalid Version" error is sent if a message was received with an invalid or unsupported version. The Error message contains the supported version in the Common header. The Error message could optionally provide the supported version in the Diagnostic parameter.

The "Unsupported Message Class" error is sent if a message with an unexpected or unsupported Message Class is received.

The "Unsupported Message Type" error is sent if a message with an unexpected or unsupported Message Type is received.

The "Unsupported Traffic Handling Mode" error is sent by a SGP if an ASP sends an ASP Active message with an unsupported Traffic Mode Type or a Traffic Mode Type that is inconsistent with the presently configured mode for the Application Server. An example would be a case in which the SGP did not support load-sharing.

The "Unexpected Message" error **MAY** be sent if a defined and recognized message is received that is not expected in the current state (in some cases the ASP may optionally silently discard the message and not send an Error message). For example, silent discard is used by an ASP if it received a DATA message from an SGP while it was in the ASP-INACTIVE state. If the Unexpected message contained Routing Context(s), the Routing Context(s) **SHOULD** be included in the Error message.

The "Protocol Error" error is sent for any protocol anomaly (i.e., reception of a parameter that is syntactically correct but unexpected in the current situation.

The "Invalid Stream Identifier" error is sent if a message is received on an unexpected SCTP stream (e.g, a Management message was received on a stream other than "0", or a Data message was received on stream "0").

The "Refused - Management Blocking" error is sent when an ASP Up or ASP Active message is received and the request is refused for management reasons (e.g, management lockout"). If this error is in response to an ASP Active message, the Routing Context(s) in the ASP Active message **SHOULD** be included in the Error message.

The "ASP Identifier Required" is sent by a SGP in response to an ASP Up message which does not contain an ASP Identifier parameter when the SGP requires one. The ASP **SHOULD** resend the ASP Up message with an ASP Identifier.

The "Invalid ASP Identifier" is send by a SGP in response to an ASP Up message with an invalid (i.e., non-unique) ASP Identifier.

The "Invalid Parameter Value" error is sent if a message is received with an invalid parameter value (e.g, a DUPU message was received with a Mask value other than "0").

The "Parameter Field Error" would be sent if a message is received with a parameter having a wrong length field.

The "Unexpected Parameter" error would be sent if a message contains an invalid parameter.

The "Destination Status Unknown" Error **MAY** be sent if a DAUD is received at an SG inquiring of the availability or congestion status of a destination, and the SG does not wish to provide the status (e.g, the sender is not authorized to know the status). For this error, the invalid or unauthorized Point Code(s) **MUST** be included along with any Network Appearance or Routing Context associated with the Point Code(s) from the DAUD message.

The "Invalid Network Appearance" error is sent by a SGP if an ASP sends a message with an invalid (unconfigured) Network Appearance value. For this error, the invalid (unconfigured) Network Appearance **MUST** be included in the Network Appearance parameter in the Error message.

The "No Configured AS for ASP" error is sent if a message is received from a peer without a Routing Context parameter and it is not known by configuration data which Application Servers are referenced.

The "Invalid Routing Context" error is sent if a message is received from a peer with an invalid (unconfigured) Routing Context value, or if a message is received from a peer without a Routing Context parameter and it is not known by configuration data which Application Servers are referenced. For this error, the invalid Routing Context(s) **MUST** be included in the Error message.

The "Subsystem Status Unknown" Error **MAY** be sent if a DAUD is received at an SG inquiring of the availability or congestion status of a subsystem, and the SG does not wish to provide the status (e.g, the sender is not authorized to know the status). For this error, the invalid or unauthorized Point Code and Subsystem Number **MUST** be included along with any Network Appearance or Routing Context associated with the Point Code and Subsystem Number from the DAUD message.

3.10.7. Status

The Status parameter identifies the type of the status that is being notified in a Notify (NTFY) message and the Status ID.

The *Status* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x000D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|         Status Type           |          Status ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Status* parameter contains the following fields:

Status Type field: 16-bits (unsigned integer)

The valid values for Status Type field are as follows:

1 Application Server state change (AS_State_Change)
2 Other

Status ID field: 16-bits (unsigned integer)

The Status ID parameter contains more detailed information for the notification, based on the value of the Status Type.

(1) If the Status Type is "AS_State_Change", then the Status ID values are as follows:

1 reserved
2 Application Server Inactive (AS-Inactive)
3 Application Server Active (AS-Active)
4 Application Server Pending (AS-Pending)

These notifications are sent from an SGP to an ASP upon a change in status of a particular Application Server. The value reflects the new state of the Application Server.

(2) If the Status Type is "Other", then the following Status Information values are defined:

1 Insufficient ASP resources active in AS
2 Alternate ASP Active
3 ASP failure

These notifications are not based on the SGP reporting the state change of an ASP or AS. In the Insufficient ASP Resources case, the SGP is indicating to an "Inactive" ASP(s) in the AS that another ASP is required to handle the load of the AS (Load-sharing mode or Broadcast mode). For the Alternate ASP Active case, an ASP is informed when an alternate ASP transitions to the ASP-Active state in Override mode.

3.10.8. ASP Identifier

The *ASP Identifier* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0011          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         ASP Identifier                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ASP Identifier* parameter contains the following fields:

ASP Identifier field: 32-bits (unsigned integer)

The ASP Identifier field contains a unique value that is locally significant among the ASPs that support an AS. The SGP should save the ASP Identifier to be used, if necessary, with the Notify message (see Section 3.7.2).

The optional ASP Identifier parameter would contain a unique value that is locally significant among the ASPs that support an AS. The SGP should save the ASP Identifier to be used, if necessary, with the Notify message (see Section 3.3.3.2).

3.10.9. Affected Point Code

The *Affected Point Code* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0012          |            Length             |
+- - - - - - - -+- - - - - - - -+- - - - - - - - - - - - - - - -+
|     Mask      |              Affected Point Code 1            |
+- - - - - - - -+- - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                              ...                              /
\                                                               \
+- - - - - - - -+- - - - - - - - - - - - - - - - - - - - - - - -+
|     Mask      |              Affected Point Code n            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Affected Point Code* parameter contains the following fields:

Affected Destination Point Code field: n x 32-bits

The *Affected Point Code* parameter contains a list of one or more Affected Destination Point Code fields. It is **OPTIONAL** to generate an Affected Point Code parameter with more than one Affected Destination Point Code field, but it is **REQUIRED** to accept it.

Each Affected Destination Point Code field in the list contains the following fields:

Affected Point Code field: 24-bits (unsigned integer)

Each Affected Point Code field is a three-octet field to allow for up to 24-bit binary formatted SS7 Point Codes. Affected Point Codes that are less than 24-bits are padded on the left to the 24-bit boundary. The following examples show ANSI and ITU-T point codes:

ANSI 24-bit Point Code:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Mask      |    Network    |    Cluster    |     Member    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                |MSB-----------------------------------------LSB|
```

ITU-T 14-bit Point Code:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Mask      |0 0 0 0 0 0 0 0 0 0|Zone |      Region     | SP  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                   |MSB--------------------LSB|
```

Mask field: 8-bits (unsigned integer)

The Mask parameter can be used to identify a contiguous range of Affected Point Codes, independent of the point code format. Identifying a contiguous range of Affected Point Codes may be useful when a management event simultaneously affects the status of a series of destinations at an SG.

The Mask parameter is an integer representing a bit mask that can be applied to the related Affected PC field. The bit mask identifies how many bits of the Affected PC field are significant and which are effectively "wild-carded". For example, a mask of "8" indicates that the last eight bits of the PC is "wild-carded". For an ANSI 24-bit Affected PC, this is equivalent to signalling that all PCs in an ANSI Cluster are unavailable. A mask of "3" indicates that the last 3 bits of the PC is "wild-carded". For a 14-bit ITU Affected PC, this is equivalent to signalling that an ITU Region is unavailable.

A Mask value equal (or greater than) the number of bits in the Point Code indicates that the entire network access is affected: this is used to indicate network isolation to the ASP.

3.10.10. Correlation Id

The *Correlation Id* parameter is used to tag messages sent to an ASP in a Broadcast group as well as during fail-over.

The *Correlation Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0013          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Correlation Id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Correlation Id* parameter can contain the following fields:

Correlation Id field: 32-bits (unsigned integer)

The *Correlation Id* field contains a Correlation Id. The Correlation Id is a 32-bit identifier that is attached to the TUA Message Header to indicate to a newly entering ASP in a Broadcast AS where in the traffic flow of TUA messages the ASP is joining. It is attached to the TUA Message Header of the first DH or CH message sent to an ASP by an SG after sending an ASP Active Ack or otherwise starting traffic to an ASP. The Correlation Id is only significant within a Routing Context [6].

3.11. TUA-Specific parameters

These TLV parameters are specific to the TUA protocol.

Parameters used in DH Messages

Parameter Name	Parameter ID	Section
Dialogue Id	0x0401	3.11.1.1
Dialogue Flags	0x0402	3.11.1.2
Quality of Service	0x0403	3.11.1.3
Destination Address	0x0404	3.11.1.4
Originating Address	0x0405	3.11.1.5
Application Context Name	0x0406	3.11.1.6

User Information	0x0407	3.11.1.7
Security Context	0x0408	3.11.1.8
Confidentiality	0x0409	3.11.1.9
Termination	0x040A	3.11.1.10
Abort Cause	0x040B	3.11.1.11
Report Cause	0x040C	3.11.1.12
Abort Reason	0x040D	3.11.1.13
Components	0x040E	3.11.1.14
Component	0x040F	3.11.1.15
Transaction Id	0x0410	3.11.1.16

Parameters used in CH Messages

Parameter Name	Parameter ID	Section
Invoke Id	0x0411	3.11.2.1
Linked Id	0x0412	3.11.2.2
Component Flags	0x0413	3.11.2.3
Operation	0x0414	3.11.2.4
Parameters	0x0415	3.11.2.5
Error	0x0416	3.11.2.6
Problem Code	0x0417	3.11.2.7
Timeout	0x0418	3.11.2.8

Other Parameters

Parameter Name	Parameter ID	Section
Subsystem Number	0x0419	3.11.3.1
Subsystem Multiplicity Indicator	0x041A	3.11.3.2
Congestion Level	0x041B	3.11.3.3
User/Cause	0x041C	3.11.3.4
Network Appearance	0x041D	3.11.3.5
Routing Key	0x041E	3.11.3.6
Registration Result	0x041F	3.11.3.7
Deregistration Result	0x0420	3.11.3.8
Address Range	0x0421	3.11.3.9
Destination Transaction Id	0x0422	3.11.3.10
Originating Transaction Id	0x0423	3.11.3.11
Transaction Id Range	0x0424	3.11.3.12
Global Title	0x0425	3.11.3.13
Point Code	0x0426	3.11.3.14

3.11.1. Parameters used in DH Messages

3.11.1.1. Dialogue Id

The *Dialogue Id* parameter is used in the TUA Message Header to identify the dialogue within the Application Server indicated by the Routing Context (also in the TUA Message Header).

The *Dialogue Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0401          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Dialogue Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Dialogue Id* parameter contains the following fields:

Dialogue Id field: 32-bits (unsigned integer)

The *Dialogue Id* field contains an identifier that is used both at the SG and the ASP to identify a dialogue within an Application Server. The Dialogue Id value must be unique within the scope of a given Application Server and Routing Context.

For a given AS and Routing Context, either the SG or the ASP is responsible for assigning Dialogue Ids, but not both.

3.11.1.2. Dialogue Flags

The *Dialogue Flags* parameter is used in the DH Message Header and is used to indicate whether components are present (when the message is sent from SG to ASP) and whether permission is granted for the receiving TC-User to terminate the dialogue.

The *Dialogue Flags* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0402          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Dialogue Flags                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Dialogue Flags* parameter contains the following fields:

Dialogue Flags field: 32-bits (bit field)

The Dialogue Flags field contains flag bits used in to indicate additional characteristics of the DH message. The Dialogue Flags field is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        reserved                     |C|P| |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bits 0-28: Reserved (coded zero)

Reserved bits are reserved for later IETF extensions and are coded zero.

Bit 29: Components Present

The *Components Present* bit is set in the indication (i.e, sent from SG to ASP) forms of Dialogue Handling (DH) messages to indicate that Component Handling (CH) messages will follow containing the components associated with the Dialogue Handing message.

Bit 20: Permission

The *Permission* bit is cleared in Dialogue Handling (DH) messages to indicate that the remote TC-User is not permitted to end the dialogue.

Bit 31: Reserved (coded zero)

Reserved bits are reserved for later IETF extensions and are coded zero.

3.11.1.3. Quality of Service

The *Quality of Service* parameter contains the QoS parameters for the underlying SCCP Network Service.

The *Quality of Service* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0403          |           Length = 8          |
+- - - - - - - -+- - - - - - - -+- - - - - - - -+-+- - -+- - - -+
| Msg Priority  |  Importance   |  Seq Control  |R|  -  | P Cls |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Quality of Service* parameter contains the following fields:

Protocol Class field: 4-bits (unsigned integer)

The Protocol Class field indicates the SCCP Protocol Class requested by the TC-User for the current Dialogue Handling message. Valid values for the Protocol Class field are as follows:

0	SCCP Protocol Class 0	TCAP Operation Class 4
1	SCCP Protocol Class 1	TCAP Operation Class 1, 2, and 3
2	SCCP Protocol Class 2	TCAP Operation Class 1, 2, and 3
3	SCCP Protocol Class 3	TCAP Operation Class 1, 2, and 3

Spare field: 3-bits (coded zero)

Spare bits are coded zero.

Return Option field: 1-bit (boolean)

Specifies whether the SCCP "return message on error" is requested when the Protocol Class field is set to SCCP Protocol Class 0 or 1. When the Protocol Class field is set to SCCP Protocol Class 2 or 3, this field **MAY** be ignored by the SG. The Return Option field has the following values:

0 No "Return On Error" option requested.
1 "Return On Error" option requested.

Sequence Control field: 8-bits (unsigned integer)

When the Protocol Class field is other than SCCP Protocol Class 0, the *Sequence Control* field provides a sequence control parameter which is used by the underlying SS7 SCCP and MTP layer at the SG to generate an SLS value. When the Protocol Class field is set to Protocol Class 0, this field **SHOULD** be coded to zero and **MUST** be ignored by the SG.

Importance field: 8-bits (unsigned integer)

The Importance field contains the SCCP Importance level requested by the TC-User. Where the underlying SCCP transport at an SG does not support SCCP flow control[Q.714], this field **SHOULD** be coded to zero and **MUST** be ignored by the SG [7]. Valid values for the Importance field are as follows:

0 SCCP Importance Level 0 or Unspecified
1 SCCP Importance Level 1
2 SCCP Importance Level 2
3 SCCP Importance Level 3
4 SCCP Importance Level 4
5 SCCP Importance Level 5
6 SCCP Importance Level 6
7 SCCP Importance Level 7

Message Priority field: 8-bits (unsigned integer)

The Message Priority field contains the MTP Message Priority requested when the underlying SS7 transport at an SG supports multiple congestion levels[Q.704]. When the underlying transport does not support multiplex congestion levels or states, this field **SHOULD** be coded to zero and **MUST** be ignored by the SG [8]. Valid values for the Message Priority field are as follows:

0	Message Priority 0 or Unspecified
1	Message Priority 1
2	Message Priority 2
3	Message Priority 3

3.11.1.4. Destination Address

The *Destination Address* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0404          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Address parameter(s)                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Destination Address* parameter contains the following fields:

Address field: variable length (address parameter list)

The *Address* field contains a list of one or more address parameters. At least one address parameter **MUST** be present in the Address field. The *Address* field can contain the following parameters:

Parameters		
Point Code	Conditional	*1
Subsystem Number	Conditional	*1
Global Title	Optional	

Note :1 When the Address field contains a *Subsystem Number* parameter, it must also contain a *Point Code* parameter.

3.11.1.5. Originating Address

The *Originating Address* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0405          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Address parameter(s)                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Originating Address* parameter contains the following fields:

Address field: variable length (address parameter list)

The *Address* field contains a list of one or more address parameters. At least one address parameter **MUST** be present in the Address field. The *Address* field can contain the following parameters:

Parameters		
Point Code	Conditional	*1
Subsystem Number	Conditional	*1
Global Title	Optional	

Note :1 When the Address field contains a *Subsystem Number* parameter, it must also contain a *Point Code* parameter.

3.11.1.6. Application Context Name

The *Application Context Name* parameter contains the identifier of the application context proposed by the dialogue initiator or by the dialogue responder. An application context is an explicitly identified set of application-service-elements, related options and any other necessary information for the interworking of application-entities on a dialogue.

For a description of the Application Context Name parameter, see the ITU[Q.771] TCAP specifications.

The *Application Context Name* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0406          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                      Application Id Type                      |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                    Application Identifier                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Application Context Name* parameter contains the following fields:

Application Id Type field: 32-bits (unsigned integer)

The *Application Id Type* field indicates the type of Application Identifier that is present in the Application Identifier field. Valid values for the Application Id Type are as follows:

0 ASN.1 OBJECT IDENTIFIER
1 ASN.1 INTEGER

Application Identifier field: variable length (bytes)

The *Application Identifier* contains an identifier of the application context that is being proposed by the dialogue initiator or responder. When the Application Type is ‘0’ this field **MUST** be formatted as an **`OBJECT IDENTIFIER`**[X.680] representing the proposed Application Id. When the Application Type is ‘1’ this field **MUST** be formatted as 32-bit unsigned integer value representing the proposed Application Id.

3.11.1.7. User Information

The *User Information* parameter contains information which can be exchanged between TC-Users independently from the Remote Operation Service.

For a description of the User Information parameter, see the ITU[Q.771] TCAP specifications.

The *User Information* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0407          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         User Information                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *User Information* parameter can contain the following fields:

User Information field: variable length (bytes)

The internal format of the *User Information* field is opaque to TUA and to TCAP. The contents of this field is a string of bytes as they were provided to the TUA layer by the TC-User in a TC-BEGIN, TC-CONT, or TC-END primitive.

3.11.1.8. Security Context

The *Security Context* parameter contains the identifier of the security context proposed by the dialogue initiator or by the dialogue responder.

The *Security Context* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0408          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Security Type                         |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                       Security Identifier                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Security Context* parameter contains the following fields:

Security Id Type field: 32-bits (unsigned integer)

The *Security Id Type* field indicates the type of Security Identifier that is present in the Security Identifier field. Valid values for the Security Id Type are as follows:

0	ASN.1 OBJECT IDENTIFIER
1	ASN.1 INTEGER

Security Identifier field: variable length (bytes)

The *Security Identifier* contains an identifier of the application context that is being proposed by the dialogue initiator or responder. When the Security Type is '0' this field **MUST** be formatted as an **`OBJECT IDENTIFIER`**[X.680] representing the proposed Security Id. When the Security Type is '1' this field **MUST** be formatted as 32-bit unsigned integer value representing the proposed Security Id.

3.11.1.9. Confidentiality

Confidentiality Identifier is coded context specific (in the context of the dialogue portion sequence), constructor.

The *Confidentiality* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0409          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                   Confidentiality Id Type                     |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                  Confidentiality Identifier                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Confidentiality* parameter contains the following fields:

Confidentiality Id Type field: 32-bits (unsigned integer)

The *Confidentiality Id Type* field indicates the type of Confidentiality Identifier that is present in the Confidentiality Identifier field. Valid values for the Confidentiality Id Type are as follows:

0 ASN.1 OBJECT IDENTIFIER
1 ASN.1 INTEGER

Confidentiality Identifier field: variable length (bytes)

The *Confidentiality Identifier* contains an identifier of the application context that is being proposed by the dialogue initiator or responder. When the Confidentiality Type is '0,' this field **MUST** be formatted as an **`OBJECT IDENTIFIER`**[X.680] representing the proposed Confidentiality Id. When the Confidentiality Type is '1,' this field **MUST** be formatted as 32-bit unsigned integer value representing the proposed Confidentiality Id.

3.11.1.10. Termination

The *Termination* parameter indicates the dialogue termination scenario chosen by the TC-User (prearranged or basic).

The *Termination* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040A          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Termination                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Termination* parameter contains the following fields:

Termination field: 32-bit (unsigned integer)

The *Termination* field indicates the dialogue termination scenario chosen by the TC-User and can have one of the following values:

0 Prearranged
1 Basic

3.11.1.11. Abort Cause

The *Abort Cause* parameter is included in the *TUAB*, *TPAB*, *TUAB* and *TPAB* messages and indicates the reason for aborting the transaction or dialogue.

The *Abort Cause* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Abort Cause                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Abort Cause* parameter contains the following fields:

Abort Cause field: 32-bit (unsigned integer)

The *Abort Cause* field indicates the reason for aborting the dialogue and has a TCAP protocol-variant-specific value. Example values for ITU[Q.773] and ANSI[T1.114] are as follows:

	ITU-T Description	ANSI Description
0	unrecognized message type	–
1	unrecognized transaction id	unrecognized package type
2	badly formatted transaction portion	incorrect transaction portion
3	incorrect transaction portion	badly structured transaction portion
4	resource limitation	unassigned responding transaction identifier
5	L_RESOURCE_LIMIT	permission to release problem
6	invalid dialogue request	resource unavailable
7	pending expired	unrecognized dialogue portion identifier
8	begin expired	badly structured dialogue portion
9	inactive expired	missing dialogue portion
10	destination address unknown	inconsistent dialog portion
11	network error	–
12	unrecognized dialogue identifier	–
13	abnormal dialogue portion	–
14	no common dialogue portion	–

3.11.1.12. Report Cause

The *Report Cause* parameter indicates the reason for the sending of an TNOT message and reflects the SCCP reason that would be used for returning a TCAP message.

The *Report Cause* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Report Cause                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Report Cause* parameter contains the following fields:

Report Cause field: 32-bit (unsigned integer)

The *Report Cause* field indicates the reason that a TC-User message could not be delivered and has the following values:

0 no translation for an address of such nature
1 no translation for this specific address
2 subsystem congestion
3 subsystem failure
4 unequipped user
5 MTP failure
6 network congestion
7 SCCP unqualified
8 error in message transport
9 error in local processing
10 destination cannot perform re-assembly
11 SCCP failure
12 hop counter violation
13 segmentation not supported
14 segmentation failed.

3.11.1.13. Abort Reason

The *Abort Reason* parameter indicates whether a dialogue is aborted because the received application context name is not supported and no alternative one can be proposed or because of any other user problem.

The *Abort Reason* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040D          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Abort Reason                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Abort Reason* parameter contains the following fields:

Abort Reason field: 32-bits (unsigned integer)

The *Abort Reason* field indicates whether the dialogue was aborted because the received application context name is not supported and no alternative can be proposed or because of any other user problem. The valid values for *Abort Reason* are as follows:

0 application context not supported
1 user specific

3.11.1.14. Components

The *Components* parameter is used to attach components directly to a TUA Dialogue Handling (DH) message instead of in separate Component Handling (CH) massages.

The *Components* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040E          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|         Tag = 0x040F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Component #1                         /
\                                                               \
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                               .                               \
/                               .                               /
\                               .                               \
/                                                               /
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|         Tag = 0x040F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Component #n                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Components* parameter contains the following parameters:

Parameters		
Component	Conditional	*1

Note 1: The *Components* parameter **MUST** contain at least one *Component* parameter, but may contain more than one *Component* parameter.

3.11.1.15. Component

The *Component Type* field identifies the type of component (CINV, CRES, CCAN, etc.) that is contained within a *Component* parameter.

The *Component Type* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x040F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Component Type                        |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                      Component parameter(s)                   /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Component Type* parameter contains the following fields:

Component Type field: 32-bit (unsigned integer)

The *Component Type* field indicates the type of component contained in the component parameter. It can take on the following values: (Note that not all values are supported for interworking with all TCAP protocol variants.)

0 Invoke Last
1 Invoke Not Last
2 Result Last
3 Result Not Last
4 Error
5 Reject (User)

6 Reject (Local)
7 Reject (Remote)
8 Cancel

Component field: variable length (TLV parameter list)

The *Component* field contains the parameters associated with the component. This field may contains the following components, however, the formatting of the Component field **MUST** be the same as for the corresponding TUA message as follows:

Component Type		**CH Msg**	**Section**
0	Invoke Last	CINV	3.4.2
1	Invoke Not Last		
2	Result Last	CRES	3.4.3
3	Result Not Last		
4	Error	CERR	3.4.4
5	Reject (User)	CREJ	3.4.5
6	Reject (Local)		
7	Reject (Remote)		
8	Cancel	CCAN	3.4.6

3.11.1.16. Transaction Id

The *Transaction Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0410          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Transaction Id                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Transaction Id* parameter contains the following fields:

Transaction Id field: 32-bits (unsigned integer)

The *Transaction Id* field contains the value of the originating or terminating transaction identifier.

3.11.2. Parameters used in CH Messages

3.11.2.1. Invoke Id

The *Invoke Id* parameter identifies an invoke component. This identifier is only significant within the scope of a transaction and need only uniquely identify a dialogue within a transaction in a given direction (e.g, from SGP to ASP). The value of the *Invoke Id* parameter is chosen by the TUA peer sending the Invoke. As both the ASP and SGP could be assigning the same values of *Invoke Id* to invocations in each direction, the *Invoke Id* need only be unique in one direction.

The *Invoke Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0411          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Invoke Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Invoke Id* parameter contains the following fields:

Invoke Id field: 32-bit (unsigned integer)

The *Invoke Id* field contains the value of the invoke identifier for the current component.

3.11.2.2. Linked Id

The *Linked Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0412          |             Length = 8        |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Linked Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Linked Id* parameter contains the following fields:

Linked Id field: 32-bit (unsigned integer)

The *Linked Id* field contains the value of the linked or correlation invoke identifier which is related to the current component.

3.11.2.3. Component Flags

The *Component Flags* parameter is used in the *CINV* and *CRES* CH messages to indicate whether the contained components are segmented and whether they represent the last segment in a sequence of component segments.

The *Component Flags* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0413          |             Length = 8        |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          unused                             |N|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Component Flags* parameter contains the following fields:

Component Flags field: 32-bits

The Component Flags field is used to convey information about the components in a Component Handling (CH) message. It contains the following bit fields:

Bits 0-30: Unused

These bits are reserved and are coded to zero.

Bit 31: Not Last Bit

The Not Last bit is used to indicate whether the component present in the CH message is the last component of a sequence of segmented components. It has the following values:

0 Last component in a component sequence.
1 Not the last component in a component sequence.

To smoothly interwork with TCAP, TUA includes a mechanism whereby components can be segmented: the CH message with the "Not Last" bit set in the Component Flags field provides for the initial segments of a segmented component, whereas the CH message with the "Not Last" bit clear in the Component Flags field provides for the final (or only) segment in a sequence of component segments representing the complete component. When interworking with TCAP, each component segment may be sent in a different TCAP package[Q.775].

3.11.2.4. Operation

The *Operation* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0414          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Operation Class                        |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                        Operation Type                         |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                        Operation Code                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Operation* parameter contains the following fields:

Operation Class field: 32-bit (unsigned integer)

The *Operation Class* field indicates the operational class of the invoke in which it appears and has the following values:

0		not specified
1	Class 1	both success and failure are reported
2	Class 2	only failure is reported
3	Class 3	only success is reported
4	Class 4	neither success, nor failure is reported

Operation Type field: 32-bit (unsigned integer)

The *Operation Type* field indicates the type of operation code and has the following values:

1	National TCAP Operation	INTEGER
2	Private TCAP Operation	INTEGER
3	Local TCAP Operation	INTEGER
4	Global TCAP Operation	OBJECT IDENTIFIER

Operation Code field: variable length (based on type)

The *Operation Code* field contains an identifier of the requested operation. When the Operation Type is "National," "Private," or "Local," this field **MUST** be formatted as 32-bit unsigned integer value representing the requested operation. When the Operation Type is "Global," this field **MUST** be formatted as an **`OBJECT IDENTIFIER`**[X.680]. representing the requested operation. The value of this field is TCAP protocol-variant-specific.

3.11.2.5. Parameters

The *Parameters* parameter identifies the parameter set or parameter sequence that accompanies an operation invocation or response.

The *Parameters* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0415          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                           Parameters                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Parameters* parameter contains the following fields:

Parameters field: variable length (bytes)

The *Parameters* field contains all of the parameters coded according to the coding[X.680] for Parameter Sequences or Parameter Sets per the applicable TCAP protocol specification. For example, ITU[Q.773] or ANSI[T1.114]. [9]

3.11.2.6. Error

The *Error* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0416          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Error Type                          |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                           Error Code                          /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Error* parameter contains the following fields:

Error Type field: 32-bit (unsigned integer)

The *Error Type* field indicates the level (i.e, local or global) at which the error was generated. It has the following values:

1	National TCAP Error	INTEGER
2	Private TCAP Error	INTEGER
3	Local TCAP Error	INTEGER
4	Global TCAP Error	OBJECT IDENTIFIER

Error Code field: variable length (based on type)

The *Error Code* field contains an identifier of the indicated error. When the Error Type is "National," "Private," or "Local," this field **MUST** be formatted as a 32-bit signed integer value representing the indicated error. When the Error Type is "Global," this field **MUST** be formatted as an **`OBJECT IDENTIFIER`**[X.680] representing the indicated error. The value of this field is TCAP protocol-variant-specific.

3.11.2.7. Problem Code

The *Problem Code* parameters identifies the reason for rejecting a component.

The *Problem Code* parameters is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0417          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Problem Type                         |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                          Problem Code                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Problem Code* parameters contains the following fields:

Problem Type field: 32-bit (unsigned integer)

The *Problem Type* field indicates the reason for rejecting a component and has the following values: (Note that not all problem type field values are applicable to all TCAP protocol variants.)

0 General Problem
1 Problem with Invoke
2 Problem with Return Result
3 Problem with Return Error
4 Problem with Transaction Portion (deprecated)

Problem Code field: variable length (signed integer)

The *Problem Code* field indicates the specific problem associated with the Problem Type. For more information on problem codes, see Q.773 Chapter 4.2.2.6 and ANSI T1.114.3 Chapter 5.16.2.

Problem Code field: 32-bit (signed integer)

The *Problem Code* field indicates the specific problem associated with the Problem Type. This is a TCAP protocol-variant-specific value. Following are some example values for ITU[Q.773] and ANSI[T1.114]:

		ITU	ANSI
General	0	unrecognized component	–
Problem	1	mis-typed component	unrecognized component type
	2	badly structured component	incorrect component portion
	3	–	badly structured component portion
Invoke	0	duplicate invoke id	–
Problem	1	unrecognized operation	duplicate invocation
	2	mis-typed parameter	unrecognized operation
	3	resource limitation	incorrect parameter
	4	initiating release	unrecognized correlation id
	5	unrecognized linked id	–
	6	linked response unexpected	–
	7	unexpected linked operation	–
Return	0	unrecognized invoke id	–
Result	1	return result unexpected	unrecognized correlation id
Problem	2	mis-typed parameter	unexpected return result
	2	–	incorrect parameter
Return	0	unrecognized invoke id	–
Error	1	return error unexpected	unexpected return error
Problem	2	unrecognized error	unrecognized error
	3	unexpected error	unexpected error
	4	mis-typed parameter	incorrect parameter
Trans	1	–	unrecognized package type

Portion	2 –	incorrect transaction portion
Problem	3 –	badly structured transaction portion
(depr.)	4 –	unassigned responding transaction id
	5 –	permission to release problem
	6 –	resource unavailable

3.11.2.8. Timeout

The *Timeout* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0418          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                            Timeout                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Timeout* parameter contains the following fields:

Timeout field: 32-bit (unsigned integer)

The *Timeout* field contains the timeout value in seconds that the sender will wait before an invocation is canceled.

3.11.3. Other Parameters

3.11.3.1. Subsystem Number

The *Subsystem Number* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0419          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                   Reserved                    |      SSN      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Subsystem Number* parameter contains the following fields:

Reserved field: 24-bits (coded zero)

Reserved bits are coded zero.

SSN field: 8-bits (unsigned integer)

The *SSN* field contains the SCCP subsystem number[Q.713, T1.112].

3.11.3.2. Subsystem Multiplicity Indicator

The *Subsystem Multiplicity Indicator* is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041A          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                   Reserved                    |      SMI      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Subsystem Multiplicity Indicator* contains the following fields:

Reserved field: 24-bits (coded zero)

Reserved bits are coded zero.

SMI field: 8-bits (unsigned integer)

The *SMI* field contains the SCCP subsystem multiplicity indicator. Valid values for the *SMI* field are as follows:

0	Reserved/Unknown
1	Solitary
2	Duplicated
3	Triplicated
4	Quadruplicated
...	...
255	Unspecified

3.11.3.3. Congestion Level

The *Congestion Level* parameter is used to indicate the MTP network congestion level or SCCP restricted importance level and is used in the Network Congestion (SCON) message.

The *Congestion Level* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Congestion Level                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Congestion Level* parameter contains the following fields:

Congestion Level field: 32-bits (unsigned integer)

The Congestion Level field contains the level at which congestion has occurred.

When the Congestion Level parameter is included in a SCON message that corresponds to an N-PCSTATE request indication primitive, the Congestion Level field indicates the MTP congestion level experienced by the local or affected signalling point as indicated by the Affected Point Code(s) also in the SCON message. In this case, valid values for the Congestion Level field are as follows:

0	No Congestion or Undefined
1	Congestion Level 1
2	Congestion Level 2
3	Congestion Level 3

When the Congestion Level parameter is included in a SCON message that corresponds to an N-STATE request or indication primitive, the Congestion Level field indicates the SCCP restricted importance level experienced by the local or affected subsystem as indicated by the Affected Point Code and Subsystem Number also in the SCON message. In this case, valid values for the Congestion Level field range from 0 to 7, where 0 indicates the least congested and 7 indicates the most congested subsystem.

3.11.3.4. User/Cause

The *User/Cause* parameter is used to report the affected user and the cause of the unavailability of the user in a *DUPU* message.

The *User/Cause* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|             Cause             |             User              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *User/Cause* parameter contains the following fields:

Cause field: 16-bits (unsigned integer)

The *Cause* field indicates the cause of the unavailability of the remote user. Valid *Cause* values are as follows:

0 Unknown
1 Unequipped Remote User
2 Inaccessible Remote User

User field: 16-bits (unsigned integer)

The *User* field contains the SI value of the MTP User[Q.704] that is being reported unavailable. For TUA, this is the SI value of the SCCP (normally SI = 3). The TC-User **MAY** ignore the *User* field.

3.11.3.5. Network Appearance

The *Network Appearance* parameter is used as a parameter in the Registration Request (REG REQ) message to indicate the network context in which the remainder of the Routing Key parameters are to be interpreted. The *Network Appearance* parameter is also used in the Error (ERR) message in response to a REG REQ message when a received Network Appearance parameter contains an invalid value.

The *Network Appearance* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                      Network Appearance                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Network Appearance* parameter can contain the following fields:

Network Appearance field: 32-bits (unsigned integer)

The Network Appearance field identifies the SS7 network context for the Routing Key. The Network Appearance value is of local significance only, coordinated between the SG and ASP. Therefore, in the case where the ASP is connected to more than one SG, the same SS7 Network context may be identified by a different Network Appearance value depending upon to which SG the ASP is registering.

In the Routing Key, the Network Appearance identifies the SS7 Point Code and Global Title Transaction Type format used, and the SCCP, TCAP and TC-User protocol (type, variant and version) used within the specific SS7 network.

3.11.3.6. Routing Key

The *Routing Key* parameter is used in the *REG REQ* message to list and identify the Routing Keys that are being registered.

The *Routing Key* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041E          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                 Local Routing Key Identifier                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                       Key parameter(s)                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Routing Key* parameter can contain the following fields:

Local Routing Key Identifier field: 32-bits (unsigned integer)

The *Local Routing Key Identifier* field is used to uniquely identify the registration request. The identifier value is assigned by the ASP and is used to correlate the response in a REG RSP message with the original registration request. The identifier value must remain unique until the REG RSP (or ERR) message is received.

Key field: variable (TLV parameters)

The key field can contain the following parameters:

Parameters		
Network Appearance	Conditional	*1
Traffic Mode Type	Optional	
Originating Address	Optional	
Destination Address	Optional	
Address Range	Optional	
Originating Transaction Id	Optional	
Destination Transaction Id	Optional	
Transaction Id Range	Optional	
Application Context Name	Optional	
User Information	Optional	

Note 1: The Network Appearance parameter **MUST** be included in the Routing Key when the ASP is able to register in multiple SS7 Network contexts.

3.11.3.7. Registration Result

The *Registration Result* parameter is used to identify and report the result of the registration request for each Routing Key that was requested registered in a *REG REQ* message.

The *Registration Result* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x041F          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                 Local Routing Key Identifier                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                      Registration Status                      |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                        Routing Context                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Registration Result* parameter can contain the following fields:

Local Routing Key Identifier field: 32-bit (unsigned integer)

The *Local Routing Key Identifier* field contains the same value as found in the matching Routing Key parameter in the *REG REQ* message.

Registration Status field: 32-bits (unsigned integer)

The *Registration Status* field indicates the success or the reason for failure of a registration request.

Its values may be:

0	Successfully Registered
1	Error - Unknown
2	Error - Invalid Destination Address
3	Error - Invalid Network Appearance
4	Error - Invalid Routing Key
5	Error - Permission Denied
6	Error - Cannot Support Unique Routing
7	Error - Routing Key not Currently Provisioned
8	Error - Insufficient Resources
9	Error - Unsupported RK parameter field
10	Error - Unsupported/Invalid Traffic Mode Type

Routing Context field: 32-bits (unsigned integer)

The *Routing Context* field contains the Routing Context value for the associated Routing Key if the registration was successful. It is set to "0" if the registration was not successful.

3.11.3.8. Deregistration Result

The *Deregistration Result* parameter is used to identify and report the result of the deregistration request for each Routing Context that was requested deregistered and a *DEREG REQ* message.

The *Deregistration Result* parameter contains the deregistration status for a single Routing Context in a DEREG REQ message. The number of results in a DEREG RSP message **MAY** by anywhere from one to the total number of Routing Context values found in the corresponding REG REQ message. Where multiple DEREG RSP messages are used in reply to a single DEREG REQ message, a specific result **SHOULD** be in only one DEREG RSP message.

The *Deregistration Result* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Tag = 0x0420         |          Length = 12          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Routing Context                        |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
|                     Deregistration Status                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Deregistration Result* parameter can contain the following fields:

Routing Context field: 32-bits (unsigned integer)

The *Routing Context* field contains the Routing Context value of the matching Routing Key to deregister, as found in the *DEREG REQ* message.

Deregistration Status field: 32-bits (unsigned integer)

The *Deregistration Status* field indicates the success or the reason for failure of the deregistration.

Its values may be:

0 Successfully Deregistered
1 Error - Unknown
2 Error - Invalid Routing Context
3 Error - Permission Denied
4 Error - Not Registered
5 Error - ASP Currently Active for Routing Context

3.11.3.9. Address Range

The *Address Range* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0421          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Address Parameter(s)                    /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Address Range* parameter can contain the following fields:

Address field: variable (TLV parameters)

The *Address* field can contain the following parameters:

Parameters		
Originating Address	Conditional	*1
Destination Address	Conditional	*1

Note 1: The Address field must contain pairs of Originating Addresses or Destination Addresses and **MUST** contain one and only one pair of addresses; but, **MUST NOT** mix Originating Addresses with Destination Addresses in the same Address field.

3.11.3.10. Destination Transaction Id

The *Destination Transaction Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0422          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                   Destination Transaction Id                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Destination Transaction Id* parameter can contain the following fields:

Destination Transaction Id field: 32-bits (unsigned integer)

The *Destination Transaction Id* field contains the Destination Transaction Identifier associated with the dialogue.

3.11.3.11. Originating Transaction Id

The *Originating Transaction Id* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0423          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                    Originating Transaction Id                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Originating Transaction Id* parameter can contain the following fields:

Originating Transaction Id field: 32-bits (unsigned integer)

The *Originating Transaction Id* field contains the Originating Transaction Identifier associated with the dialogue.

3.11.3.12. Transaction Id Range

The *Transaction Id Range* parameter is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0424          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Transaction Id Parameter(s)                /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Transaction Id Range* parameter can contains the following fields:

Transaction Id field: list of 32-bit (unsigned integer)

The *Transaction Id* field can contain the following parameters:

Parameters		
Originating Transaction Id	Optional	*1
Destination Transaction Id	Optional	*1

Note 1: The Transaction Id field must contain pairs of Originating Transaction Ids or Destination Transaction Ids and **MUST** contain one and only one pair of Transaction Id parameters; but, **MUST NOT** mix Originating Transaction Ids with Destination Transaction Ids in the same Transaction Id field.

3.11.3.13. Global Title

The *Global Title* parameters is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0425          |            Length             |
+- - - - - - - -+- - - - - - - -+- - - -+- - - -+- - - - - - - -+
| Num. Digits   |  Trans. Type  | N Plan| E Sch | Nature of Add |
+- - - - - - - -+- - - - - - - -+- - - -+- - - -+- - - - - - - -+
\                                                               \
/                      Global Title Address                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Global Title* parameters contains the following fields:

Number of Digits field: 8-bits (unsigned integer)

The *Number of Digits* field contains the number of address signals that are represented in the *Global Title Address* field.

Translation Type field: 8-bits (unsigned integer)

The *Translation Type* field contains the translation type to be performed on the address information in the *Global Title* parameter. This is a TCAP protocol-variant-specific value. Example valid values for ITU[Q.713] are as follows:

0	unknown
1 – 63	international services
128 – 254	national network specific

Numbering Plan field: 4-bits (unsigned integer)

The *Numbering Plan* field contains the numbering plan to which the address information contained in the *Global Title Address* field belongs. This is a TCAP protocol-variant-specific value. Example valid values for ITU[Q.713] are as follows:

0	unknown
1	ISDN/telephony numbering plan (E.163 and E.164)
2	generic numbering plan
3	data numbering plan (X.121)
4	telex numbering plan (F.69)
5	maritime mobile numbering plan (E.210, E.211)
6	land mobile numbering plan (E.212)
7	ISDN/mobile numbering plan (E.214)
14	private network or network-specific numbering plan

Encoding Scheme field: 4-bits (unsigned integer)

The *Encoding Scheme* field contains the format for the address information contained in the *Global Title Address* field. This is a TCAP protocol-variant-specific value. Example valid values for ITU[Q.713] are as follows:

0	unknown
1	BCD, odd number of digits
2	BCD, even number of digits
3	national specific

Nature of Address field: 8-bits (unsigned integer)

The *Nature of Address* field contains an indication of the nature of the information represented in the *Global Title Address* field. This is a TCAP protocol-variant-specific value. Example valid values for ITU[Q.713] are as follows:

0	unknown
1	subscriber number
2	reserved for national use
3	national significant number
4	international number

Global Title Address field: variable length (bytes)

The *Global Title Digits* field contains the global title address information. This information is formatted according to the Encoding Scheme, belongs to the Numbering Plan, has the Nature of Address, and contains the Number of Digits. When the encoding scheme is BCD, the Global Title Digits field is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Dig 2 | Dig 1 | Dig 4 | Dig 3 | Dig 6 | Dig 5 | Dig 8 | Dig 7 |
+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+
| Dig 10| Dig 9 |  ...  |  ...  |  ...  |  ...  |  ...  |  ...  |
+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+
\                               .                               \
/                               .                               /
\                               .                               \
/                                                               /
+- - - - - - - -+- - - -+- - - -+- - - -+- - - -+- - - -+- - - -+
|               |filler | Dig n |  ...  |  ...  |  ...  |  ...  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where each digit is coded as follows:

0x0	digit 0
0x1	digit 1
0x2	digit 2
0x3	digit 3
0x4	digit 4
0x5	digit 5
0x6	digit 6
0x7	digit 7
0x8	digit 8
0x9	digit 9
0xA	spare
0xB	code 11
0xC	code 12
0xD	spare
0xE	spare
0xF	ST

When the Encoding Scheme is not "BCD," both the TUA layer at the ASP and the TUA layer at the SG should treat the Global Title Address as opaque.

3.11.3.14. Point Code

The *Point Code* parameters is formatted as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Tag = 0x0426          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Point Code                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Point Code* parameters contains the following fields:

Point Code field: 32-bits (unsigned integer)

The *Point Code* field contains an SS7 signalling point code. Point codes that are less than 32-bits are padded on the left to the 32-bit boundary. The following examples show ANSI and ITU-T point codes:

ANSI 24-bit Point Code:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0|    Network    |    Cluster    |     Member    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                |MSB----------------------------------------LSB|
```

ITU-T 14-bit Point Code:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|Zone |      Region     | SP  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                    |MSB--------------------LSB|
```

4. Procedures

The TUA layer needs to respond to various local primitives it receives from other layers as well as the messages that it receives from the peer TUA layer. This section describes the TUA procedures in response to these events.

4.1. Procedures to Support the TC-User

4.1.1. Receipt of Primitives from the TC-User

Upon receiving a TC request or response primitive from the upper layer at an ASP or IPSP, the TUA layer sends a corresponding TUA Dialogue Handling (DH) or Component Handling (CH) message (see Section 3) to its TUA peer. The TUA peer receiving the DH or CH message delivers the corresponding TC primitive to the TC-User at the IPSP or Nodal Interworking Function at the SG as illustrated in *Figure 4*. The mapping of TC primitives to TUA DH Messages is listed in *Table 2*, and the CH Messages in *Table 3* (see Section 1.6.1).

4.1.2. Receipt of Primitives from TCAP

Upon receiving a TC indication or confirmation primitive from TCAP at an SG, the Nodal Interworking Function passes the primitive to TUA. The TUA layer sends a corresponding TUA Dialogue Handling (DH) or Component Handling (CH) message (see Section 3) to its TUA peer at the ASP.

The TUA peer receiving the DH or CH message delivers the corresponding TC primitive to the TC-User at the ASP as illustrated in *Figure 5*. The mapping of TC primitives to TUA DH Messages is listed in *Table 2*, and the CH Messages in *Table 3* (see Section 1.6.1).

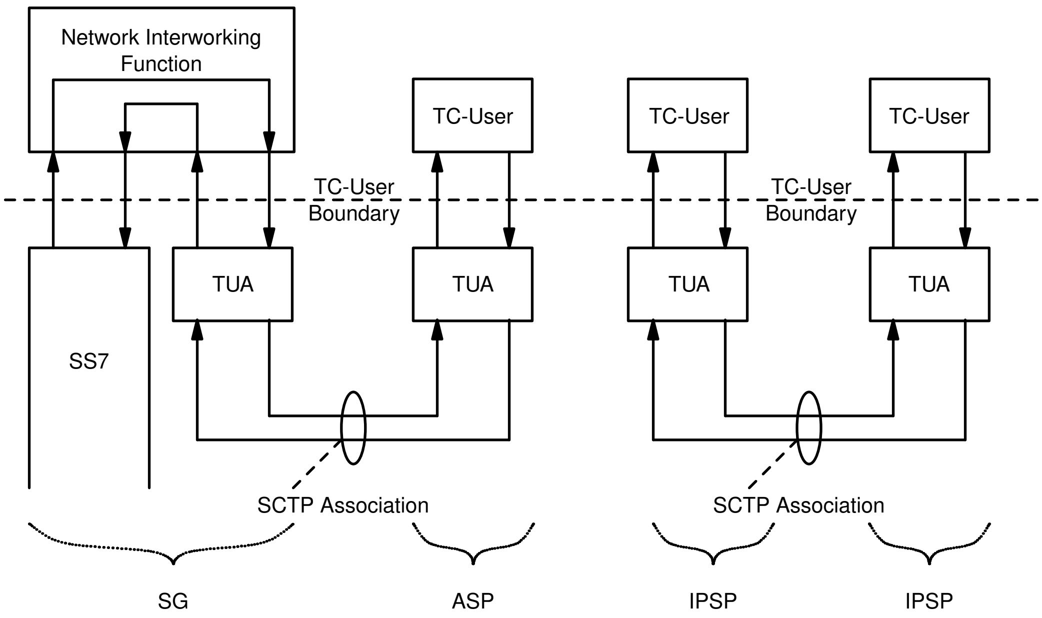


Figure 4. TUA Layer Model

The TUA Transaction Mapping Function (see Section)

For TC-BEGIN indications, the TUA Transaction Mapping Function (TMF) determines the Application Server (AS) based on comparing the address and dialogue portion information in the primitive with a provisioned Routing Key.

From the list of ASPs within an AS table, an ASP in the ASP-ACTIVE state is selected and a TQRY message is constructed and issued on the corresponding SCTP association. The TUA at the SG is also responsible for assigning and managing a Dialogue Identifier which is sent to the ASP in the TQRY message to identify the newly created dialogue to the ASP. Information associated with the dialogue is stored in the SG in an implementation dependent manner; however, the SG must be capable of associating further TUA messages with the correct Dialogue at the SG. The SG will have to access this stored information to continue processing the dialogue.

The TUA Transaction Mapping Function (TMF) determines the Application Server (AS) based on comparing the information in the primitive with a provisioned Routing Key.

4.1.2.1. Receipt of Management Primitives from TCAP

When TCAP Management indications are received (N-STATE, N-PCSTATE, N-COORD), TCAP Management determines whether there are concerned local TC-Users. When these local TC-Users are in fact Application Servers, serviced by ASPs, TUA management is transparently informed with the N-STATE, N-PCSTATE, N-COORD indication primitive upon which it formats and transfers the applicable SSNM message (DUNA, DAVA, DRST, DUPU or SCON) to the list of concerned ASPs.

The TUA message distribution function determines the Application Server (AS) based on comparing the information in the TC-BEGIN, TC-CONTINUE, TC-END, or TC-ABORT primitive with a provisioned Routing Key.

From the list of ASPs within the AS table, an ASP in the ASP-ACTIVE state is selected and Dialogue Handling (DH) and Component Handling (CH) messages are constructed and issued on the corresponding SCTP association. If more than one ASP is in the ASP-ACTIVE state (i.e., traffic is to be load-shared across more than one ASP), one of the ASPs in the ASP-ACTIVE state is selected from the list. (If the ASPs are in Broadcast Mode, all active ASPs will be selected and the message sent to each of the active ASPs.) The selection algorithm is implementation dependent but could, for example, be round robin or based on the SLS. The appropriate selection algorithm must be chosen carefully as it is dependent on application assumptions and understanding of the degree of state coordination between the ASP-ACTIVE ASPs in the AS.

In addition, the message needs to be sent on the appropriate SCTP stream, again taking care to meet the message sequencing needs of the signalling application. Dialogue Handling (DH) and Component Handling (CH) messages **SHOULD** be sent on an SCTP stream other than stream ‘0’.

When there is no Routing Key match, or only a partial match, for an incoming SS7 message, a default treatment **MAY** be specified. Possible solutions are to provide a default Application Server at the SGP that directs all unallocated traffic to a (set of) default ASP(s), or to drop the message and provide a notification to Layer Management in an M-ERROR indication primitive. The treatment of unallocated traffic is implementation dependent.

4.1.3. Receipt of Primitive from the Layer Management

On receiving primitives from the local Layer Management, the TUA layer will take the requested action and provide an appropriate response primitive to Layer Management.

An M-SCTP_ESTABLISH request primitive from Layer Management at an ASP or IPSP will initiate the establishment of an SCTP association. The TUA layer will attempt to establish an SCTP association with the remote TUA peer by sending an SCTP-ASSOCIATE primitive to the local SCTP layer.

When an SCTP association has been successfully established, the SCTP will send an SCTP-COMMUNICATION_UP notification primitive to the local TUA layer. At the SGP or IPSP that initiated the request, the TUA layer will send an M-SCTP_ESTABLISH confirm primitive to Layer Management when the association setup is complete. At the peer TUA layer, an M-SCTP_ESTABLISH indication primitive is sent to Layer Management upon successful completion of an incoming SCTP association setup.

An M-SCTP_RELEASE request primitive from Layer Management initiates the shutdown of an SCTP association. The TUA layer accomplishes a graceful shutdown of the SCTP association by sending an SCTP-SHUTDOWN primitive to the SCTP layer.

When the graceful shutdown of the SCTP association has been accomplished, the SCTP layer returns an SCTP-SHUTDOWN_COMPLETE notification primitive to the local TUA layer. At the TUA Layer that initiated the request, the TUA layer will send an M-SCTP_RELEASE confirm primitive to Layer Management when the association shutdown is complete. At the peer TUA Layer, an M-SCTP_RELEASE indication primitive is sent to Layer Management upon abort or successful shutdown of an SCTP association.

An M-SCTP_STATUS request primitive supports a Layer Management query of the local status of a particular SCTP association. The TUA layer simply maps the M-SCTP_STATUS request primitive to an SCTP-STATUS primitive to the SCTP layer. When the SCTP responds, the TUA layer maps the association status information to an M-SCTP_STATUS confirm primitive. No peer protocol is invoked.

Similar LM-to-TUA-to-SCTP and SCTP-to-TUA-to-LM primitive mappings can be described for the various other SCTP Upper Layer primitives in RFC 2960 [2960] such as INITIALIZE, SET PRIMARY, CHANGE HEARTBEAT, REQUEST HEARTBEAT, GET SRTT REPORT, SET FAILURE THRESHOLD, SET PROTOCOL PARAMETERS, DESTROY SCTP INSTANCE, SEND FAILURE, AND NETWORK STATUS CHANGE. Alternatively, these SCTP Upper Layer primitives (and Status as well) can be considered for modeling purposes as a Layer Management interaction directly with the SCTP Layer.

M-NOTIFY indication and M-ERROR indication primitives indicate to Layer Management the notification or error information contained in a received TUA Notify or Error message respectively. These indications can also be generated based on local TUA events.

An M-ASP_STATUS request primitive supports a Layer Management query of the status of a particular local or remote ASP. The TUA layer responds with the status in an M-ASP_STATUS confirm primitive. No TUA peer protocol is invoked. An M-AS_STATUS request supports a Layer Management query of the status of a particular AS. The TUA responds with an M-AS_STATUS confirm primitive. No TUA peer protocol is invoked.

M-ASP_UP request, M-ASP_DOWN request, M-ASP_ACTIVE request and M- ASP_INACTIVE request primitives allow Layer Management at an ASP to initiate state changes. Upon successful completion, a corresponding confirm primitive is provided by the TUA layer to Layer Management. If an invocation is unsuccessful, an Error indication primitive is provided in the primitive. These requests result in outgoing ASP Up, ASP Down, ASP Active and ASP Inactive messages to the remote TUA peer at an SGP or IPSP.

4.2. Procedures to Support the Management of SCTP Associations

4.2.1. Receipt of TUA Peer Management Messages

Upon successful state changes resulting from reception of ASP Up, ASP Down, ASP Active and ASP Inactive messages from a peer TUA, the TUA layer **MAY** invoke corresponding M-ASP_UP, M-ASP_DOWN, M-ASP_ACTIVE and M-ASP_INACTIVE, M-AS_ACTIVE, M-AS_INACTIVE, and M-AS_DOWN indication primitives to the local Layer Management.

M-NOTIFY indication and M-ERROR indication primitives indicate to Layer Management the notification or error information contained in a received TUA Notify or Error message. These indications can also be generated based on local TUA events.

All MGMT, ASPSM, ASPTM and RKM messages, except BEAT and BEAT Ack, **SHOULD** be sent with sequenced delivery to ensure ordering. All MGMT, ASPSM and RKM messages, with the exception of BEAT and BEAT Ack messages **MUST** be sent on SCTP stream '0'. ASPTM messages **MAY** be sent on one of the streams used to carry data traffic related to the Routing Context(s), to minimize possible message loss. BEAT and BEAT Ack messages **MAY** be sent using out-of-order delivery, and **MAY** be sent on any stream.

4.3. AS and ASP State Maintenance

The TUA layer on the SGP maintains the state of each remote ASP, in each Application Server that the ASP is configured to receive traffic, as input to the TUA message distribution function. Similarly, where IPSPs use TUA in a point-to-point fashion, the TUA layer in an IPSP maintains the state of remote IPSPs. For the purposes of the following procedures, only the SGP and ASP case is described but the SGP side of the procedures also apply to an IPSP sending traffic to an AS consisting of a set of remote IPSPs.

4.3.1. ASP States

The state of each remote ASP, in each AS that it is configured to operate, is maintained in the TUA layer in the SGP. The state of a particular ASP in a particular AS changes due to events. The events include:

- reception of messages from the peer TUA layer at the ASP;
- reception of some messages from the peer TUA layer at other ASPs in the AS (e.g, ASP Active message indicating "Override");
- reception of indications from the SCTP layer; or,
- Local Management intervention.

The ASP state transition diagram is shown in *Figure 5*. The possible states of an ASP are:

ASP-DOWN: The remote TUA peer at the ASP is unavailable or the related SCTP association is down. Initially all ASPs will be in this state. An ASP in this state **SHOULD NOT** be sent any TUA messages, with the exception of Heartbeat, ASP Down Ack and Error messages.

ASP-INACTIVE:
The remote TUA peer at the ASP is available (and the related SCTP association is up) but application traffic is stopped. In this state, the ASP **SHOULD NOT** be sent any DH, CH or SSNM messages for the AS for which the ASP is inactive.

ASP-ACTIVE: The remote TUA peer at the ASP is available and application traffic is active (for a particular Routing Context or set of Routing Contexts).

SCTP CDI: The SCTP CDI denotes the local SCTP layer's Communication Down Indication to the Upper Layer Protocol (TUA) on an SGP. The local SCTP layer will send this indication when it detects the loss of

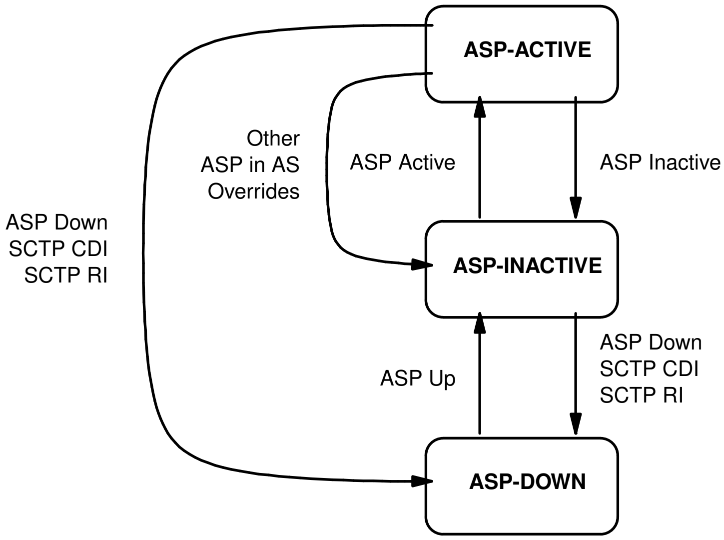


Figure 5. ASP State Transition Diagram (Per AS)

connectivity to the ASPs peer SCTP layer. SCTP CDI is understood as either a SHUTDOWN_COMPLETE notification or COMMUNICATION_LOST notification from the SCTP layer.

SCTP RI: The local SCTP layer's Restart indication to the upper layer protocol (TUA) on an SG. The local SCTP will send this indication when it detects a restart from the ASPs peer SCTP layer.

4.3.2. AS States

The state of the AS is maintained in the TUA layer on the SGP. The state of an AS changes due to events. These events include:

- ASP state transitions
- Recovery timer triggers

The possible states of an AS are:

AS-DOWN: The Application Server is unavailable. This state implies that all related ASPs are in the ASP-DOWN state for this AS. Initially the AS will be in this state. An Application Server is in the AS-DOWN state when it is removed from a configuration.

AS-INACTIVE: The Application Server is available but no application traffic is active (i.e., one or more related ASPs are in the ASP-INACTIVE state, but none in the ASP-ACTIVE state). The recovery timer T(r) is not running or has expired.

AS-ACTIVE: The Application Server is available and application traffic is active. This state implies that at least one ASP is in the ASP-ACTIVE state.

AS-PENDING: An active ASP has transitioned to ASP-INACTIVE or ASP-DOWN and it was the last remaining active ASP in the AS. A recovery timer T(r) **SHOULD** be started and all incoming signalling messages **SHOULD** be queued by the SGP. If an ASP becomes ASP-ACTIVE before T(r) expires, the AS is moved to the AS-ACTIVE state and all the queued messages will be sent to the ASP.

If T(r) expires before an ASP becomes ASP-ACTIVE, and the SGP has no other alternative, the SGP may stop queuing messages and discard all previously queued messages. The AS will move to the AS-INACTIVE state if at least one ASP is in ASP-INACTIVE state, otherwise it will move to AS-DOWN state.

Figure 6 shows an example AS state machine for the case where the AS data is pre-configured. For other cases where the ASP configuration data is created dynamically, there would be differences in the state machine, especially at creation of the AS.

For example, where the AS configuration data is not created until Registration of the first ASP, the AS-INACTIVE state is entered directly upon the first successful REG REQ from an ASP. Another example is where the AS configuration data is not created until the first ASP successfully enters the ASP-ACTIVE state. In this case the AS-ACTIVE state is entered directly.

4.3.2.1. IPSP Considerations

The AS state diagram for the AS-SG case is applicable for IPSP communication.

4.3.3. TUA Management Procedures for Primitives

Before the establishment of an SCTP association the ASP state at both the SGP and ASP is assumed to be in the state ASP-DOWN.

Once the SCTP association is established (see Section 4.2.1) and assuming that the local TC-User is ready, the local TUA ASP Maintenance (ASPM) function will initiate the relevant procedures, using the ASP Up, ASP Down, ASP Active and ASP Inactive messages to convey the ASP state to the SGP (see Section 4.3.4).

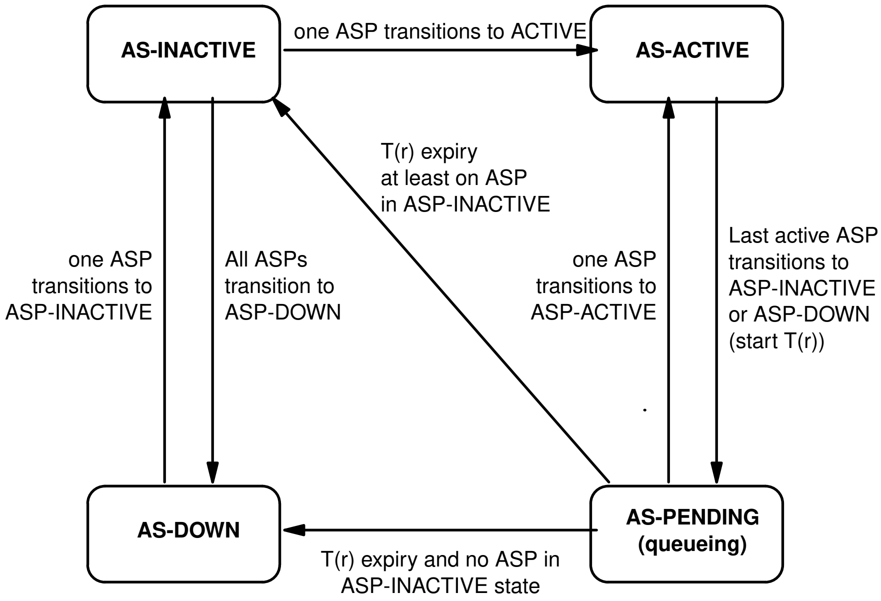


Figure 6. AS State Transition Diagram

If the TUA layer subsequently receives an SCTP-COMMUNICATION_DOWN or SCTP-RESTART indication primitive from the underlying SCTP layer, it will inform the Layer Management by invoking the M-SCTP_STATUS indication primitive. The state of the ASP will be moved to ASP-DOWN.

At an ASP, the TC-User will be informed of the unavailability of any affected SS7 destination through the use of N-PC-STATE indication primitives.

In the case of SCTP-COMMUNICATION_DOWN, the SCTP client **MAY** try to re-establish the SCTP association. This **MAY** be done by the TUA layer automatically, or Layer Management **MAY** re-establish using the M- SCTP_ESTABLISH request primitive.

In the case of an SCTP-RESTART indication at an ASP, the ASP is now considered by its TUA peer to be in the ASP-DOWN state. The ASP, if it is to recover, must begin any recovery with the ASP-Up procedure.

4.3.4. ASPM Procedures for Peer-to-Peer Messages

4.3.4.1. ASP Up Procedures

After an ASP has successfully established an SCTP association to an SGP, the SGP waits for the ASP to send an ASP Up message, indicating that the ASP TUA peer is available. The ASP is always the initiator of the ASP Up message. This action **MAY** be initiated at the ASP by an M-ASP_UP request primitive from Layer Management or **MAY** be initiated automatically by an TUA management function.

When an ASP Up message is received at an SGP and internally the remote ASP is in the ASP-DOWN state and not considered locked-out for local management reasons, the SGP marks the remote ASP in the state ASP-INACTIVE and informs Layer Management with an M-ASP_Up indication primitive. If the SGP is aware, via current configuration data, which Application Servers the ASP is configured to operate in, the SGP updates the ASP state to ASP-INACTIVE in each AS that it is a member.

Alternatively, the SGP may move the ASP into a pool of Inactive ASPs available for future configuration within Application Server(s), determined in a subsequent Registration Request or ASP Active procedure. If the ASP Up message contains an ASP Identifier, the SGP should save the ASP Identifier for that ASP. The SGP **MUST** send an ASP Up Ack message in response to a received ASP Up message even if the ASP is already marked as ASP-INACTIVE at the SGP.

If for any local reason (e.g, management lock-out) the SGP cannot respond with an ASP Up Ack message, the SGP responds to an ASP Up message with an Error message with Reason "Refused - Management Blocking".

At the ASP, the ASP Up Ack message received is not acknowledged. Layer Management is informed with an M-ASP_UP confirm primitive.

When the ASP sends an ASP Up message it starts timer T(ack). If the ASP does not receive a response to an ASP Up message within T(ack), the ASP **MAY** restart T(ack) and resend ASP Up messages until it receives an ASP Up Ack message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of ASP Up messages **MAY** be put under control of Layer Management. In this method, expiry of T(ack) results in an M-ASP_UP confirm primitive carrying a negative indication.

The ASP must wait for the ASP Up Ack message before sending any other TUA messages (e.g, ASP Active or REG REQ). If the SGP receives any other TUA messages before ASPUP message is received (other than ASPDN - see Section 4.3.4.2), the SGP **SHOULD** discard them.

If an ASP Up message is received and internally the remote ASP is in the ASP-ACTIVE state, an ASP Up Ack message is returned, as well as an Error message ("Unexpected Message), and the remote ASP state is changed to ASP-INACTIVE in all relevant Application Servers.

If an ASP Up message is received and internally the remote ASP is already in the ASP-INACTIVE state, an ASP Up Ack message is returned and no further action is taken.

4.3.4.1.1. TUA Version Control

If an ASP Up message with an unsupported version is received, the receiving end responds with an Error message, indicating the version the receiving node supports and notifies Layer Management.

This is useful when protocol version upgrades are being performed in a network. A node upgraded to a newer version should support the older versions used on other nodes it is communicating with. Because ASPs initiate the ASP Up procedure it is assumed that the Error message would normally come from the SGP.

4.3.4.1.2. IPSP Considerations

An IPSP may be considered in the ASP-INACTIVE state after and ASPUP or ASPUP Ack has been received from it. An IPSP can be considered in the ASP-DOWN state after an ASPDN or ASPDN Ack has been received from it. The IPSP may inform Layer Management of the change in state of the remote IPSP using M-ASP_UP or M-ASP_DN indication or confirmation primitives.

Alternatively, an interchange of ASPUP messages from each end can be performed. This option follows the ASP state transition diagram. It would need four messages for completion.

If for any local reason (e.g, management lock-out) and IPSP cannot respond to an ASP Up message with an ASP Up Ack message, it responds to an ASP Up message with an Error message with Reason "Refused - Management Blocking" and leaves the remote IPSP in the ASP-DOWN state.

4.3.4.2. ASP Down Procedures

The ASP will send an ASP Down message to an SGP when the ASP wishes to be removed from service in all Application Servers that it is a member and no longer receive any DATA, SSNM or ASPTM messages. This action **MAY** be initiated at the ASP by an M-ASP_DOWN request primitive from Layer Management or **MAY** be initiated automatically by an TUA management function.

Whether the ASP is permanently removed from any AS is a function of configuration management. Whenever the ASP previously used the Registration procedures (see Section 4.4.1) to register within Application Servers but has not deregistered from all of them prior to sending the ASP Down message, the SGP **MUST** consider the ASP as Deregistered in all Application Servers that it is still a member.

The SGP marks the ASP as ASP-DOWN, informs Layer Management with an M-ASP_Down indication primitive, and returns an ASP Down Ack message to the ASP.

The SGP **MUST** send an ASP Down Ack message in response to a received ASP Down message from the ASP even if the ASP is already marked as ASP-DOWN at the SGP.

At the ASP, the ASP Down Ack message received is not acknowledged. Layer Management is informed with an M-ASP_DOWN confirm primitive. If the ASP receives an ASP Down Ack without having sent an ASP Down message, the ASP should now consider itself as in the ASP-DOWN state. If the ASP was previously in the ASP-ACTIVE or ASP_INACTIVE state, the ASP should then initiate procedures to return itself to its previous state.

When the ASP sends an ASP Down message it starts timer T(ack). If the ASP does not receive a response to an ASP Down message within T(ack), the ASP **MAY** restart T(ack) and resend ASP Down messages until it receives an ASP Down Ack message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of ASP Down messages **MAY** be put under control of Layer Management. In this method, expiry of T(ack) results in an M-ASP_DOWN confirm primitive carrying a negative indication.

4.3.4.3. ASP Active Procedures

Anytime after the ASP has received an ASP Up Ack message from the SGP or IPSP, the ASP **MAY** send an ASP Active message to the SGP indicating that the ASP is ready to start processing traffic. This action **MAY** be initiated at the ASP by an M-ASP_ACTIVE request primitive from Layer Management or **MAY** be initiated automatically by an TUA management function. Whenever an ASP wishes to process the traffic for more than one Application Server across a common SCTP association, the ASP Active message(s) **SHOULD** contain a list of one or more Routing Contexts to indicate for which Application Servers the ASP Active message applies. It is not necessary for the ASP to include all Routing Contexts of interest in a single ASP Active message, thus requesting to become active in all Routing Contexts at the same time. Multiple ASP Active messages **MAY** be used to activate within the Application Servers independently, or in sets. Whenever an ASP Active message does not contain a Routing Context parameter, the receiver must know, via configuration data, which Application Server(s) the ASP is a member.

For the Application Servers that the ASP can successfully activate, the SGP or IPSP responds with one or more ASP Active Ack messages, including the associated Routing Context(s) and reflecting any Traffic Mode Type values present in the related ASP Active message. The Routing Context parameter **MUST** be included in the ASP Active Ack message(s) if the received ASP Active message contained any Routing Contexts. Depending on any Traffic Mode Type request in the ASP Active message or local configuration data if there is no request, the SGP moves the ASP to the correct ASP traffic state within the associated Application Server(s). Layer Management is informed with an M-ASP_Active indication. If the SGP or IPSP receives any Data messages before an ASP Active message is received, the SGP or IPSP **MAY** discard them. By sending an ASP Active Ack message, the SGP or IPSP is now ready to receive and send traffic for the related Routing Context(s). The ASP **SHOULD NOT** send Data messages for the related Routing Context(s) before receiving an ASP Active Ack message, or it will risk message loss.

Multiple ASP Active Ack messages **MAY** be used in response to an ASP Active message containing multiple Routing Contexts, allowing the SGP or IPSP to independently acknowledge the ASP Active message for different (sets of) Routing Contexts. The SGP or IPSP **MUST** send an Error message ("Invalid Routing Context") for each Routing Context value that cannot be successfully activated.

Whenever an "out-of-the-blue" ASP Active message is received (i.e., the ASP has not registered with the SG or the SG has no static configuration data for the ASP), the message **MAY** be silently discarded.

The SGP **MUST** send an ASP Active Ack message in response to a received ASP Active message from the ASP, if the ASP is already marked in the ASP-ACTIVE state at the SGP.

At the ASP, the ASP Active Ack message received is not acknowledged. Layer Management is informed with an M-ASP_ACTIVE confirm primitive. It is possible for the ASP to receive Data message(s) before the ASP Active Ack message as the ASP Active Ack and Data messages from an SG or IPSP may be sent on different SCTP streams. Message loss is possible, as the ASP does not consider itself in the ASP-ACTIVE state until reception of the ASP Active Ack message.

When the ASP sends an ASP Active message it starts timer T(ack). If the ASP does not receive a response to an ASP Active message within T(ack), the ASP **MAY** restart T(ack) and resend ASP Active messages until it receives an ASP Active Ack message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of ASP Active messages **MAY** be put under control of Layer Management. In this method, expiry of T(ack) results in an M-ASP_ACTIVE confirm primitive carrying a negative indication.

There are three modes of Application Server traffic handling in the SGP TUA layer: Override, Load-share and Broadcast. When included, the Traffic Mode Type parameter in the ASP Active message indicates the traffic-handling mode to be used in a particular Application Server. If the SGP determines that the mode indicated in an ASP Active message is unsupported or incompatible with the mode currently configured for the AS, the SGP responds with an Error message ("Unsupported/Invalid Traffic Handling Mode"). If the traffic- handling mode of the Application Server is not already known via configuration data, then the traffic-handling mode indicated in the first ASP Active message causing the transition of the Application Server state to AS-ACTIVE **MAY** be used to set the mode.

In the case of an Override mode AS, reception of an ASP Active message at an SGP causes the (re)direction of all traffic for the AS to the ASP that sent the ASP Active message. Any previously active ASP in the AS is now considered to be in state ASP-INACTIVE and **SHOULD** no longer receive traffic from the SGP within the AS. The SGP or IPSP then **MUST** send a Notify message ("Alternate ASP Active") to the previously active ASP in the AS, and **SHOULD** stop traffic to or from that ASP. The ASP receiving this Notify **MUST** consider itself now in the ASP-INACTIVE state, if it is not already aware of this via inter- ASP communication with the Overriding ASP.

In the case of a Load-share mode AS, reception of an ASP Active message at an SGP or IPSP causes the direction of traffic to the ASP sending the ASP Active message, in addition to all the other ASPs that are currently active in the AS. The algorithm at the SGP for load-sharing traffic within an AS to all the active ASPs is implementation dependent. The algorithm could, for example, be round robin or based on information in the Data message (e.g, the SLS or SSN).

An SGP or IPSP, upon reception of an ASP Active message for the first ASP in a Load-share AS, **MAY** choose not to direct traffic to a newly active ASP until it determines that there are sufficient resources to handle the expected load (e.g, until there are "n" ASPs in state ASP-ACTIVE in the AS).

All ASPs within a load-sharing mode AS must be able to process any Data message received for the AS, to accommodate any potential fail-over or re-balancing of the offered load.

In the case of a Broadcast mode AS, reception of an ASP Active message at an SGP or IPSP causes the direction of traffic to the ASP sending the ASP Active message, in addition to all the other ASPs that are currently active in the AS. The algorithm at the SGP for broadcasting traffic within an AS to all the active ASPs is a simple broadcast algorithm, where every message is sent to each of the active ASPs. An SGP or IPSP, upon reception of an ASP Active message for the first ASP in a Broadcast AS, **MAY** choose not to direct traffic to a newly active ASP until it determines that there are sufficient resources to handle the expected load (e.g, until there are "n" ASPs in state ASP-ACTIVE in the AS).

Whenever an ASP in a Broadcast mode AS becomes ASP-ACTIVE, the SGP **MUST** tag the first DATA message broadcast in each SCTP stream with a unique Correlation Id parameter. The purpose of this Correlation Id is to permit the newly active ASP to synchronize it's processing of traffic in each ordered stream with the other ASPs in the broadcast group.

4.3.4.3.1. IPSP Considerations

Either of the IPSPs can initiate communication. When an IPSP receives an ASP Active, it should mark the peer as ASP-ACTIVE and return an ASP Active Ack message. An ASP receiving an ASP Active Ack message may mark the peer as ASP-Active, if it is not already in the ASP- ACTIVE state.

Alternatively, an interchange of ASPAC messages from each end can be performed. This option follows the ASP state transition diagram and gives the additional advantage of selecting a particular AS to be activated from each end. It is especially useful when an IPSP is serving more than one AS. It would need four messages for completion.

4.3.4.4. ASP Inactive Procedures

When an ASP wishes to withdraw from receiving traffic within an AS, the ASP sends an ASP Inactive message to the SGP or IPSP. This action **MAY** be initiated at the ASP by an M-ASP_INACTIVE request primitive from Layer

Management or **MAY** be initiated automatically by an TUA management function. Whenever an ASP is processing the traffic for more than one Application Server across a common SCTP association, the ASP Inactive message contains one or more Routing Contexts to indicate for which Application Servers the ASP Inactive message applies. Whenever an ASP Inactive message does not contain a Routing Context parameter, the receiver must know, via configuration data, which Application Servers the ASP is a member and move the ASP to the ASP-INACTIVE state in each all Application Servers. In the case of an Override mode AS, where another ASP has already taken over the traffic within the AS with an ASP Active ("Override") message, the ASP that sends the ASP Inactive message is already considered by the SGP to be in state ASP-INACTIVE. An ASP Inactive Ack message is sent to the ASP, after ensuring that all traffic is stopped to the ASP.

In the case of a Load-share mode AS, the SGP moves the ASP to the ASP-INACTIVE state and the AS traffic is re-allocated across the remaining ASPs in the state ASP-ACTIVE, as per the load-sharing algorithm currently used within the AS. A Notify message ("Insufficient ASP resources active in AS") **MAY** be sent to all inactive ASPs, if required. An ASP Inactive Ack message is sent to the ASP after all traffic is halted and Layer Management is informed with an M-ASP_INACTIVE indication primitive.

In the case of a Broadcast mode AS, the SGP moves the ASP to the ASP- INACTIVE state and the AS traffic is broadcast only to the remaining ASPs in the state ASP-ACTIVE. A Notify message ("Insufficient ASP resources active in AS") **MAY** be sent to all inactive ASPs, if required. An ASP Inactive Ack message is sent to the ASP after all traffic is halted and Layer Management is informed with an M-ASP_INACTIVE indication primitive.

Multiple ASP Inactive Ack messages **MAY** be used in response to an ASP Inactive message containing multiple Routing Contexts, allowing the SGP or IPSP to independently acknowledge for different (sets of) Routing Contexts. The SGP or IPSP sends an Error message ("Invalid Routing Context") message for each invalid or unconfigured Routing Context value in a received ASP Inactive message.

The SGP **MUST** send an ASP Inactive Ack message in response to a received ASP Inactive message from the ASP and the ASP is already marked as ASP-INACTIVE at the SGP.

At the ASP, the ASP Inactive Ack message received is not acknowledged. Layer Management is informed with an M-ASP_INACTIVE confirm primitive. If the ASP receives an ASP Inactive Ack without having sent an ASP Inactive message, the ASP should now consider itself as in the ASP-INACTIVE state. If the ASP was previously in the ASP-ACTIVE state, the ASP should then initiate procedures to return itself to its previous state. When the ASP sends an ASP Inactive message it starts timer T(ack). If the ASP does not receive a response to an ASP Inactive message within T(ack), the ASP **MAY** restart T(ack) and resend ASP Inactive messages until it receives an ASP Inactive Ack message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of ASP Inactive messages **MAY** be put under control of Layer Management. In this method, expiry of T(ack) results in a M-ASP_Inactive confirm primitive carrying a negative indication.

If no other ASPs in the Application Server are in the state ASP- ACTIVE, the SGP **MUST** send a Notify message ("AS-Pending") to all of the ASPs in the AS which are in the state ASP-INACTIVE. The SGP **SHOULD** start buffering the incoming messages for T(r) seconds, after which messages **MAY** be discarded. T(r) is configurable by the network operator. If the SGP receives an ASP Active message from an ASP in the AS before expiry of T(r), the buffered traffic is directed to that ASP and the timer is canceled. If T(r) expires, the AS is moved to the AS-INACTIVE state.

4.3.4.4.1. IPSP Considerations

An IPSP may be considered in the ASP-INACTIVE state by a remote IPSP after an ASP Inactive or ASP Inactive Ack message has been received from it.

Alternatively, an interchange of ASPIA messages from each end can be performed. This option follows the ASP state transition diagram and gives the additional advantage of selecting a particular AS to be deactivated from each end. It is especially useful when an IPSP is serving more than one AS. It would need four messages for completion.

4.3.4.5. Notify Procedures

A Notify message reflecting a change in the AS state **MUST** be sent to all ASPs in the AS, except those in the ASP-DOWN state, with appropriate Status Information and any ASP Identifier of the failed ASP. At the ASP, Layer Management is informed with an M- NOTIFY indication primitive. The Notify message must be sent whether the AS state change was a result of an ASP failure or reception of an ASP State management (ASPSM) or ASP Traffic Management (ASPTM) message. In the second case, the Notify message **MUST** be sent after any ASP State or Traffic Management related acknowledgments messages (e.g, ASP Up Ack, ASP Down Ack, ASP Active Ack, or ASP Inactive Ack).

Whenever a Notify ("AS-PENDING") message is sent by an SGP that now has no ASPs active to service the traffic, or where a Notify ("Insufficient ASP resources active in AS") message **MUST** be sent in the Load-share or Broadcast mode, the Notify message does not explicitly compel the ASP(s) receiving the message to become active. The ASPs remain in control of what (and when) traffic action is taken.

Whenever a Notify message does not contain a Routing Context parameter, the receiver must know, via configuration data, of which Application Servers the ASP is a member and take the appropriate action in each AS.

4.3.4.5.1. IPSP Considerations (NTFY)

Notify works in the same manner as in the SG-AS case. One of the IPSPs can send this message to any remote IPSP that is not in the ASP-DOWN state.

4.3.4.6. Heartbeat Procedures

The optional Heartbeat procedures **MAY** be used when operating over transport layers that do not have their own heartbeat mechanism for detecting loss of the transport association (i.e., other than SCTP).

Either TUA peer may optionally send Heartbeat messages periodically, subject to a provisionable timer T(beat). Upon receiving a Heartbeat message, the TUA peer **MUST** respond with a Heartbeat Ack message.

If no Heartbeat Ack message (or any other TUA message) is received from the TUA peer within 2*T(beat), the remote TUA peer is considered unavailable. Transmission of Heartbeat messages is stopped and the signalling process **SHOULD** attempt to re-establish communication if it is configured as the client for the disconnected TUA peer.

The Heartbeat message may optionally contain an opaque Heartbeat Data parameter that **MUST** be echoed back unchanged in the related Heartbeat Ack message. The sender, upon examining the contents of the returned Heartbeat Ack message, **MAY** choose to consider the remote TUA peer as unavailable. The contents and format of the Heartbeat Data parameter is implementation-dependent and only of local interest to the original sender. The contents may be used, for example, to support a Heartbeat sequence algorithm (to detect missing Heartbeats), or a time-stamp mechanism (to evaluate delays).

Note: Heartbeat related events are not shown in Figure 4 "ASP state transition diagram".

4.4. Routing Key Management Procedures

4.4.1. Registration

An ASP **MAY** dynamically register with an SGP as an ASP within an Application Server using the REG REQ message. A Routing Key parameter in the REG REQ message specifies the parameters associated with the Routing Key.

The SGP examines the contents of the received Routing Key parameter and compares it with the currently provisioned Routing Keys. If the received Routing Key matches an existing SGP Routing Key entry, and the ASP is not currently included in the list of ASPs for the related Application Server, the SGP **MAY** authorize the ASP to be added to the AS. Or, if the Routing Key does not currently exist and the received Routing Key data is valid and unique, an SGP supporting dynamic configuration **MAY** authorize the creation of a new Routing Key and related Application Server and add the ASP to the new AS. In either case, the SGP returns a Registration Response message to the ASP, containing the same Local-RK-Identifier as provided in the initial request, and a Registration Result "Successfully Registered". A unique Routing Context value assigned to the SGP Routing Key is included. The method of Routing Context value assignment at the SGP is implementation dependent but must be guaranteed to be unique for each Application Server or Routing Key supported by the SGP. If the SGP determines that the received Routing Key data is invalid, or contains invalid parameter values, the SGP returns a Registration Response message to the ASP, containing a Registration Result "Error - Invalid Routing Key", "Error - Invalid DPC", "Error - Invalid Network Appearance" as appropriate.

If the SGP does not support the registration procedure, the SGP returns an Error message to the ASP, with an error code of "Unsupported Message Type".

If the SGP determines that a unique Routing Key cannot be created, the SGP returns a Registration Response message to the ASP, with a Registration Status of "Error - "Cannot Support Unique Routing." An incoming signalling message received at an SGP should not match against more than one Routing Key.

If the SGP does not authorize the registration request, the SGP returns a REG RSP message to the ASP containing the Registration Result "Error - Permission Denied".

If an SGP determines that a received Routing Key does not currently exist and the SGP does not support dynamic configuration, the SGP returns a Registration Response message to the ASP, containing a Registration Result "Error - Routing Key not Currently Provisioned".

If an SGP determines that a received Routing Key does not currently exist and the SGP supports dynamic configuration but does not have the capacity to add new Routing Key and Application Server entries, the SGP returns a Registration Response message to the ASP, containing a Registration Result "Error - Insufficient Resources".

If an SGP determines that one or more of the Routing Key parameters are not supported for the purpose of creating new Routing Key entries, the SGP returns a Registration Response message to the ASP, containing a Registration Result "Error - Unsupported RK parameter field". This result **MAY** be used if, for example, the SGP does not support RK Address parameter.

A Registration Response "Error - Unsupported Traffic Handling Mode" is returned if the Routing Key in the REG REQ contains a Traffic Handling Mode that is inconsistent with the presently configured mode for the matching Application Server.

An ASP **MAY** register multiple Routing Keys at once by including a number of Routing Key parameters in a single REG REQ message. The SGP **MAY** respond to each registration request in a single REG RSP message, indicating the success or failure result for each Routing Key in a separate Registration Result parameter. Alternatively the SGP **MAY** respond with multiple REG RSP messages, each with one or more Registration Result parameters. The ASP uses the Local-RK-Identifier parameter to correlate the requests with the responses.

An ASP **MAY** modify an existing Routing Key by including a Routing Context parameter in the REG REQ. If the SGP determines that the Routing Context applies to an existing Routing Key, the SG **MAY** adjust the existing Routing Key to match the new information provided in the Routing Key parameter. A Registration Response "Routing Context Registration Refused" is returned if the SGP does not accept the modification of the Routing Key.

Upon successful registration of an ASP in an AS, the SGP can now send related SS7 Signalling Network Management messaging, if this did not previously start upon the ASP transition to state ASP-INACTIVE

4.4.2. Deregistration

An ASP **MAY** dynamically deregister with an SGP as an ASP within an Application Server using the DEREG REQ message. A Routing Context parameter in the DEREG REQ message specifies which Routing Keys to deregister. An ASP **SHOULD** move to the ASP-INACTIVE state for an Application Server before attempting to deregister the Routing Key (i.e., deregister after receiving an ASP Inactive Ack). Also, an ASP **SHOULD** deregister from all Application Servers that it is a member before attempting to move to the ASP-Down state.

The SGP examines the contents of the received Routing Context parameter and validates that the ASP is currently registered in the Application Server(s) related to the included Routing Context(s). If validated, the ASP is deregistered as an ASP in the related Application Server.

The deregistration procedure does not necessarily imply the deletion of Routing Key and Application Server configuration data at the SGP. Other ASPs may continue to be associated with the Application Server, in which case the Routing Key data **MUST NOT** be deleted. If a Deregistration results in no more ASPs in an Application Server, an SGP **MAY** delete the Routing Key data.

The SGP acknowledges the deregistration request by returning a DEREG RSP message to the requesting ASP. The result of the deregistration is found in the Deregistration Result parameter, indicating success or failure with cause.

An ASP **MAY** deregister multiple Routing Contexts at once by including a number of Routing Contexts in a single DEREG REQ message. The SGP **MAY** respond to each deregistration request in a single DEREG RSP message, indicating the success or failure result for each Routing Context in a separate Deregistration Result parameter.

4.4.3. IPSP Considerations (REG/DEREG)

The Registration and Deregistration procedures work in the IPSP cases in the same way as in AS-SG cases. An IPSP may register an RK in the remote IPSP. An IPSP is responsible for deregistering the RKs that it has registered.

4.5. Procedures to Support Point Code and Subsystem State

4.5.1. At an SGP

On receiving an N-STATE, N-PCSTATE, N-COORD indication primitive from the nodal inter-working function at an SGP, the SGP TUA layer will send a corresponding SS7 Signalling Network Management (SSNM) DUNA, DAVA, DUPU, DRST or SCON message (see Section 3) to the TUA peers at concerned ASPs. The TUA layer must fill in various fields of the SSNM messages consistently with the information received in the primitives.

SSNM messages **SHOULD NOT** be sent on stream "0" and **MAY** use ordered delivery.

4.5.2. At an ASP

4.5.2.1. Single SG Configurations

At an ASP, upon receiving an SS7 Signalling Network Management (SSNM) message from the remote TUA Peer, the TUA layer invokes the appropriate primitive indications to the resident TC-Users. Local management is informed.

Whenever a local event has caused the unavailability or congestion status of SS7 destinations, user parts or subsystems, the TUA layer at the ASP **SHOULD** pass up appropriate indications in the primitives to the TUA User, as though equivalent SSNM messages were received. For example, the loss of an SCTP association to an SGP may cause the unavailability of a set of SS7 destinations, user parts or subsystems. N-PCSTATE indication primitives to the TUA User are appropriate.

4.5.2.2. Multiple SG Configurations

At an ASP, upon receiving a SS7 Signalling Network Management (SSNM) message from the remote TUA Peer, the TUA layer updates the status of the affected route(s) via the originating SG and determines, whether or not the overall availability or congestion status of the effected destination(s) or subsystem(s) has changed. If so, the TUA layer invokes the appropriate primitive indications to the resident TC-Users [10]. Local management is informed.

4.5.3. ASP Auditing

An ASP may optionally initiate an audit procedure to inquire of an SG the availability and, if the national congestion method with multiple congestion levels and message priorities is used, congestion status of an SS7 destination or set of destinations. In addition, the ASP may inquire of an SG the availability and congestion status of a subsystem. A Destination Audit (DAUD) message is sent from the ASP to the SGP requesting the current availability and congestion status of one or more SS7 destinations or subsystems.

The DAUD message **MAY** be sent with unordered delivery. The ASP **MAY** send the DAUD in the following cases:

- Periodic: A Timer originally set upon reception of a DUNA, SCON or DRST message has expired without a subsequent DAVA, DUNA, SCON or DRST message updating the availability and congestion status of the affected destinations or subsystems. The Timer is reset upon issuing a DAUD. In this case the DAUD is sent to the SGP that originally sent the SSNM message [11].

- Isolation: The ASP is newly ASP-ACTIVE or has been isolated from an SG for an extended period. The ASP **MAY** request the availability and congestion status of one or more SS7 destinations or subsystems to which it expects to communicate.

The SGP **MUST** either respond to a DAUD messages with SSNM messages indicating the availability and congestion status of the destination or subsystem, or **MUST** respond with an ERR ("Destination Status Unknown") or ERR ("Subsystem Status Unknown") message for each destination or subsystem requested in the DAUD message.

The status of each SS7 destination or subsystem requested is indicated in a DUNA message (if unavailable), a DAVA message (if available), or a DRST (if restricted and the SGP supports this feature). If the SS7 destination or subsystem is available and congested, the SGP responds with an SCON message in addition to the DAVA message. If the SS7 destination is restricted and congested, the SGP responds with an SCON message in addition to the DRST. If the SGP cannot return information on the availability or congestion status of the SS7 destination or subsystem, the SGP responds with an ERR ("Destination Status Unknown") or ERR ("Subsystem Status Unknown") with a list of all the destinations and subsystems for which the SGP cannot provide information.

In some cases, the SGP **MAY** chose not to respond to a DAUD message or a component of a DAUD message on the basis of policy [12].

Any DUNA or DAVA message in response to a DAUD message **MAY** contain a list of Affected Point Codes.

4.5.4. TCAP – TUA Interworking at the SG

On the SG, the TCAP routing or interworking function determines that the message must be sent to an AS via the TUA stack, based on information in the incoming message. The TUA outgoing mapping function identifies the appropriate Application Server (AS) and selects an active ASP from the list of ASPs servicing this AS. The appropriate ASP can be determined based on the routing information in the incoming message, local load sharing information, etc. The appropriate TUA message is then constructed and sent to the appropriate endpoint, via the correct SCTP association and stream.

4.5.4.1. Primitives received from the local TC-User

These support the TUA transport of TC-User boundary primitives. The same services as supported by TCAP are to be provided by TUA. The TC-users at the SG should be able to use the same primitive interface to TCAP/TUA without any changes. The TCAP-TUA interworking function takes care of selecting the appropriate stack.

The TUA needs to setup and maintain the appropriate SCTP association to the selected endpoint. TUA also manages the usage of SCTP streams. The address information passed by the TUA-user at an ASP must contain: .np a valid SS7 address to reach a destination in the SS7 network via the appropriate SCTP association to a SG .np a valid IP address or host name to reach another ASP in the IP network via the appropriate SCTP association.

4.5.4.2. Segmenting and Reassembly of Components

When it is expected that TCAP signalling messages will not fit into the maximum PDU size of the underlying transport (e.g, SCCP, MTP), then segmentation and reassembly **SHOULD** be performed by the TC-User. In the event that the SG receives a TQRY, TCNV and TRSP message with included or associated components that exceed the maximum PDU size of the underling transport, the SGP will respond with a TNOT message with "Segmentation Not Supported" or "Segmentation Failed" or "Destination cannot perform reassembly" indicated in the *Report Cause* within the TNOT message considering local SG SCCP procedures [13].

5. Examples of TUA Procedures

5.1. Establishment of Association and Traffic between SGPs and ASPs

5.1.1.1. Single ASP in an Application Server ("1+0" sparing)

This scenario shows the example TUA message flows for the establishment of traffic between an SG and an ASP, where only one ASP is configured within an AS (no backup). It is assumed that the SCTP association is already set-up.

```
SG                              ASP
 |                               |
 |<-------------ASP Up-----------|
 |-----------ASP-Up Ack--------->|
 |                               |
 |<------- ASP Active------------|
 |-----ASP Active Ack----------->|
 |                               |
```

5.1.1.2. Two ASPs in Application Server ("1+1" sparing)

This scenario shows the example TUA message flows for the establishment of traffic between an SG and two ASPs in the same Application Server, where ASP1 is configured to be "active" and ASP2 a "standby" in the event of communication failure or the withdrawal from service of ASP1. ASP2 may act as a hot, warm, or cold standby depending on the extent to which ASP1 and ASP2 share call or transaction state or can communicate call state under failure or withdrawal events. The example message flow is the same whether the ASP-Active messages are Override or Load-share mode although typically this example would use an Override mode.

```
SG                        ASP1                      ASP2
 |                         |                          |
 |<--------ASP Up----------|                          |
 |-------ASP-Up Ack------->|                          |
 |                         |                          |
 |<----------------------------ASP Up-----------------|
 |----------------------------ASP-Up Ack------------->|
 |                         |                          |
 |                         |                          |
 |<-------ASP Active-------|                          |
 |------ASP-Active Ack---->|                          |
 |                         |                          |
```

5.1.1.3. Two ASPs in an Application Server ("1+1" sparing, load-sharing case)

This scenario shows the example TUA message flows for the establishment of traffic between an SG and two ASPs in the same Application Server, where the two ASPs are brought to "active" and load-share the traffic load. In this case, one ASP is sufficient to handle the total traffic load.

```
SG                        ASP1                      ASP2
 |                         |                          |
 |<---------ASP Up---------|                          |
 |--------ASP-Up Ack------>|                          |
 |                         |                          |
 |<-----------------------------ASP Up----------------|
 |----------------------------ASP Up Ack------------->|
 |                         |                          |
 |                         |                          |
 |<--ASP Active -----------|                          |
 |-----ASP-Active Ack----->|                          |
 |                         |                          |
 |<----------------------------ASP Active ------------|
 |------------------------------ASP-Active Ack------>|
 |                         |                          |
```

5.1.1.4. Three ASPs in an Application Server ("n+k" sparing, load-sharing case)

This scenario shows the example TUA message flows for the establishment of traffic between an SG and three ASPs in the same Application Server, where two of the ASPs are brought to "active" and share the load. In this case, a minimum of two ASPs are required to handle the total traffic load (2+1 sparing).

```
SG                   ASP1                 ASP2                 ASP3
 |                    |                    |                    |
 |<------ASP Up-------|                    |                    |
 |-----ASP-Up Ack---->|                    |                    |
 |                    |                    |                    |
 |<--------------------------ASP Up-------|                    |
 |------------------------ASP-Up Ack---->|                    |
 |                    |                    |                    |
 |<---------------------------------------------ASP Up--------|
 |-------------------------------------------ASP-Up Ack---->|
 |                    |                    |                    |
 |                    |                    |                    |
 |<--ASP Act ---------|                    |                    |
 |----ASP-Act Ack---->|                    |                    |
 |                    |                    |                    |
 |<--------------------ASP Act  ----------|                    |
 |----------------------ASP-Act Ack----->|                    |
 |                    |                    |                    |
```

5.1.2. ASP Traffic Fail-over Examples

5.1.2.1. (1+1 Sparing, withdrawal of ASP, Back-up Override)

ASP1 withdraws from service:

```
SG                       ASP1                        ASP2
 |                         |                           |
 |<-----ASP Inactive-------|                           |
 |----ASP Inactive Ack---->|                           |
 |----------------------NTFY(ASP-Inact.)(Optional)-->|
 |                         |                           |
 |<------------------------------ ASP Active----------|
 |-----------------------------ASP-Active Ack------->|
 |                                                     |
```

Note: If the SG detects loss of the TUA peer (TUA heartbeat loss or detection of SCTP failure), the initial SG-ASP1 ASP Inactive message exchange would not occur.

5.1.2.2. (1+1 Sparing, Back-up Override)

ASP2 wishes to override ASP1 and take over the traffic:

```
SG                       ASP1                        ASP2
 |                         |                           |
 |<------------------------------ ASP Active----------|
 |-------------------------------ASP-Active Ack------>|
 |----NTFY(Alt ASP-Act)--->|
 |                         |                           |
```

5.1.2.3. (n+k Sparing, Load-sharing case, withdrawal of ASP)

ASP1 withdraws from service:

```
SG                   ASP1                 ASP2                 ASP3
 |                    |                    |                    |
 |<----ASP Inact.-----|                    |                    |
 |---ASP-Inact Ack--->|                    |                    |
 |                    |                    |                    |
 |---------------------------------NTFY(Ins. ASPs)(Optional)->|
 |                    |                    |                    |
 |<-----------------------------------------ASP Act ----------|
 |------------------------------------------ASP Act (Ack)--->|
 |                    |                    |                    |
```

The Notify message to ASP3 is optional, as well as the ASP-Active from ASP3. The optional Notify can only occur if the SG maintains knowledge of the minimum ASP resources required - for example if the SG knows that "n+k" = "2+1" for a load-share AS and "n" currently equals "1".

Note: If the SG detects loss of the ASP1 TUA peer (TUA heartbeat loss or detection of SCTP failure), the first SG-ASP1 ASP Inactive message exchange would not occur.

5.1.3. TCAP/TC-User Service Translation Examples

When the TUA layer on the ASP has a DH message to send to the SG, it will do the following:

(1) Determine the correct SGP

(2) Find the SCTP association to the chosen SGP

(3) Determine the correct stream in the SCTP association based on the DID

(4) Build the DH message, fill TUA Message Header, fill Common Header

(5) Send the DH message to the remote TUA peer in the SG, over the SCTP association

When the TUA layer on the SG has a DH message to send to the ASP, it will do the following:

(1) Determine the AS

(2) Determine the Active ASP (SCTP association) within the AS

(3) Determine the correct stream in the SCTP association based on the DID

(4) Build the DH message, fill in TUA Message Header, fill in Common Header

(5) Send the DH message to the remote TUA peer in the ASP, over the SCTP association

An example of the message flows for establishing a dialogue service is shown below. An active association between ASP and SG is established (Section 5.1) prior to the following message flows.

```
                SG                             ASP

                          <----------- Invoke Request
                          <----------- Query(Begin) Request

      Conversation(Continue)
          Indication      ---------->
      Result Indication   ---------->

                          <----------- Invoke Request
                          <----------- Conversation(Continue) Request
                              .
                              .
                              .
      End(response)Indication ----------->

          Result Indication ----------->
```

An example of the message flows for a failed attempt to establish a dialogue on the signalling channel is shown below. In this case, the gateway has a problem with its physical connection , so it cannot establish a dialogue on the signalling channel.

```
                SG                             ASP

                          <----------- Invoke Request
                          <----------- Query(Begin) Request

          Abort Indication   ---------->
```

5.2. IP-IP Architecture

The sequences below outline logical steps for a variety of scenarios within an IP-IP architecture. Please note that these scenarios cover a Primary/Backup configuration. Where there is a load-sharing configuration then the AS can declare availability when 1 ASP issues ASPAC but can only declare unavailability when all ASPs have issued ASPIA.

5.2.1. Establishment of TUA connectivity

The following shows an example establishment of TUA connectivity. In this example, each IP SP consists of a Management Instance (MI) and two ASPs. The Management Instance handles the address mapping mechanisms and monitors the states of the remote peer. For simplicity, the Management Instances and ASPs are considered as a separate entity. This is not a requirement, as they can be collocated with an ASP.

The following must be established before TUA traffic can flow. A connection-less flow is shown for simplicity.

Each node is configured (via MIB, for example) with the connections that need to be setup

```
IP SEP A                                                  IP SEP B
ASP-a1     ASP-a2     MI a             MI b       ASP-b2    ASP-b1
(Primary) (Backup)                               (Backup) (Primary)

            Establish SCTP Connectivity

                      |-- Est. SCTP Ass.--|

|------ Establish SCTP Association -------|
|------------- Establish SCTP Association -------------|
|------------------ Establish SCTP Association ------------------|

           |--- Establish SCTP Assoc. ----|
           |------- Establish SCTP Association --------|
           |------------ Establish SCTP Association -------------|

                      |-- Establish SCTP Association -|
                      |------- Establish SCTP Association ------|

            Establish TUA Connectivity

+---------------ASP Up------------------->
<---------------ASP Up Ack---------------+

           +------------ASP Up----------->
           <------------ASP Up Ack-------+

                     <--------------ASP Up-------------+
                     +--------------ASP Up Ack--------->

                     <----------------ASP Up---------------------+
                     +----------------ASP Up Ack----------------->

+---------------ASP Act------------------>
<---------------ASP Act Ack--------------+

                     <----------------ASP Act--------------------+
                     +----------------ASP Act Ack---------------->

Traffic can now flow directly between ASPs.

+-------------------------------TCAP_User Message------------------>
```

5.2.2. Fail-over scenarios

The following sequences address fail-over of ASP

5.2.2.1. Successful ASP Fail-over scenario

The following is an example of a successful fail-over scenario, where there is a fail-over from ASP-a1 to ASP-a2, i.e, Primary to Backup. Since data transfer passes directly between peer ASPs, ASP-b1 is notified of the fail-over of ASP-a1 and must buffer outgoing data messages until ASP-a2 becomes available.

```
IP SEP A                                                  IP SEP B
ASP-a1     ASP-a2     MI a              MI b       ASP-b2    ASP-b1
(Primary) (Backup)                                (Backup) (Primary)
+--------------ASP Inact----------------->
<--------------ASP Inact Ack-------------+

           <----NTFY (ASP-a1 Inactive)---+

           +----------ASP Act------------>
           <----------ASP Act Ack--------+
```

5.2.2.2. Unsuccessful ASP Fail-over scenario

The sequence is the same as 5.2.2.1 except that, since the backup fails to come in then, the Notify messages declaring the availability of the backup are not sent.

6. Security

6.1. Introduction

TUA is designed to carry signalling messages for telephone services. As such, TUA involves the security needs of several parties: the end users of the services; the network providers and the applications involved. Additional security requirements may come from local regulation. While having some overlapping security needs, any security solution should fulfill all of the different parties' needs.

6.2. Threats

There is no quick fix, one-size-fits-all solution for security. As a transport protocol, TUA has the following security objectives:

- Availability of reliable and timely user data transport.
- Integrity of user data transport.
- Confidentiality of user data.

TUA runs on top of SCTP. SCTP provides certain transport related security features, such as:

- Blind Denial of Service Attacks
- Flooding
- Masquerade
- Improper Monopolization of Services

When TUA is running in professionally managed corporate or service provider network, it is reasonable to expect that this network include an appropriate security policy framework. The "Site Security Handbook" [2196] should be consulted for guidance.

When the network in which TUA runs in involves more than one party, it may not be reasonable to expect that all parties have implemented security in a sufficient manner. End-to-end security should be the goal; therefore, it is recommended that IPSEC be used to ensure confidentiality of user payload. Consult [2409] for more information on configuring IPSEC services.

6.3. Protecting Confidentiality

Particularly for mobile users, the requirement for confidentiality may include the masking of IP addresses and ports. In this case application level encryption is not sufficient; IPSEC ESP should be used instead. Regardless of which level performs the encryption, the IPSEC ISAKMP service should be used for key management.

7. IANA Considerations

7.1. SCTP Payload Protocol ID

IANA has assigned a TUA value for the Payload Protocol Identifier in the SCTP DATA chunk. The following SCTP Payload Protocol Identifier is registered:

TUA "4"

The SCTP Payload Protocol Identifier value "4" **SHOULD** be included in each SCTP DATA chunk, to indicate that the SCTP is carrying the TUA protocol. The value "0" (unspecified) is also allowed but any other values **MUST NOT** be used. This Payload Protocol Identifier is not directly used by SCTP but **MAY** be used by certain network entities to identify the type of information being carried in a DATA chunk.

The User Adaptation peer **MAY** use the Payload Protocol Identifier, as a way of determining additional information about the data being presented to it by SCTP. A request will be made to IANA to assign CTP Payload Protocol IDs.

7.2. Port Number

IANA has registered SCTP Port Number 14001 for TUA. It is recommended that SGPs use this SCTP port number for listening for new connections. SGPs **MAY** also use statically configured SCTP port numbers instead.

7.3. Protocol Extensions

This protocol may also be extended through IANA in three ways:

- Through definition of additional message classes.
- Through definition of additional message types.
- Through definition of additional message parameters.

The definition and use of new message classes, types and parameters is an integral part of SIGTRAN adaptation layers. Thus, these extensions are assigned by IANA through an IETF Consensus action as defined in[RFC 2434].

The proposed extension **MUST** in no way adversely affect the general working of the protocol.

A new registry will be created by IANA to allow

7.3.1. IETF Defined Message Classes

The documentation for a new message class **MUST** include the following information:

(1) A long and short name for the message class;

(2) A detailed description of the purpose of the message class.

7.3.2. IETF Defined Message Types

Documentation of the message type **MUST** contain the following information:

(1) A long and short name for the new message type;

(2) A detailed description of the structure of the message.

(3) A detailed definition and description of intended use of each field within the message.

(4) A detailed procedural description of the use of the new message type within the operation of the protocol.

(5) A detailed description of error conditions when receiving this message type.

When an implementation receives a message type which it does not support, it **MUST** respond with an Error (ERR) message, with an Error Code = Unsupported Message Type.

7.3.3. IETF-defined TLV Parameter Extension

Documentation of the message parameter **MUST** contain the following information:

(1) Name of the parameter type.

(2) Detailed description of the structure of the parameter field. This structure **MUST** conform to the general type-length-value format described earlier in the document.

(3) Detailed definition of each component of the parameter value.

(4) Detailed description of the intended use of this parameter type, and an indication of whether and under what circumstances multiple instances of this parameter type may be found within the same message type.

8. Acknowledgments

The authors would like to thank Jianxing Hou, Min Lin for their original input to this document, and to the authors of M2UA, M3UA and SUA for the large sections of text which apply also to TUA and was included here.

9. Author's Addresses

Brian Bidulock
OpenSS7 Corporation
4701 Preston Park Boulevard
Suite 424
Plano, TX 75093
USA

Phone: +1-972-839-4489
Email: bidulock@openss7.org
URL: http://www.openss7.org/

This draft expires July 2002.

End Notes

[1] **IMPLEMENTATION NOTE:–** Only one SCTP port may be defined for each endpoint, but each SCTP endpoint may have multiple IP addresses[RFC 2960].

[2] **IMPLEMENTATION NOTE:–** Where more than one route (or SG) is possible for routing to the SS7 network, the ASP could, for example, maintain a dynamic table of available SG routes for the SS7 destinations and subsystems, taking into account the destination and subsystem availability and congestion status received from the SG(s), the availability status of individual SGs and configuration changes or fail-over mechanisms.

[3] **IMPLEMENTATION NOTE:–** When the TC-User selects sequenced delivery using the "Sequence Control" fields in the Quality of Service parameter, the DH message **SHOULD** be sent on an SCTP stream using ordered delivery. When the TC-User does not select sequenced delivery and does not utilize the optional component handling interface (i.e. the DH message has components included), the DH message **MAY** be sent on an SCTP stream using unordered delivery.

[4] **IMPLEMENTATION NOTE:–** The use of TLV in principle allows the parameters to be placed in a random order in the message. However, some guidelines should be considered for easy processing in the following order:

(1) parameters needed to correctly process other message parameters, preferably should precede these parameters (such as Routing Context), mandatory parameters preferably should precede any optional parameters,

(2) the data parameter will normally be the final one in the message.

[5] **IMPLEMENTATION NOTE:–** An Application Server Process may be configured to process traffic for more than one logical Application Server. From the perspective of an ASP, a Routing Context defines a range of signalling traffic that the ASP is currently configured to receive from the SG.

Additionally, the Routing Context parameter identifies the SS7 network context for the message, for the purposes of logically separating the signalling traffic between the SGP and the Application Server Process over a common SCTP Association, when needed. An example is where an SGP is logically partitioned to appear as an element in several different national SS7 networks. It implicitly defines the SS7 Point Code format used, the SS7 Network Indicator value and TCAP protocol type/variant/version used within a separate SS7 network. It also defines the network context for the PC and SSN values. Where an SGP operates in the context of a single SS7 network, or individual SCTP associations are dedicated to each SS7 network context, this functionality is not needed.

[6] **IMPLEMENTATION NOTE:–** Correlation Id parameter can be used for features like Synchronization of ASPs and SGPs in a Broadcast Mode AS or SG; avoid message duplication and mis-sequencing in case of fail-over of association from one ASP or SGP to another ASP or SGP, etc.

[7] **IMPLEMENTATION NOTE:–** The value in the Importance field in the Quality of Service parameter **MAY** be ignored or modified by a Signalling Gateway if the value contained is not consistent with SCCP flow control policy at the SG.

[8] **IMPLEMENTATION NOTE:–** The value in the Message Priority field in the Quality of Service parameter **MAY** be ignored or modified by a Signalling Gateway if the value contained is not consistent with MTP congestion policy at the SG.

IMPLEMENTATION NOTE:– The Signalling Gateway **MAY**, at its option, segment the *Parameters* field into multiple parameters set to be send in multiple Invoke (Last/Not Last) or Return Result (Last/Not Last) components in separate TCAP packages to meet the maximum PDU requirements imposed by the underlying SCCP transport. Otherwise, if the Signalling Gateway finds that the resulting component is too large to fit into an SCCP UNITDATA message[Q.713], the SG **MAY**, at its option, return a TNOT message indicating to the TC-User that the component was too large.

[10] **IMPLEMENTATION NOTE:–** To accomplish the handling of SSNM messages from multiple SGs in a multiple SG configuration, the TUA layer at an ASP maintains the status of routes via each SG.

[11] **IMPLEMENTATION NOTE:–** In the case of a Periodic audit, the auditing procedure might not be invoked for the case of a received SCON message containing a congestion level value of "no congestion" or undefined" (i.e., congestion Level = "0"). This is because the value indicates either congestion abatement or that the ITU MTP3 international congestion method is being used. In the international congestion method, the MTP3 layer at the SGP does not maintain the congestion status of any destinations and therefore the SGP cannot provide any congestion information in response to the DAUD. For the same reason, in the second of the cases above a DAUD message cannot reveal any congested destination(s).

[12] **IMPLEMENTATION NOTE:–** For example, an SGP may chose to not respond to a request for the destination or subsystem status of a specific point code in the DAUD message because the ASP that issued the DAUD message is not authorized to obtain information concerning the status of the destination as requested.

[13] **IMPLEMENTATION NOTE:–** Typically a TC-User is responsible for performing the segmentation and reassembly of components.

References

RFC 2960.
R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, H. Kalla, L. Zhang, V. Paxson, “Stream Control Transmission Protocol (SCTP),” RFC 2960, The Internet Society (February 2000).

Q.771. ITU, “Signalling System No. 7 – Functional Description of Transaction Capabilities,” ITU-T Recommendation Q.771, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

T1.114.
ANSI, “Signalling System No. 7 – Transaction Capabilities Application Part,” ANSI T1.114, American National Standards Institue (1992).

RFC 2719.
L. Ong, I. Rytina, M. Holdrege, L. Coene, M.-A. Garcia, C. Sharp, I. Juhasz, H. P. Lin, HannsJ. Schwarzbauer, “Framework Architecture for Signaling Transport,” RFC 2719, The Internet Society (October, 1999).

Q.701. ITU, “Functional Description of the Message Transfer Part (MTP) of Signalling System No. 7,” ITU-T Recommendation Q.701, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

T1.111.
ANSI, “Signalling System No. 7 – Message Transfer Part,” ANSI T1.111, American National Standards Institue (1992).

Q.711. ITU, “Functional Description of Signalling Connection Control Part,” ITU-T Recommendation Q.711, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

RFC 2916.
P. Falstrom, “E.164 number and DNS (ENUM),” RFC 2916, The Internet Society (September 2000).

Q.704. ITU, “Message Transfer Part – Signalling Network Functions and Messages,” ITU-T Recommendation Q.704, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

SUA. J. Loughney, G. Sidebottom, G. Mousseau, S. Lorusso, L. Coene, G. Verwimp, J. Keller, F. E. Gonzalez, W. Sully, S. Furniss, B. Bidulock, “SS7 SCCP-User Adaptation Layer (SUA),” <draft-ietf-sigtran-sua-09.txt>, Internet

Engineering Task Force - Signalling Transport Working Group (June 15, 2001). Work In Progress.

M3UA. G. Sidebottom, J. Pastor-Balbes, I. Rytina, G. Mousseau, L. Ong, H. J. Schwarzbauer, K. Gradischnig, K. Morneault, M. Kalla, N. Glaude, B. Bidulock, N. Glaude, “SS7 MTP3-User Adaptation Layer (M3UA),” <draft-ietf-sigtran-m3ua-10.txt>, Internet Engineering Task Force - Signalling Transport Working Group (November, 2001). Work In Progress.

Q.705. ITU, “Signalling System No. 7 – Signalling Network Structure,” ITU-T Recommendation Q.705, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

T1.112.
ANSI, “Signalling System No. 7 – Signalling Connection Control Part,” ANSI T1.112, American National Standards Institue (1992).

RFC 2119.
S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119 - BCP 14, Internet Engineering Task Force (March 1997).

Q.773. ITU, “Signalling System No. 7 – Transaction Capabilities Formats and Encoding,” ITU-T Recommendation Q.773, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

Q.775. ITU, “Signalling System No. 7 – Guidelines for Using Transaction Capabilities,” ITU-T Recommendation Q.775, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

Q.714. ITU, “Signalling Connection Control Part Procedures,” ITU-T Recommendation Q.714, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

X.680. ITU, “Abstract Syntax Notation One (ASN.1): Specification of Basic Notation,” ITU-T Recommendation X.680, ITU-T Telecommunication Standardization Sector of ITU, Geneva (July 1994). (Previously "CCITT Recommendation")

Q.713. ITU, “Signalling Connection Control Part Formats and Codes,” ITU-T Recommendation Q.713, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

RFC 2434.
T. Narten, H. T. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” RFC 2434, The Internet Society (October, 1998).

Appendices

A. Operational Considerations

A.1. Signalling Network Architecture

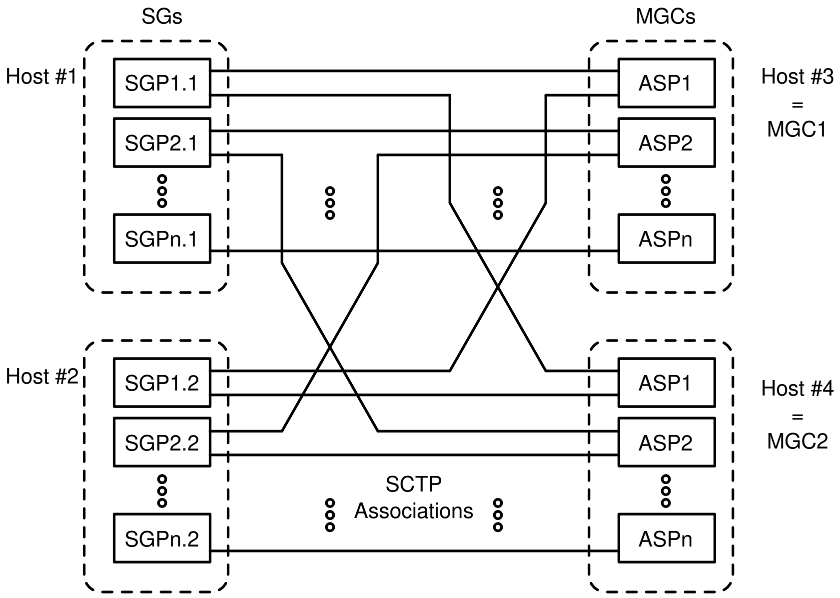
A Signalling Gateway is used to support the transport of TC-User signalling traffic received from the SS7 network to multiple distributed ASPs (e.g., MGCs and IP Databases). Clearly, the TUA protocol is not designed to meet the performance and reliability requirements for such transport by itself. However, the conjunction of distributed architecture and redundant networks provides support for reliable transport of signalling traffic over IP. The TUA protocol is flexible enough to allow its operation and management in a variety of physical configurations, enabling Network Operators to meet their performance and reliability requirements.

To meet the stringent SS7 signalling reliability and performance requirements for carrier grade networks, Network Operators might require that no single point of failure is present in the end-to-end network architecture between an SS7 node and an IP-based application. This can typically be achieved through the use of redundant SGPs or SGs, redundant hosts, and the provision of redundant QOS-bounded IP network paths for SCTP Associations between SCTP End Points. Obviously, the reliability of the SG, the MGC and other IP-based functional elements also needs to be taken into account. The distribution of ASPs and SGPs within the available Hosts MAY also be considered. As an example, for a particular Application Server, the related ASPs could be distributed over at least two Hosts.

One example of a physical network architecture relevant to SS7 carrier-grade operation in the IP network domain is shown in *Figure 7*.

In this model, each host MAY have many application processes. In the case of the MGC, an ASP may provide service to one or more Application Servers, and is identified as an SCTP end point. One or more Signalling Gateway Processes make up a single Signalling Gateway.

This example model can also be applied to IPSP-IPSP signalling. In this case, each IPSP MAY have its services distributed across 2 hosts or more, and may have multiple server processes on each host.



SGP1.1 and SGP1.2 are part of SG1
SGP2.1 and SGP2.2 are part of SG2

Figure 7. Physical Model

In the example above, each signalling process (SGP, ASP or IPSP) is the end point to more than one SCTP association, leading to more than one other signalling processes. To support this, a signalling process must be able to support distribution of TUA messages to many simultaneous active associations. This message distribution function is based on the status of provisioned Routing Keys, the status of the signalling routes to signalling points in the SS7 network , and the redundancy model (override, loadsharing, broadcast) of the remote signalling processes.

For carrier grade networks, the failure or isolation of a particular signalling process should not cause transactions to be lost. This implies that signalling processes need, in some cases, to share the transaction state or be able to pass the transaction state information between each other. However, this sharing or communication of transaction state information is outside the scope of this document.

This model serves as an example. TUA imposes no restrictions as to the exact layout of the network elements, the message distribution algorithms and the distribution of the signalling processes. Instead, it provides a framework and a set of messages that allow for a flexible and scalable signalling network architecture, aiming to provide reliability and performance.

A.2. Redundancy Models

A.2.1. Application Server Redundancy

At the SGP, an Application Server list contains active and inactive ASPs to support ASP broadcast, loadsharing and override procedures. The list of ASPs within a logical Application Server is kept updated in the SGP to reflect the active Application Server Processes.

For example, in the network shown in *Figure 7*, all messages to SSN x could be sent to ASP1 in Host3 or ASP1 in Host4. The AS list at SGP1 in Host 1 might look like the following:

```
Routing Key {SSN=x) - "Application Server #1"
        ASP1/Host3      -   State = Active
        ASP1/Host4      -   State = Inactive
```

In this "1+1" redundancy case, ASP1 in Host3 would be sent any incoming message with SSN=x. ASP1 in Host4 would normally be brought to the "active" state upon failure of, or loss of connectivity to, ASP1/Host1.

The AS List at SGP1 in Host1 might also be set up in loadshare mode:

```
Routing Key {SSN=x) - "Application Server #1"
        ASP1/Host3      -   State = Active
        ASP1/Host4      -   State = Active
```

In this case, both the ASPs would be sent a portion of the traffic. For example the two ASPs could together form a database, where incoming queries may be sent to any active ASP.

Care might need to be exercised by a Network Operator in the selection of the routing information to be used as the Routing Key for a particular AS.

For example, where Application Servers are defined using ranges of GT Address values, the Operator is implicitly splitting up control of the related address groups. Some GT address value range assignments may interfere with TCAP subsystem management procedures.

In the process of failover, it is recommended that in the case of ASPs that transactions do not fail. For example, the two ASPs may share transaction state via shared memory, or may use an ASP to ASP protocol to pass transaction state information. Any ASP-to-ASP protocol to support this function is outside the scope of this document.

A.2.2. Signalling Gateway Redundancy

Signalling Gateways may also be distributed over multiple hosts. Much like the AS model, SGs may comprise one or more SG Processes (SGPs), distributed over one or more hosts, using an override, loadshare or broadcast model. Should an

SGP lose all or partial SS7 connectivity and other SGPs exist, the SGP may terminate the SCTP associations to the concerned ASPs or send an unsolicited ASPIA ACK for the concerned Application Servers.

It is possible for an ASP to route signalling messages destined to the SS7 network using more than one SGP. In this model, a Signalling Gateway is deployed as a cluster of hosts acting as a single SG. An override redundancy model is possible, where the unavailability of the SCTP association to a primary SGP could be used to reroute affected traffic to an alternate SGP. A loadsharing model is possible, where the signalling messages are loadshared between multiple SGPs. A broadcast model is also possible, where signalling messages are sent to each active SGP in the SG. The distribution of the TC-user messages over the SGPs should be done in such a way to minimize message missequencing, as required by the SS7 User Parts.

It may also be possible for an ASP to use more than one SG to access a specific SS7 end point, in a model that resembles an SS7 STP mated pair. Typically, SS7 STPs are deployed in mated pairs, with traffic loadshared between them. Other models are also possible, subject to the limitations of the local SS7 network provisioning guidelines.

From the perspective of the TUA layer at an ASP, a particular SG is capable of transferring traffic to a provisioned SS7 destination, subsystem or application X if an SCTP association with at least one SGP of the SG is established, the SGP has returned an acknowledgement to the ASP to indicate that the ASP is actively handling traffic for that destination, subsysstem or application X, and the SGP has not indicated that the destination, subsystem or application X is inaccessible. When an ASP is configured to use multiple SGPs for transferring traffic to the SS7 network, the ASP must maintain knowledge of the current capability of the SGPs to handle traffic to destinations, subsystems and applications of interest. This information is crucial to the overall reliability of the service, for override, loadsharing and broadcast models, in the event of failures, recovery and maintenance activities. The ASP TUA may also use this information for congestion avoidance purposes. The distribution of the TC-user messages over the SGPs should be done in such a way to minimize message missequencing, as required by the some TCAP applications.

List of Tables

List of Illustrations

Table of Contents

Copyright Statement