Network Working Group Brian Bidulock
INTERNET-DRAFT OpenSS7 Corporation

Expires in six months January 5, 2003

SS7 ISUP-User Adaptation Layer
ISUA
<draft-bidulock-sigtran-isua-00.ps>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 or RFC 2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as 'work in progress'.

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

To learn the current status of any Internet-Draft, please check the ´1id-abstracts.txt´ listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This document defines a protocol for the transport of any SS7 ISUP-User signalling (e.g, Call Control) 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.

Contents

A complete table of contents appears the end of this document.

1. Introduction

This draft defines a protocol for the transport of SS7 ISUP [Q.761, T1.113] Users (i.e, Call Control) 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 ISUP 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 Call Control messages 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 Call Control messages between two end-points wholly contained within and IP network.

The delivery mechanism addresses the following criteria:

- Support for transfer of ISUP messages (Call Control)
- Support for the seamless operation of Call Control 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. Change History

EDITOR'S NOTE:– This change history section will be deleted if and when the draft is advanced.

1.2.1. Version 0.0

This is the initial version of this document.

1.3. 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 one-to-one 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 ISUP protocol data units and ISUA layer peer messages.

Call Control – The layer above the ISDN User Part in the SS7 protocol stack that exchanges primitives with the ISUP provider. Call Control has two major functional blocks: Call Processing and Circuit Supervision. Unlike other layers of the SS7 stack, ISUP does not have individual "Users" or ISUP-SAPs. A single Call Control entity is responsible for controlling both ISUP and other switch signalling stacks at the Application Layer of the ISO 7-layer model. for

Call Processing] – Call Processing is a major functional block of both ISUP and Call Control which is responsible for signalling and controlling the state of calls (as opposed to circuits).

Circuit Supervision] – Circuit Supervision is a major functional block of both ISUP and Call Control which is responsible for signalling and controlling the state of circuits (as opposed to calls).

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 ISUA in a point-to-point fashion.

ISDN User Part (ISUP) – The Integrated Services Digital Network (ISDN) User Part [Q.761, T1.113] of the SS7 protocol.

Layer Management (LM) – a nodal function that handles the inputs and outputs between the ISUA 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 ISUP and ISUA 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 ISUA.

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 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.

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

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].

1.4. ISUA Overview

1.4.1. 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 ISUA architecture can be modeled as a peer-to-peer architecture. The first section considers the SS7-to-IP interworking architectures for ISUP call control. For this case, it is assumed that the ASP initiates the establishment of the SCTP association with the SG.

1.4.2. Protocol Architecture for Call Control

In this architecture (illustrated in *Figure 1*), the ISUP and ISUA layers interface in the SG. A Nodal Interworking Function (NIF) provides for interworking between the ISUP and ISUA layers and provides for the transfer of the call processing as well as circuit supervision messages.

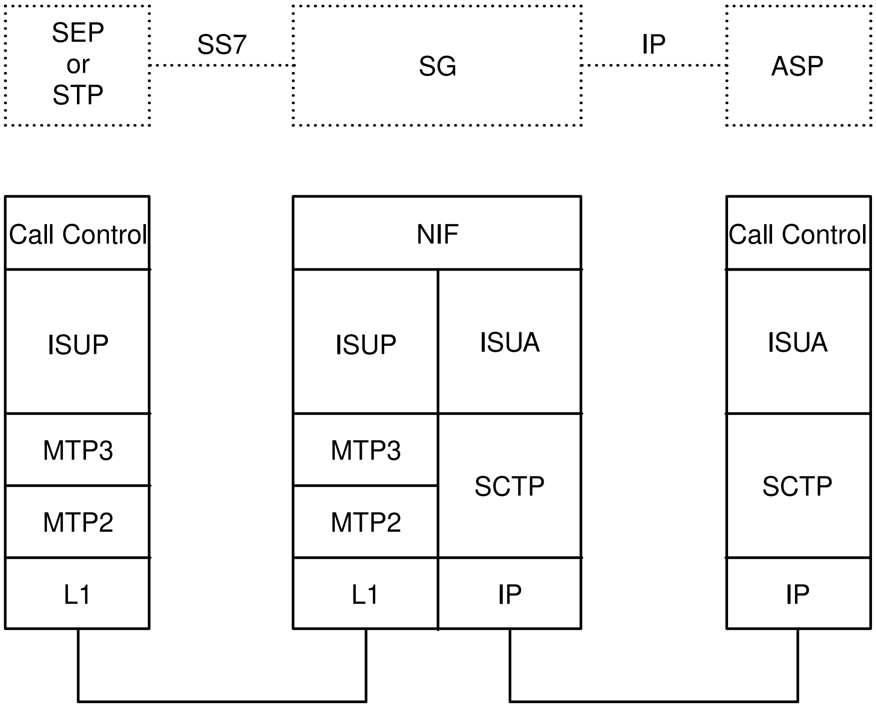

Figure 1. Protocol Architecture

1.4.3. All IP Architecture

This architecture, illustrated in *Figure 2*, can be used to carry a protocol which uses the transport services of ISUP, 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.

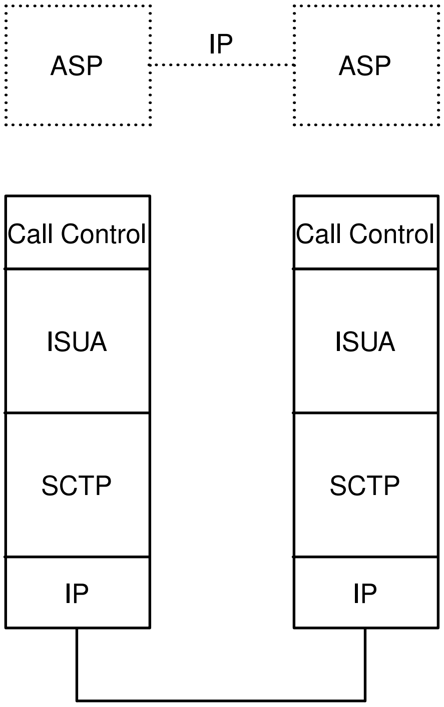

Figure 2. All IP Architecture

1.4.4. ASP Fail-over Model and Terminology

The ISUA 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 Call Control 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.

1.4.5. Services Provided by the ISUA Layer

1.4.5.1. Support for the transport of Call Control Messages

The ISUA supports the transfer of Call Control messages. The ISUA layer at the SG and the ASP support the seamless transport of user messages between the SG and the ASP.

1.4.5.1.1. ISUP Call Control Support

Depending on the specific implementation of Call Control supported, the ISUA shall support Call Control transparently. Call Control consists of two major functional blocks:

Call Processing is responsible for signalling and control of calls (as opposed to circuits). Call processing functions move a call through its life-cycle by providing the following functions:

- call setup,
- call suspend/resume,
- call release,
- call exception handling.

Circuit Supervision is responsible for signalling and control of circuits (as opposed to calls). Circuit supervision functions affect the management state of circuits and provides the following functions:

- circuit testing,
- circuit reset,
- circuit blocking and unblocking due to hardware failure and recovery events,
- circuit blocking and unblocking maintenance action,
- circuit state query.

1.4.5.2. Native Management Functions

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

1.4.5.3. Interworking with Circuit Supervision Functions

The ISUA layer provides interworking with Circuit Supervision functions at the SG for seamless inter-operation between the SCN network and the IP network. ISUA provides the following circuit supervision functions:

- Provides an indication or accpets a request to perform a continuity check on a circuit.
- Provides an indication or accepts a request to reset a circuit or circuit group.
- Provides an indication or accepts a request to block a circuit or circuit group.
- Provides an indication or accepts a request to unblock a circuit or circuit group.
- Provides an indication or accepts a request to query a circuit or circuit group.

The interworking with ISUP circuit supervision messages consists of *CCNT*, *CCNA*, *CREP*, *CRSC*, *CBLO*, *CBLA*, *CUBL*, *CUBA*, *CQRY* and *CQRA* messages on receipt of circuit supervision events to the appropriate ASPs. The primitives in *Table 1* are sent between the ISUP and ISUA circuit supervision functions in the SG to trigger events in the IP and SS7 domain.

The ISUA layer provides transparent passing of circuit reset, blocking and query primitives (RESET, BLOCKING, UNBLOCKING, CCT GROUP QUERY) as provided for in ITU-T Q.724 [Q.724] Q.764 [Q.764], and ANSI T1.113 [T1.113].

Table 1. Mapping of Circuit Supervision Primitives

Name		Message		ISUA Cc’t Supv. Message
Generic [2]	**Specific**	**ITU-T Q.764**	**ANSI T1.113**	
CONT RECHECK	Request Indication	CCR	CCR	CCNT
	Response Confirm	–	LPA	CCNA
CONT REPORT	Request Indication	COT	COT	CREP
RESET	Request	RSC, GRS	RSC, GRS	CRSC
	Confirm	RLC, GRA	RLC, GRA	CRSA
BLOCKING	Request Indication	BLO, CGB	BLO, CGB	CBLO
	Response Confirm	BLA, CGBA	BLA, CGBA	CBLA
UNBLOCKING	Request Indication	UBL, CGU	UBL, CGU	CUBL
	Response Confirm	UBA, CGUA	UBA, CGUA	CUBA
CCT GRP QUERY	Request Indication	CQM	CQM	CQRY
	Response Confirm	CQR	CQR	CQRA

1.4.5.4. Support for the Management of SCTP Associations

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

The ISUA layer **MAY** be instructed by local management to establish an SCTP association to a peer ISUA 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 ISUA node. To avoid redundant SCTP associations between two ISUA peers, one side (client) **SHOULD** be designated to establish the SCTP association, or ISUA 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 ISUA layer the status of the underlying SCTP associations using the M-SCTP_STATUS request and confirm primitives. Also, the ISUA **MAY** autonomously inform local management of the reason for the release of an SCTP association, determined either locally within the ISUA layer or by a primitive from the SCTP.

Also, the ISUA 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.

1.5. Functional Areas

1.5.1. Circuit Identifiers, Routing Contexts and Routing Keys

1.5.1.1. Overview

The mapping of ISUP messages into calls between the SGP and the Application Servers is determined by Circuit Identifiers, Routing Keys and their associated Routing Contexts.

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

Possible ISUP address/routing information that comprise a Routing Key entry includes, for example, a local and remote Point Code, and a Circuit Identification Code or Call Control specific information such as Circuit Group or Trunk Group

Identifiers. The particular information used to define a ISUA Routing Key is application and network dependent, and none of the above examples are requirements for ISUA.

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.

1.5.1.2. Routing Key Limitations

Routing Keys **SHOULD** be unique in the sense that each received ISUP 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 call processing for multiple ranges of circuits that are not represented by contiguous Circuit Identification Codes.

1.5.1.3. 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 ISUA Routing Key registration procedures.

When using a management interface to configure Routing Keys, the Circuit 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.

1.5.1.4. Circuit Mapping Function

To perform its addressing and relaying capabilities, the ISUA makes use of an Circuit Mapping Function (CMF). This function is considered part of ISUA, but the way it is realized is left implementation or deployment dependent (local tables, database, etc.)

The CMF is invoked when a message is received at the incoming interface. The CMF is responsible for resolving the Circuit Identification Code (CIC) and any necessary ISUP message parameters presented in the incoming ISUP message to SCTP associations and destinations within the IP network. The CMF 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 ISUA ASP management messages.

Possible SS7 routing information that comprise a Routing Key entry includes, for example, ISUP Circuit Identification Code (CIC), Range and Status parameters.

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

1.5.1.4.1. Circuit Mapping at the SG

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

To support this circuit mapping, the SGP might, for example, maintain the equivalent of a network address translation table, mapping incoming ISUP message information to an Application Server for a particular application and set of transactions. This could be accomplished by comparing the circuit identification code and range and status portions of the incoming ISUP 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, call handling capabilities and congestion to the SGP using various management messages defined in the ISUA protocol.

The list of ASPs in the AS is assumed to be dynamic, taking into account the availability, call 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 calls) 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 ISUP call clearing 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 call processing and circuit supervision messges to a (set of) default ASP(s), or to drop the messages and provide a notification to management. The treatment of unallocated circuits is implementation dependent.

1.5.1.4.2. Circuit Mapping at the ASP

To direct messages to the SS7 network, the ASP **MAY** perform a circuit mapping to choose the proper SGP for the given message. This is accomplished by observing the Circuit Identification Code, Range and Status, 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. There is, however, no ISUA 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 call control 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.

1.5.1.5. Signalling Gateway SS7 Layers

The SG is responsible for terminating up to the Call Control 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 ISUP messages to and from the SS7 Network over standard SS7 network interfaces, using the services of the MTP [Q.704] to provide transport of the messages.

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

The ISUP-SAP through which ISUA at the SG obtains its services could reside at a Signalling Transfer Point (STP) or Signalling End Point (SEP) [Q.701].

1.5.1.6. SS7 and ISUA 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 ISUA adaptation layer. It allows the ISUP application to exchange call control messages with an IP-based Application Server Process where the peer Call Control protocol layer exists.

To perform ISUP circuit supervision, it is required that the Call Control protocols at ASPs receive indications of circuit state, as well as call state as they would be expected by an SS7 ISUP application. To accomplish this, the RESET, BLOCKING, UNBLOCKING and CCT GROUP QUERY primitives received at the ISUP upper layer interface at the SG need to be propagated to the remote Call Control lower layer interface at the ASP.

ISUP call processing and circuit supervision mesages (such as BLO, BLA, CGB, CGBA) received from the SS7 network **MUST NOT** be encapsulated. The SG **MUST** terminate these messages and generate ISUA message as appropriate.

1.5.1.7. Application Server

A cluster of Application Servers is responsible for providing the overall support for one ore more SS7 upper layers. From an ISUP standpoint, Call Control provides complete support for the upper layer service for given Circuits or Trunk Groups. As an example, Call Control could provide complete support for Central Office Call Control for a given point code.

1.5.1.8. SCTP Stream Mapping

The ISUA supports SCTP streams. The SG and AS need to maintain a list of SCTP and Call Control for mapping purposes. Call Control requiring sequenced message transfer need to be sent over a stream using sequenced delivery.

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

All ISUA Circuit Supervision (CS) messages **MAY** select unordered delivery, depending on the requirements of Call Control. Normally one stream is used to send ISUA Circuit Supervision (CS) messages between peers, regardless of Application Server.

All Call Processing (CP) messages **MUST** be sent using ordered delivery. All Call Processing (CP) messages relating to the same call **MUST** be sent on the same stream as other Call Processing (CP) messages relating to the same call. The stream selected is based upon the Call Reference given by the Call Control over the primitive interface and other traffic information available to the SGP or ASP.

1.5.2. Redundancy Models

1.5.2.1. Application Server Redundancy

All CSET and Circuit Supervision (CS) messages (e.g, SETUP, RESET, BLOCKING) 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.

1.5.3. 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.

1.5.4. Congestion Management

The ISUA 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 ISUA layer indicates congestion to local Call Control by means of an appropriate ISUP primitive, as per current ISUP 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 ISUP [Q.764, T1.113] or MTP [Q.704, T1.111] standard. (The triggering of SS7 Management messages from an SG is an implementation-dependent function.)

1.6. Definition of ISUA Boundaries

ISUA has three protocol boundaries: an upper boundary between ISUA and Call Control; a lower boundary between ISUA and SCTP; and a layer management boundary between ISUA and the Layer Management Function. *Figure 3* illustrates the ISUA protocol boundaries.

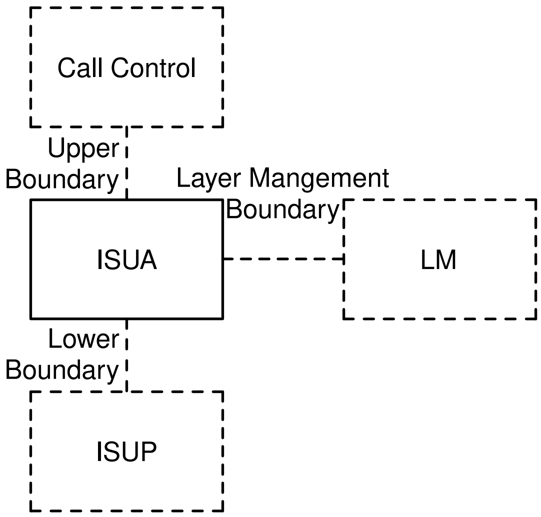

Figure 3. ISUA Protocol Boundaries

1.6.1. Definition of Upper Boundary

The primitives and messages listed in *Table 2* are provided between the ISUA and Call Control in support of Call Control [Q.761, T1.113].

Table 2. Mapping of Call Control Primitives

Generic Name	Specific Name	ITU-T Q.764 Message	ANSI T1.113 Message	ISUA Msg
Call Setup Messages				
SETUP	Request Indication	IAM	IAM	CSET
	Response Confirm	ANM, CON	ANM	CCON
MORE INFO	Request Indication	–	–	CMOR
TIMEOUT	Indication	–	–	CTOT
INFO	Request Indication	SAM	SAM	CINF
PROC	Request Indication	ACM, CPG	ACM, CPG	CPRO
ALERT	Request Indication	ACM, CPG	ACM, CPG	CALR
PROG	Request Indication	ACM, CPG	ACM, CPG	CPRG
Call Established Messages				
SUSPEND	Request Indication	SUS	SUS	CSUS
RESUME	Request Indication	RES	RES	CRES
Call Termination Messages				
REATTEMPT	Indication	–	–	CREA
CALL FAILURE	Indication	RST, REL, RLC	RST, REL, RLC	CERR
IBI	Request Indication	ACM, CPG	ACM, CPG	CIBI
RELEASE	Request Indication	REL	REL	CREL
	Response Confirm	REL, RLC	REL, RLC	CRLC

1.6.2. Definition of Boundary between ISUA and Layer Management

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

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

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

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

M-SCTP_RELEASE confirm

Direction: ISUA -> LM
Purpose: ASP confirms to LM that it has released SCTP association with its peer.

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

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

M-SCTP_STATUS request
Direction: LM -> ISUA
Purpose: LM requests ISUA to report the status of an SCTP association.

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

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

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

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

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

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

M-NOTIFY indication
Direction: ISUA -> LM
Purpose: ISUA reports that it has received a *Notify (NTFY)* message from its peer.

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

M-ASP_UP request
Direction: LM -> ISUA
Purpose: LM requests ASP to start its operation and send an *ASP Up (ASPUP)* message to its peer.

M-ASP_UP confirm
Direction: ISUA -> LM

Purpose: ASP reports that is has received an *ASP UP Ack (ASPUP ACK)* message from its peer. T} ; ls l1lw(5.7i). M-ASP_UP indication Direction:;ISUA -> LM Purpose:;T{ ISUA reports it has successfully processed an incoming *ASP Up (ASPUP)* message from its peer.

M-ASP_DOWN request
Direction: LM -> ISUA
Purpose: LM requests ASP to stop its operation and send an *ASP Down (ASPDN)* message to its peer.

M-ASP_DOWN confirm
Direction: ISUA -> LM
Purpose: ASP reports that is has received an *ASP Down Ack (ASPDN ACK)* message from its peer.

M-ASP_DOWN indication
Direction: ISUA -> LM
Purpose: ISUA reports it has successfully processed an incoming *ASP Down (ASPDN)* message from its peer, or the SCTP association has been lost or reset.

M-ASP_ACTIVE request
Direction: LM -> ISUA
Purpose: LM requests ASP to send an *ASP Active (ASPAC)* message to its peer.

M-ASP_ACTIVE confirm
Direction: ISUA -> LM
Purpose: ASP reports that is has received an *ASP Active Ack (ASPAC ACK)* message from its peer.

M-ASP_ACTIVE indication
Direction: ISUA -> LM
Purpose: ISUA reports it has successfully processed an incoming *ASP Active (ASPAC)* message from its peer.

M-ASP_INACTIVE request
Direction: LM -> ISUA
Purpose: LM requests ASP to send an *ASP Inactive (ASPIA)* message to its peer.

M-ASP_INACTIVE confirm
Direction: LM -> ISUA
Purpose: ASP reports that is has received an *ASP Inactive Ack (ASPIA ACK)* message from its peer.

M-ASP_INACTIVE indication
Direction: ISUA -> LM
Purpose: ISUA reports it has successfully processed an incoming *ASP Inactive (ASPIA)* message from its peer.

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

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

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

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

M-RK_REG confirm
Direction: ISUA -> LM
Purpose: ASP reports that it has received *Registration Response (REG RSP)* message with registration status as successful from its peer.

M-RK_REG indication
Direction: ISUA -> LM
Purpose: ISUA informs LM that it has successfully processed an incoming *Registration Request (REG REQ)* message.

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

M-RK_DEREG confirm
Direction: ISUA -> LM
Purpose: ASP reports that it has received *Deregistration Request (DEREG REQ)* message with deregistration status as successful from its peer.

M-RK_DEREG indication
Direction: ISUA -> LM
Purpose: ISUA informs LM that it has successfully processed an incoming *Deregistration Request (DEREG REQ)* message from its peer.

1.6.3. 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 ISUP-User Adaptation Protocol (ISUA) 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 ISUA messages **SHALL** contain zero or more ISUA parameters, and **SHALL NOT** contain an encapsulated ISUP message.
- All fields in the ISUA message **MUST** be transmitted in the network byte order, unless otherwise stated.

3.1.1. ISUA Protocol Version

Version: 8-bits (unsigned integer)

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

1 – ISUA 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	Reserved for Other Signalling Adaptation Layers
3	ASP State Maintenance (ASPSM) Messages
4	ASP Traffic Maintenance (ASPTM) Messages
5	Reserved for Other Signalling Adaptation Layers
6	Reserved for Other Signalling Adaptation Layers
7	Reserved for Other Signalling Adaptation Layers
8	Reserved for Other Signalling Adaptation Layers
9	Routing key Management (RKM) Messages
10	ISUA Call Processing (CP) Messages
11	ISUA Circuit Supervision (CS) Messages
12 - 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

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

ISUA Call Processing (CP) Messages

0	Reserved
1	Setup (CSET)
2	More Information (CMOR)
3	Timeout (CTOT)
4	Information (CINF)
5	Proceeding (CPRO)
6	Alerting (CALR)
7	Progress (CPRG)
8	Connect (CCON)
9	Suspend (CSUS)
10	Resume (CRES)
11	Reattempt (CREA)
12	Failure (CERR)
13	In Band Information (CIBI)
14	Release (CREL)
15	Release Complete (CRLC)
16 - 127	Reserved by the IETF
128 - 255	Reserved for IETF-Defined Message Class Extensions

ISUA Circuit Supervision (CS) Messages

0	Reserved
1	Continuity Check (CCNT)
2	Loopback (CLBK)
3	Report (CREP)
4	Reset (CRSC)
5	Reset Acknowledgement (CRSA)
6	Block (CBLO)
7	Block Acknowledgement (CBLA)
8	Unblock (CUBL)
9	Unblock Acknowledgement (CUBA)

10	Query (CQRY)
11	Query Acknowledgement (CQRA)
12 - 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

ISUA 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 [2].

```
 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. However, composite parameters will contain all padding bytes, since all parameters contained within this composite parameter will considered multiples of 4 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. ISUA Message Header

In addition to the Common Message Header, a specific message header is included for ISUA messages. The ISUA message header will immediately follow the Common Message Header in ISUA Call Processing (CP) and Circuit Supervision (CS) messages.

The *ISUA 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                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ISUA Message header can contain the following parameters:

Parameters		
Routing Context	Conditional	*1
Correlation Id	Conditional	*2

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

Note 2: Under some circumstances, the *Correlation Id* parameter **MUST** be included in the ISUA Message Header. See sections "3.9.9 – Correlation Id" and "4.3.4.3 – ASP Active Procedures".

3.3. ISUA Call Processing (CP) Messages

The following section describes the ISUA Call Processing (CP) messages and parameter contents. The general message format includes a Common Message Header, the ISUA Message Header and the CP 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.

These messages are ISUA Call Processing (CP) messages:

ISUA Call Processing (CP) Messages

Message Name		**Message Type**	**Section**
CP Header			3.3.1
Setup	CSET	1	3.3.2
More Information	CMOR	2	3.3.3
Timeout	CTOT	3	3.3.4
Information	CINF	4	3.3.5
Proceeding	CPRO	5	3.3.6
Alerting	CALR	6	3.3.7
Progress	CPRG	7	3.3.8
Connect	CCON	8	3.3.9
Suspend	CSUS	9	3.3.10
Resume	CRES	10	3.3.11
Reattempt	CREA	11	3.3.12
Failure	CERR	12	3.3.13
In Band Information	CIBI	13	3.3.14
Release	CREL	14	3.3.15
Release Complete	CRLC	15	3.3.16

3.3.1. CP Message Header

In addition to the Common Message Header and ISUA Message Header, a specific message header is included for ISUA Call Processing (CP) messages. The CP Message Header will immediately follow the ISUA Message header in these messages.

The *CP 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 = 0x0520          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Circuit Id                           |
+-------------------------------+-------------------------------+
|         Tag = 0x0501          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Call Reference                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CP Message Header* contains the following parameters:

Parameters		
Circuit Id	Conditional	*1
Call Reference	Conditional	*2

Note 1: The *Circuit Id* **MUST** be placed in the ISUA CP Message Header for all CP messages sent from the SGP to the ASP, and is *OPTIONAL* in the ISUA CP Message Header for all CP messages sent from the ASP to the SGP for which a *Circuit Id* was assigend to the call by the SGP before the message was sent. If *Circuit Id* was not assigned by the SGP before the ASP sends a CP message, the ASP *MAY* include the *Circuit Id* parameter for simplicity, but it *MUST* then be coded zero (0). CP messages for which a *Circuit Id* has not been assigned by the SGP include only the *Setup (CSET)* request message sent from the ASP to the SGP.

Note 2: The *Call Reference* **MUST** be placed in the ISUA CP Message Header for all CP messages sent from ASP to the SGP, and is *OPTIONAL* in the ISUA CP Message Header for all CP messages sent from the SGP to the ASP for which a *Call Reference* was assigned to the call by the ASP before the message was sent. If *Call Reference* was not assigned by the ASP before the SGP sends a CP message, the SGP *MAY* include the *Call Reference* parameter for simplicity, but it *MUST* then be coded zero (0). CP messages for which a *Call Reference* has not been assigned by the ASP include only the *Setup (CSET)* indication message sent from the SGP to the ASP.

3.3.2. Setup (CSET)

The *Setup (CSET)* Request message is sent from an ASP to an SG or IPSP to initiate an outgoing ISUP call setup. The *CSET* Indication message is sent from an SGP to an ASP to indicate an incoming ISUP call setup.

The *CSET* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Setup’ (Request, Indication) primitive and the ITU-T and ANSI ISUP ‘IAM’ message [Q.763, T1.113].

The *CSET* 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 = 0x0502          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Call Type                           |
+-------------------------------+-------------------------------+
|         Tag = 0x0503          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Call Flags                          |
+-------------------------------+-------------------------------+
|         Tag = 0x0504          |          Length               |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Called Party Number                     /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |          Length               |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Optional Parameters                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CSET* message can contain the following parameters:

Parameters		
Call Type	Mandatory	
Call Flags	Mandatory	
Called Party Number	Mandatory	
Optional Parameters	Optional	*1

Note 1: Although the *Optional Parameters* are optional in the *CSET* message, the specific ISUP variant and network policy in which the implementation is operating could require that the implementation always place specific parameters in the *Optional Parameters* parameter. An example of this would be the *Charge Number* of GR-394 networks.

3.3.3. More Information (CMOR)

The *More Information (CMOR)* message is sent from an SGP to an ASP to request additional address information for an outgoing ISUP call setup.

The *CMOR* message does not correspond to a Call Control primitive or ISUP message.

The *COMR* message has no message-type-specific parameters beyond the CP Message Header.

3.3.4. Timeout (CTOT)

The *Timeout (CTOT)* message is sent from an SGP to an ASP to indicate that the SG has timed out while waiting for additional address information.

The *CTOT* message does not correspond to a Call Control primitive or ISUP message.

The *CTOT* message has no message-type-specific parameters beyond the CP Message Header.

3.3.5. Information (CINF)

The *Information (CINF)* message is sent from an ASP to an SGP to provide additional address information for an outgoing ISUP call setup. The *CINF* message is sent from an SGP to an ASP to provide additional address information for an incoming ISUP call setup.

The *CINF* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Info’ primitive and the ITU-T and ANSI ISUP ‘SAM’ message [Q.763, T1.113].

The *CINF* 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 = 0x0505          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Subsequent Number                        /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Optional Parameters                       /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CINF* message can contain the following parameters:

Parameters	
Subsequent Number	Mandatory
Optional Parameters	Optional

Note 1:

3.3.6. Proceeding (CPRO)

The *Proceeding (CPRO)* message is sent from an ASP to an SG to indicate that an outgoing call setup is proceeding. The *CPRO* message is sent from an SGP to an ASP to indicate that an incoming call setup is proceeding.

The *CPRO* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘Proceeding’ primitive and the ITU-T and ANSI ISUP ‘ACM’ and ‘CPG’ message [Q.763, T1.113].

The *CPRO* 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 = 0x0508          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Proceeding Flags                        |
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Optional Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CPRO* message can contain the following parameters:

Parameters	
Proceeding Flags	Mandatory
Optional Parameters	Optional
	*1

Note 1:

3.3.7. Alerting (CALR)

The *Alerting (CALR)* message is sent from an ASP to an SG to indicate that the terminating access on a incoming call setup is being alerted. The *CALR* message is sent from an SGP to an ASP to indicate that the terminating access on an outgoing call setup is being alerted.

The *CALR* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘Alerting’ primitive and the ITU-T and ANSI ‘IAM’ message [Q.763, T1.113].

The *CALR* 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 = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Optional Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CALR* message can contain the following parameters:

Parameters	
Optional Parameters	Optional
	*1

Note 1:

3.3.8. Progress (CPRG)

The *Progress (CPRG)* message is sent from an ASP to an SG to indicate that an incoming call setup is in progress. The *CPRG* message is sent from an SGP to an ASP to indicate that an outgoing call setup is in progress.

The *CPRG* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘Progress’ primitive and the ITU-T and ANSI ISUP ‘ACM’ and ‘CPG’ message [Q.763, T1.113].

The *CPRG* 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 = 0x0509          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Progress Event                        |
+-------------------------------+-------------------------------+
|         Tag = 0x050A          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Progress Flags                        |
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |           Length              |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Optional Parameters                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CPRG* message can contain the following parameters:

Parameters	
Progress Event	Mandatory
Progress Flags	Mandatory
Optional Parameters	Optional
	*1

Note 1:

3.3.9. Connect (CCON)

The *Connect (CCON)* message is sent from an ASP to an SG to indicate that an incoming ISUP call has been connected. The *CCON* message is sent from an SGP to an ASP to indicate that an outgoing ISUP call has ben connected.

The *CCON* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Setup’ (Response and Confirmation) primitive and the ITU-T ‘ANM’ and ‘CON’ and ANSI ‘ANM’ message [Q.763, T1.113].

The *CCON* 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 = 0x050E          |           Length              |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Optional Parameters                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CCON* message can contain the following parameters:

Parameters	
Optional Parameters	Optional
	*1

Note 1:

3.3.10. Suspend (CSUS)

The *Suspend (CSUS)* message is sent from the ASP to an SG or from the SGP to the ASP to indicate that an established call has been suspended.

The *CSUS* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Suspend’ primitive and the ITU-T and ANSI ‘SUS’ message [Q.763, T1.113].

The *CSUS* 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 = 0x050B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                     Suspend/Resume Flags                      |
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Optional Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CSUS* message can contain the following parameters:

Parameters	
Suspend/Resume Flags	Mandatory
Optional Parameters	Optional

Note 1:

3.3.11. Resume (CRES)

The *Resume (CRES)* message is sent from the ASP to an SG or from the SGP to the ASP to indicate that a previously suspended established call has been resumed.

The *CRES* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Resume’ primitive and the ITU-T and ANSI ‘RES’ message [Q.763, T1.113].

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 = 0x050B          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                     Suspend/Resume Flags                      |
+-------------------------------+-------------------------------+
|         Tag = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Optional Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CRES* message can contain the following parameters:

Parameters	
Suspend/Resume Flags	Mandatory
Optional Parameters	Optional

Note 1:

3.3.12. Reattempt (CREA)

The *Reattempt (CREA)* Indication message is sent from an SGP to an ASP to indicate that a call attempt on a circuit should be reattempted on an alternate circuit.

If the ASP selected the outgoing circuit in the corresponding *CSET*, then the ASP is responsible for selecting another circuit and issuing a new *CSET* message. If the ASP did not select the outgoing circuit in the corresponding *CSET* message, then the SGP is responsible for performing an automatic reattempt on a new circuit or subsequently indicating call failure with a **CERR** message.

The *CREA* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Reattempt’ primitive and does not correspond to an ISUP message.

The *CREA* 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 = 0x0506          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Reattempt Reason                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CREA* message can contain the following parameters:

Parameters	
Reattempt Reason	Mandatory

3.3.13. Failure (CERR)

The *Failure (CERR)* message is sent from an ASP to an SG to indicate the failure of an incoming call setup. The *CERR* message is sent from an SGP to an ASP to indicate the failure of an outgoing call setup.

The *CERR* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Failure ‘Call Failure’ primitive and the ITU-T and ANSI ‘RST,’ ‘REL’ and ‘RLC’ message [Q.763, T1.113].

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 = 0x050C          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Failure Reason                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CERR* message can contain the following parameters:

Parameters		
Failure Reason	Mandatory	
		*1

Note 1:

3.3.14. In Band Information (CIBI)

The *In Band Information (CIBI)* message is sent from an ASP to an SG to indicate that in band information is now available for an incoming call. The *CIBI* message is sent from an SGP to an ASP to indicate that in band information is now available for an outgoing call.

The *CIBI* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Setup ‘In Band Information’ primitive and the ITU-T and ANSI ‘ACM’ and ‘CPG’ message [Q.763, T1.113].

The *CIBI* 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 = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Optional Parameters                     /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CIBI* message can contain the following parameters:

Parameters		
Optional Parameters	Optional	
		*1

3.3.15. Release (CREL)

The *Release (CREL)* message is sent from an ASP to an SG or from the SGP to an ASP to release a call during the setup or established phase.

The *CREL* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control 'Release' (Request, Indication) primitive and the ITU-T and ANSI 'REL' message [Q.763, T1.113].

The *CREL* 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 = 0x050D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                             Cause                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CREL* message can contain the following parameters:

Parameters	
Cause	Mandatory

3.3.16. Release Complete (CRLC)

The *Release Complete (CRLC)* message is sent from an ASP to an SG or from an SGP to an ASP to confrim the release of a call.

The *CRLC* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control 'Release' (Response, Confirmation) primitive and the ITU-T and ANSI 'REL' and 'RLC' message [Q.763, T1.113].

The *CRLC* 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 = 0x050D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                             Cause                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CRLC* message can contain the following parameters:

Parameters	
Cause	Mandatory

3.4. ISUA Circuit Supervision (CS) Messaegs

ISUA Circuit Supervision (CS) Messages are used to convey circuit management information to Call Control. Theses messages correspond to specific RESET, BLOCKING, UNBLOCKING and CCT GROUP QUERY primitives. The general message format includes a Common Message Header, the ISUA Message Header, and the CS 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.

These messages are ISUA Circuit Supervision (CS) Messages:

ISUA Circuit Supervision (CS) Messages

Message Name		**Message Type**	**Section**
CS Header			3.4.1
Continuity Check	CCNT	6	3.4.2
Loopback	CLBK	7	3.4.3
Report	CREP	8	3.4.4
Reset	CRSC	1	3.4.5
Reset Acknowledgement	CRSA	2	3.4.6
Block	CBLO	3	3.4.7
Block Acknowledgement	CBLA	4	3.4.8
Unblock	CUBL	5	3.4.9
Unblock Acknowledgement	CUBA	6	3.4.10
Query	CQRY	7	3.4.11

Query Acknowledgement	CQRA	8	3.4.12

3.4.1. CS Message Header

In addition the the Common Message Header and ISUA Message Header, a specific message header is included for ISUA Circuit Supervision (CS) messages. The CS Message Header will immediately follow the ISUA Message Header in these messages.

The *CS 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 = 0x0520          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Circuit Id                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CS Message Header* contains the following parameters:

Parameters	
Circuit Id	Mandatory

3.4.2. Continuity Check (CCNT)

The *Continuity Check (CCNT)* message is sent from an ASP to an SGP to request an continuity check on a specified circuit. The *CCNT* message is sent from an SGP to an ASP to indicate an a continuity check request on the specified circuit.

The *CCNT* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Continuity Recheck’ (Request) primitive and the ITU-T and ANSI ISUP ‘CCR’ message [Q.763, T1.113].

The *CCNT* message has no message-type-specific parameters beyond the CS Message Header.

3.4.3. Loopback (CLBK)

The *Loopback (CLBK)* message is sent from an ASP to an SGP to indicate that a loopback has been established on the local end of the specified circuit. The *CLBK* message is sent from an ASP to an SGP to indicate that a loopback has been establish on the remote end of the specified circuit.

The *CLBK* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] Call Control ‘Continuity Recheck’ (Confirmation) primitive and the ITU-T and ANSI ISUP ‘LPA’ message [Q.763, T1.113].

The *CLBK* message has no message-type-specific parameters beyond the CS Message Header.

3.4.4. Report (CREP)

The *Report (CREP)* Request message is sent from an ASP to SG or from an SGP to an ASP to indicate the success or failure of a continuity test operation on the specified circuit.

The *CREP* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘Continuity Report’ primitive and the ITU-T and ANSI ISUP ‘COT’ message [Q.763, T1.113].

The *CREP* 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 = 0x0507          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Check Result                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CREP* message can contain the following parameters:

Parameters	
Check Result	Mandatory

3.4.5. Reset (CRSC)

The *Reset (CRSC)* message is sent from an ASP to an SG to request the reset of the specified circuit(s). The *CRSC* message is sent from the SGP to an ASP to indicate the reset reset of the specified circuit(s).

The *CRSC* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] 'Reset' (Request) primitive and the ITU-T and ANSI ISUP 'RSC' and 'GRS' message [Q.763, T1.113].

The *CRSC* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Circuit Range                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CRSC* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CRSC* message corresponds to the 'GRS' message. When the *Circuit Range* is not present in the message, the *CRSC* message corresponds to the 'RSC' message.

3.4.6. Reset Acknowledgement (CRSA)

The *Reset Acknowledgement (CRSA)* message is sent from an SGP to an ASP to confirm the reset of the specified circuit(s).

The *CRSA* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] 'Reset' (Confirmation) primitive and the ITU-T and ANSI ISUP 'RLC' and 'GRA' message.

The *CRSA* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Circuit Range                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CRSA* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CRSA* message corresponds to the 'GRA' message and the *Circuit Range* parameter **SHOULD** match the corresponding parameter in the *CRSC* request message. When the *Circuit Range* is not present in the message, the *CRSA* message corresponds to the 'RLC' message.

3.4.7. Block (CBLO)

The *CBLO* Request message is sent from an ASP to an SG or IPSP to perform a blocking request. The *CBLO* Indication message is sent from the SGP to an ASP to indicate the blocking indication.

The *CBLO* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] 'BLOCKING' primitive and the ITU-T and ANSI 'BLO' and 'CGB' message.

The *CBLO* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Circuit Range                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CBLO* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CBLO* message corresponds to the ‘CGB’ message. When the *Circuit Range* is not present in the message, the *CBLO* message corresponds to the ‘BLO’ message.

3.4.8. Block Acknowledgement (CBLA)

The *Block Acknowledgement (CBLA)* Request message is sent from an ASP to an SG or IPSP to perform a blocking response. The *CBLA* Indication message is sent from the SGP to an ASP to indicate the blocking confirmation.

The *CBLA* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘BLOCKING’ primitive and the ITU-T and ANSI ‘BLA’ and ‘CGBA’ message.

The *CBLA* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Circuit Range                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CBLA* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CBLA* message corresponds to the ‘CGBA’ message and the *Circuit Range* parameter **SHOULD** match the corresponding parameter in the *CBLO* request message. When the *Circuit Range* is not present in the message, the *CBLA* message corresponds to the ‘BLA’ message.

3.4.9. Unblock (CUBL)

The *Unblock (CUBL)* Request message is sent from an ASP to an SG or IPSP to perform a unblocking request. The *CUBL* Indication message is sent from the SGP to an ASP to indicate the unblocking indication.

The *CUBL* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘UNBLOCKING’ primitive and the ITU-T and ANSI ‘UBL’ and ‘CGU’ message.

The *CUBL* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                          Circuit Range                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CUBL* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CUBL* message corresponds to the 'CGU' message. When the *Circuit Range* is not present in the message, the *CUBL* message corresponds to the 'UBL' message.

3.4.10. Unblock Acknowledgement (CUBA)

The *Unblock Acknowledgement (CUBA)* Request message is sent from an ASP to an SG or IPSP to perform a unblocking response. The *CUBA* Indication message is sent from the SGP to an ASP to indicate the unblocking confirmation.

The *CUBA* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] 'UNBLOCKING' primitive and the ITU-T and ANSI 'UBA' and 'CGUA' message.

The *CUBA* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Circuit Range                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CUBA* message can contain the following parameters:

Parameters		
Circuit Range	Conditional	*1

Note 1: When the *Circuit Range* parameter is included in the message, the *CUBA* message corresponds to the 'CGUA' message and the *Circuit Range* parameter **SHOULD** match the corresponding parameter in the *CUBL* request message. When the *Circuit Range* is not present in the message, the *CUBA* message corresponds to the 'UBA' message.

3.4.11. Query (CQRY)

The *Query (CQRY)* Request message is sent from an ASP to an SG or IPSP to perform a query request. The *CQRY* Indication message is sent from the SGP to an ASP to indicate the query indication.

The *CQRY* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] 'CCT GROUP QUERY' primitive and the ITU-T and ANSI 'CQM' message.

The *CQRY* 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Circuit Range                         /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CQRY* message can contain the following parameters:

Parameters		
Circuit Range	Mandatory	1

3.4.12. Query Acknowledgement (CQRA)

The *Query Acknowledgement (CQRA)* Request message is sent from an ASP to an SG or IPSP to perform a query response. The *CQRA* Indication message is sent from the SGP to an ASP to indicate the query confirmation.

The *CQRA* message corresponds to the ITU-T [Q.764] and ANSI [T1.113] ‘CCT GROUP QUERY’ primitive and the ITU-T and ANSI ‘CQMA’ message.

The *CQRA* 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 = 0x05XX          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Circuit Status                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *CQRA* message can contain the following parameters:

Parameters		
Circuit Status	Mandatory	*1

Note 1: The *Circuit Status* parameter **SHOULD** contain a circuit status for each of the circuit identifiers present in the corresponding *CQRY* message.

3.5. Application Server Process State Maintenance (ASPSM) Messages

3.5.1. ASP Up (UP)

The *ASP Up (UP)* message is used to indicate to a remote ISUA 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).

3.5.2. ASP Up Ack (UP ACK)

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

3.5.3. ASP Down (DOWN)

The *ASP Down (DOWN)* message is used to indicate to a remote ISUA 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

3.5.4. ASP Down Ack (DOWN ACK)

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

3.5.5. Heartbeat (BEAT)

The *Heartbeat (BEAT)* message is optionally used to ensure that the ISUA 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

3.5.6. 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

3.6. Application Server Process Traffic Maintenance (ASPTM) Messages

3.6.1. ASP Active (ASPAC)

The *ASP Active (ASPAC)* message is sent by an ASP to indicate to a remote ISUA 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.

3.6.2. ASP Active Ack (ASPAC ACK)

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

3.6.3. ASP Inactive (ASPIA)

The *ASP Inactive (ASPIA)* message is sent by an ASP to indicate to a remote ISUA 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.

3.6.4. ASP Inactive Ack (ASPIA ACK)

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

3.7. Management (MGMT) Messages

3.7.1. Error (ERR)

The *Error (ERR)* message is used by a ISUA 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 = 0x0521          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                       Network Appearance                      |
+-------------------------------+-------------------------------+
|         Tag = 0x0006          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Routing Context                       /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0520          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                           Circuit Id                          |
+-------------------------------+-------------------------------+
|         Tag = 0x0501          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Call Reference                        |
+-------------------------------+-------------------------------+
|         Tag = 0x0007          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Diagnostic Info                       /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *ERR* message can contain the following parameters:

Parameters		
Error Code	Mandatory	
Routing Context	Conditional	*1
Call Reference	Conditional	*2
Circuit Id	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 "Call Reference Unknown," the Call Reference parameter **MUST** contain the call reference for which status is unknown or unauthorized.

Note 3: When the Error Code is "Circuit Status Unknown," the *Circuit Id* parameter **MUST** contain the circuit 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 at least the first 40 bytes of the message that caused the ERR message to be sent.

3.7.2. Notify (NTFY)

The Notify message is used to provide an autonomous indication of ISUA 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".

3.8. 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.

3.8.1. Registration Request (REG REQ)

The *Registration Request (REG REQ)* message is sent by an ASP to indicate to a remote ISUA 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 = 0x0522          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                         Routing Key 1                         /
\                                                               \
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0522          |            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.

3.8.2. 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 = 0x0014          |             Length            |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                     Registration Result 1                     /
\                                                               \
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0014          |             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.

3.8.3. Deregistration Request (DEREG REQ)

The *Deregistration Request (DEREG REQ)* message is sent by an ASP to indicate to a remote ISUA 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.

3.8.4. Deregistration Response (DEREG RSP)

The *Deregistration Response (DEREG RSP)* message is used as a response to the *DEREG REQ* message from a remote ISUA 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 = 0x0015          |          Length = 12          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                    Deregistration Result 1                    /
\                                                               \
+-------------------------------+-------------------------------+
\                                                               \
/                              ...                              /
\                                                               \
+-------------------------------+-------------------------------+
|         Tag = 0x0015          |          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.

3.9. Common Parameters

These TLV parameters are common across the different adaptation layers:

Parameter Name	**Parameter ID**	**Section**
Reserved	0x0000	–
Not used in ISUA	0x0001	–
Not used in ISUA	0x0002	–
Not used in ISUA	0x0003	–
Info String	0x0004	3.9.1
Not used in ISUA	0x0005	–
Routing Context	0x0006	3.9.2
Diagnostic Info	0x0007	3.9.3
Not used in ISUA	0x0008	–
Heartbeat Data	0x0009	3.9.4
Not used in ISUA	0x000A	–
Traffic Mode Type	0x000B	3.9.5
Error Code	0x000C	3.9.6
Status	0x000D	3.9.7
Not used in ISUA	0x000E	–
Not used in ISUA	0x000F	–
Not used in ISUA	0x0010	–
ASP Identifier	0x0011	3.9.8
Not used in ISUA	0x0012	–
Correlation Id	0x0013	3.9.9
Registration Result	0x0014	3.9.10
Deregistration Result	0x0015	3.9.11
Registration Status	0x0016	3.9.12
Deregistration Status	0x0017	3.9.13
Local Routing Key Identifier	0x0018	3.9.14

3.9.1. 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 UTF-8 [RFC 2279] 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.

3.9.2. Routing Context

The *Routing Context* parameter is included in all ISUA CP and CS 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 one-to-one relationship between a Routing Context value, an SG Routing Key and an Application Server [3]. If the Routing Context parameter is present, it **SHOULD** be the first parameter in the message as it defines the format and/or interpretation of the parameters containing a PC or SSN value.

3.9.3. Diagnostic Information

The *Diagnostic Info* parameter is used in the MGMT *)Error (ERR)* message to provide additional information concerning the message that generated an *ERR* message reply. The *Diagnostic Info* parameter **SHOULD** contain at least 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.

3.9.4. 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 (BEAT)* 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 (BEAT)* message useful to the sending node (e.g. the data field can contain a time stamp, a sequence number, etc.).

3.9.5. 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	Load-share
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 ASPs.

3.9.6. 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 *ERR* 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
21	Invalid Network Appearance
22	Missing Parameter
23	Routing Key Change Refused
25	Invalid Routing Context
26	No Configured AS for ASP
34	*Circuit Status Unknown*
35	*Call Reference Status Unknown*

The "Invalid Version" error is sent if a message was received with an invalid or unsupported version. The *ERR* message contains the supported version in the Common header. The *ERR* 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 (ASPAC)* 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 *ERR* message). For example, silent discard is used by an ASP if it received a ISUA CP 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 *ERR* 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 ISUA CP message was received on stream "0").

The "Refused - Management Blocking" error is sent when an *ASP Up (ASPUP)* or *ASP Active (ASPAC)* 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 (ASPAC)* message, the Routing Context(s) in the *ASP Active (ASPAC)* message **SHOULD** be included in the *ERR* message.

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

The "Invalid ASP Identifier" is send by a SGP in response to an *ASP Up (ASPUP)* 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 "Invalid Network Appearance" error is sent by a SGP if an ASP sends a message with an invalid (not configured) Network Appearance value. For this error, the invalid (not configured) Network Appearance **MUST** be included in the Network Appearance parameter in the *ERR* message.

The "Missing Parameter" error is sent if a mandatory parameter was not included in a message.

The "Routing Key Change Refused" error is sent when an SG refuses a change in the *Routing Key* parameters.

The "Invalid Routing Context" error is sent if a message is received from a peer with an invalid (not configured) 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 *ERR* 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 "Circuit Status Unknown" Error **MAY** be sent it a *CQRY* is receive at an SG inquiring of the status of a circuit or circuits 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 *Circuit Id* **MUST** be included along with any *Network Appearance* or *Routing Context* associated with the *Circuit Id* from the *CQRY* message.

The "Call Reference Status Unknown" Error **MAY** be sent it a *CQRY* is receive at an SG inquiring of the status of a circuit or circuits 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 *Call ReferenceFR* **MUST** be included along with any *Network Appearance* or *Routing Context* associated with the *Call Reference* from the *CQRY* message.

3.9.7. 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.

3.9.8. 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 (NTFY)* message (see Section 3.7.2).

3.9.9. 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 ISUA Message Header to indicate to a newly entering ASP in a Broadcast AS where in the traffic flow of ISUA messages the ASP is joining. It is attached to the ISUA Message Header of the first CP 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 [4].

3.9.10. Registration Result

The *Registration Result* parameter is used to indicate the result of a successful or unsuccessful registration operation for a specific *Routing Key*.

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 = 0x0014          |            Length             |
+-------------------------------+-------------------------------+
|                 Local Routing Key Identifier                  |
+---------------------------------------------------------------+
|                      Registration Status                      |
+---------------------------------------------------------------+
|                        Routing Context                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Local Routing Key Identifier: TLV

The *Local Routing Key Identifier* field is mandatory in the *Registration Result* parameter. The *Local Routing Key Identifier* field contains the same TLV formatted parameter value as found in the corresponding *Routing Key* parameter in the *Registration Request (REG REQ)* message.

Registration Status: TLV

The *Registration Status* field is mandatory in the *Registration Result* parameter. The *Registration Status* field indicates the success or reason for failure of the corresponding registration request. For details on the format of the *Registration Status* parameter, see Section 3.9.12.

Routing Context: TLV

The *Routing Context* field is mandatory in the *Registration Result* parameter. The *Routing Context* field contains the TLV formatted *Routing Context* parameter for the associated *Routing Key* if the registration was successful. If the registration was not successful, it is set to zero (0).

3.9.11. Deregistration Result

The *Deregistration Result* parameter is used to indicate the result of a successful or unsuccessful deregistration operation for a specific *Routing Key*.

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 = 0x0015          |            Length             |
+-------------------------------+-------------------------------+
|                        Routing Context                        |
+---------------------------------------------------------------+
|                     Deregistration Status                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Routing Context: TLV

The *Routing Context* field is mandatory in the *Deregistration Result* parameter. The *Routing Context* field contains the same TLV formatted *Routing Context* parameter as found in the corresponding *Deregistration Request (DEREG REQ)* message.

Deregistration Status: TLV

The *Deregistration Status* field is mandatory in the *Deregistration Result* parameter. The *Deregistration Status* field indicates the success or reason for failure of the corresponding deregistration request. For details on the format of the *Deregistration Status* parameter, see Section 3.9.13.

3.9.12. Registration Status

The *Registration Status* parameter is used to indicate the success or failure of a registration operation.

The *Registration 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 = 0x0016          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                      Registration Status                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Registration Status: 32-bits (unsigned integer)

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

Its values can be:

0	Successfully Registered
1	Error - Unknown
2	Error - Invalid Circuit Identifier
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
11	Error - Routing Context Registration Refused

3.9.13. Deregistration Status

The *Deregistration Status* parameter is used to indicate the success or failure of a deregistration operation.

The *Deregistration 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 = 0x0017          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                     Deregistration Status                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Deregistration Status: 32-bits (unsigned integer)

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

Its values can 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

3.9.14. Local Routing Key Identifier

The *Local Routing Key Identifier* parameter is used for correlating the *Routing Key* parameter in a specific *Registration Request (REG REQ)* message with the *Registration Result* parameter in the corresponding *Registration Response (REG RSP)* message.

The *Local Routing Key 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 = 0x0018          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                  Local Routing Key Identifier                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

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

The *Local Routing Key Identifier* value is assigned by the ASP and is used to correlate the response in a *Registration Response (REG RSP)* message with the original registration request from the *Registration Request (REG REQ)* message. The *Local Routing Key Identifier* value must remain unique until the *REG RSP* message is received.

3.10. ISUA-Specific parameters

These TLV parameters are specific to the ISUA protocol:

Parameters used in CP Messages

Parameter Name	Parameter ID	Section
Call Reference	0x0501	3.10.1.1
Call Type	0x0502	3.10.1.2
Call Flags	0x0503	3.10.1.3
Called Party Number	0x0504	3.10.1.4
Subsequent Number	0x0505	3.10.1.5
Reattempt Reason	0x0506	3.10.1.6
Check Result	0x0507	3.10.1.7
Proceeding Flags	0x0508	3.10.1.8
Progress Event	0x0509	3.10.1.9
Progress Flags	0x050A	3.10.1.10
Suspend/Resume Flags	0x050B	3.10.1.11
Failure Reason	0x050C	3.10.1.12
Cause	0x050D	3.10.1.13
Optional Parameters	0x050E	3.10.1.14

Parameters used in CS Messages

Parameter Name	Parameter ID	Section
Circuit Status	0x0510	3.10.2.1

Other Parameters

Parameter Name	Parameter ID	Section
Circuit Id	0x0520	3.10.3.1
Network Appearance	0x0521	3.10.3.2
Routing Key	0x0522	3.10.3.3
Circuit Range	0x0523	3.10.3.4
Local Point Code	0x0524	3.10.3.5
Remote Point Code	0x0525	3.10.3.5

3.10.1. Parameters used in CP Messages

3.10.1.1. Call Reference

The *Call Reference* parameter is used in the ISUA Message Header to identify the call within the Application Server indicated by the *Routing Context* (also in the ISUA Message Header).

The *Call Reference* 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 = 0x0501          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Call Reference                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Call Reference* parameter contains the following fields:

Call Reference field: 32-bits (unsigned integer)

The *Call Reference* field contains an identifier that is used both at the SG and the ASP to identify a call within an Application Server. The Call Reference 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 Call Reference, but not both.

3.10.1.2. Call Type

The *Call 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 = 0x0502          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Call Type                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Call Type* parameter contains the following fields:

Call Type field: 32-bits (unsigned integer)

The *Call Type* field can take on the following values:

0	Speech
1	64 kbit/s unrestricted digital information
2	3.1 kHZ audio
3	64 kbit/s preferred
4	2 x 64 kbit/s unrestricted digital information
5	284 kbit/s unrestricted digital information
6	1536 kbit/s unrestricted digital information
7	1920 kbit/s unrestricted digital information

3.10.1.3. Call Flags

The *Call 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 = 0x0503          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Call Flags                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Call Flags* parameter contains the following fields:

Call Flags field: 32-bits (bit field)

The *Call Flags* field consists of the following fields:

Satellite Indicator: 2-bits (bits 30-31)

The Satellite Indicator field corresponds to the Nature of Address Indicators of ITU-T ISUP [Q.763] and indicate the number of satellites present in the ISUP connection. The Satellite Indicator field can take on the following values:

0 no satellite in circuit
1 one satellite in circuit
2 two satellites in circuit
3 *(reserved)*

Continuity Check Indicator: 2-bits (bits 28-29)

The Continuity Check Indicator field corresponds to the Nature of Address Indicators of ITU-T ISUP [Q.763] and indicates whether a continuity check is required on the circuit, whether a check has previously been performed, or which a check is not required on the circuit. The Continuity Check Indicator field can take on the following values:

0 no continuity check required
1 continuity check performed on previous circuit
2 continuity check required
3 *(reserved)*

Outgoing Half Echo Control Device: 1-bit (bit 27)

The Outgoing Half Echo Control Device field corresponds to the Nature of Address Indicator of ITU-T ISUP [Q.763] and indicates whether an outgoing half echo control device is included on the circuit. The Outgoing Half Echo Control Device field can take on the following values:

0 no outgoing half echo control device included
1 outgoing half echo control device included

International/National: 1-bit (bit 26)

The Internation/National field corresponsds to the Forward Call Indicators of ITU-T ISUP [Q.763] and indicates whether the call is an International or National call. The International/National field can take on the following values:

0 National call
1 International call

End to End Method: 2-bits (bit 24-25)

The End to End Method field corresponds to the Forward Call Indicators of ITU-T ISUP [Q.763] and indicates which end to end methods are available. The End to End Method field can take on the following values:

0 link by link method only
1 pass along method available
2 SCCP end to end method available
3 both methods available

Interworking Encountered: 1-bit (bit 23)

The Interworking Encountered field corresponds to the Forward Call Indicators of ITU-T ISUP [Q.763] and indicates whether interworking was encountered on the call. The Interworking Encountered field can take on the following values:

0 no interworking encountered
1 interworking encountered

End to End Information Available: 1-bit (bit 22)

The End to End Information Available field corresponds to the Forward Call Indicators of ITU-T ISUP [Q.763] and indicates whether end to end information is now available. The End to End Information Available field can take on the following values:

0 no end to end information available
1 end to end information available

ISUP All the Way: 1-bit (bit 21)

The ISUP All the Way field corresponds to the Forward Call Indicators of the ITU-T ISUP [Q.763] and indicates whether ISDN User Part is used all the way. The ISUP All the Way field can take on the following values:

0 ISDN User Part not used all the way
1 ISDN User Part used all the way

Originating Access ISDN: 1-bit (bit 20)

The Originating Access ISDN field corresponds to the Forward Call Indicators of the ITU-T ISUP [Q.763] and indicates whether the originating access is ISDN. The Originating Access ISDN field can take on the following values:

0 originating access is not ISDN
1 originating access is ISDN

SCCP Methods Available: 2-bits (bit 18-19)

The SCCP Methods Available field corresponds to the Forward Call Indicators of the ITU-T ISUP [Q.763] and indicates the SCCP method available. The SCCP Methods Available field can take on the following values:

0 no SCCP method available
1 connectionless SCCP method available
2 connection oriented SCCP method available
3 both methods available

3.10.1.4. Called Party Number

The *Called Party 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 = 0x0504          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Called Party Number                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Called Party Number* parameter contains the following fields:

Called Party Number field: 32-bits (unsigned integer)

3.10.1.5. Subsequent Number

The *Subsequent 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 = 0x0505          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                       Subsequent Number                       /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Subsequent Number* parameter contains the following fields:

Subsequent Number field: 32-bits (unsigned integer)

3.10.1.6. Reattempt Reason

The *Reattempt 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 = 0x0506          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Reattempt Reason                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Reattempt Reason* parameter contains the following fields:

Reattempt Reason field: 32-bits (unsigned integer)

The Reattempt Reason field indicates the reason that a call reattempt is indicated. The Reattempt Reason field can take on one of the following values:

1	dual sizeure
2	reset
3	blocking
4	T24 timeout
5	unexpected message
6	continuity check failure
	all other values reserved

The Reattempt Reason values are interpreted as follows:

The *"dual sizeure"* reason indicates that the selected circuit was siezed by a controlling exchange during the initial setup of the call (i.e. before any backward message was received).

The *"reset"* reason indicates that the selected circuit was reset during the initial setup of the call (i.e. before any backward message was received).

The *"blocking"* reason indicates that the selected circuit was blocked during the initial setup of the call (i.e. before any backward message was received).

The *"T24 timeout"* reason indicates that continuity check failure occured due to timeout on the selected circuit.

The *"unexpected message"* reason indicates that an unexpected messagew as received for the call during the initial setup of the call (i.e. before any backward message was received).

The *"continuity check failure"* reason indicates that continuity check failed on the selected circuit.

3.10.1.7. Check Result

The *Check 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 = 0x0507          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Check Result                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Check Result* parameter contains the following fields:

Check Result field: 32-bits (unsigned integer)

The Check Result field indicates the success of failure of the continuity check. The Check Result field can take on one of the following values:

0 continuity check failed
1 continuity check successful

3.10.1.8. Proceeding Flags

The *Proceeding 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 = 0x0509          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Proceeding Flags                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Proceeding Flags* parameter contains the following fields:

Proceeding Flags field: 32-bits (bit field)

The Proceeding Flags field contains the following bit fields:

Charge: 2-bits (bit 30-31)

The Charge field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether the call is to be charged. The Charge field can take on one of the following values:

1 charge
2 no charge
all other values reserved

Free: 2-bits (bit 28-29)

The Free field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether the call is subscriber free or connection free. The Free field can take on one of the following values:

0 no indication
1 subscriber free
2 connection free
all other values reserved

Payphone: 2-bits (bit 26-27)

The Payphone field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether the call has terminated to an ordinary subscriber or a payphone. The Payphone field can take on one of the following values:

1 ordinary subscriber
2 payphone
all other values reserved

End to End Method Available: 2-bits (bit 24-25)

The End to End Method Available field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates which end to end methods are available. The End to End Method Available field can take on one of the following values:

0 link by link method available
1 pass along method available
2 SCCP method available
3 all methods available

Interworking Encountered: 1-bit (bit 23)

The Interworking Encountered field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether interworking was encountered on the call. The Interworking Encountered field can take on one of the following values:

0	no interworking encountered
1	interworking encountered

End to End Information Available: 1-bit (bit 22)

The End to End Information Available field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether end to end information is available. The End to End Information Available field can take on one of the following values:

0	no end to end information available
1	end to end information available

ISUP All the Way: 1-bit (bit 21)

The ISUP All the Way field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether ISDN User Part was used all the way. The ISUP All the Way field can take on one of the following values:

0	ISDN user part not used all the way
1	ISDN user part used all the way

Holding Requested: 1-bit (bit 20)

The Holding Requested field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether holding was requested. The Holding Requested field can take on one of the following values:

0	holding not requested
1	holding requested

Terminating Access ISDN: 1-bit (bit 19)

The Terminating Access ISDN field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether the terminating access is ISDN. The Terminating Access ISDN field can take on one of the following values:

0	terminating access not ISDN
1	terminating access ISDN

Incoming Half Echo Control Device: 1-bit (bit 18)

The Incoming Half Echo Control Device field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates whether an incoming half echo control device has been included on the call. The Incoming Half Echo Control Device field can take on one of the following values:

0	no incoming half echo control device
1	incoming half echo control device

SCCP Methods Available: 2-bits (bit 16-17)

The SCCP Methods Available field corresponds to the Backwards Call Indicators of ITU-T ISUP [Q.763] and indicates the SCCP methods available. The SCCP Methods Available field can take on one of the following values:

0	no SCCP method available

1 connectionless SCCP method available
2 connection oriented SCCP method available
3 both SCCP methods available

3.10.1.9. Progress Event

The *Progress Event* 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 = 0x0509          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Progress Event                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Progress Event* parameter contains the following fields:

Progress Event field: 32-bits (unsigned integer)

The Progress Event field indicates the progress event associated with the call. The Progress Event field can take on one of the following values:

1 alerting
2 progress
3 in band information
4 call forwarded on busy
5 call forwarded on no answer
6 call forwarded unconditional
all other values reserved

The Progress Event values are interpreted as follows:

The *"alerting"* event indicates that the called party is being alerted. This event is indicated only if a *CPRO* message has already been received.

The *"progress"* event indicates that the call is progressing with the specified optional parameters.

The *"in band information"* event is indicated only via the *CIBI* message and **MUST NOT** be indicated in the *CPRG* message.

The *"call forwarded on busy"* event indicates that the call has been forwarded on busy and the optional parameters (if any) in the message contain the attributes of the forwarding (e.g. redirecting number).

The *"call forwarded on no answer"* event indicates that the call has been forwarded on no answer and the optional parameters (if any) in the message contain the attributes of the forwarding (e.g. redirecting number).

The *"call forwarded unconditional"* event indicates that the call has been forwarded unconditionally and the optional parameters (if any) in the message contain the attributes of the forwarding (e.g. redirecting number).

3.10.1.10. Progress Flags

The *Progress 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 = 0x050A          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                        Progress Flags                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Progress Flags* parameter contains the following fields:

Progress Flags field: 32-bits (bit field)

The Progress Flags field contains the following bit fields:

Presentation Restricted: 1-bit (bit 31)

The Presentation Restricted field indicates whether the event (and any associated optional parameters, such as redirecting number) is presentation restricted. The Presentation Restricted field can take on the following values:

0 event presentation allowed
1 event presentation restricted

3.10.1.11. Suspend/Resume Flags

The *Suspend/Resume 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 = 0x050B          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                      Suspend/Resume Flags                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Suspend/Resume Flags* parameter contains the following fields:

Suspend/Resume Flags field: 32-bits (bit field)

The Suspend/Resume flags field contains the following bit fields:

Network Initiated: 1-bit (bit 31)

The Network Initiated field indicates whether the suspend or resume operation was user or network initiated. The Network Initiated field can take on the following values:

0 user initiated
1 network initiated

3.10.1.12. Failure Reason

The *Failure 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 = 0x050C          |           Length = 8          |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                         Failure Reason                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Failure Reason* parameter contains the following fields:

Failure Reason field: 32-bits (unsigned integer)

The Failure Reason indicates the reason for call setup failure and can take on the following values:

1 continuity check failure
2 received release complete
3 blocking
4 T6 timeout
5 T7 timeout
6 T8 timeout
7 T9 timeout

8	T35 timeout
9	T38 timeout
	all other values reserved

The values of the Failure Reason field are interpreted as follows:

The "continuity check failure" reason indicates that continuity check on the circuit failed. The applies to incoming calls only.

The "received release complete" reason indicates that the selected circuit was not completely released by the distant end. The applies to incoming calls only.

The "blocking" reason indicates that the circuit was blocked during call setup. The applies to incoming calls only.

The "T6 timeout" reason indicates that the call was suspended beyond the allowable period. The applies to all established calls.

The "T7 timeout" reason indicates that there was no response to the call setup request. The applies to outgoing calls only.

The "T8 timeout" reason indicates that the call failed waiting for a continuity check report from the distant end. The applies to incoming calls only.

The "T9 timeout" reason indicates that the call failed while waiting for the distant end to answer. The applies to outgoing calls only.

The "T35 timeout" reason indicates that additional information (digits) were not received from the caller within a sufficient period. The applies to incoming calls only.

The "T38 timeout" reason indicates that the call was suspended beyond the allowable period. The applies to all established calls.

3.10.1.13. Cause

The *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 = 0x050D          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                             Cause                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Cause* parameter contains the following fields:

Cause field: 32-bits (unsigned integer)

The Cause field indicates the reason for call release and can take on the following values:

	ITU-T	ANSI
1	unalloc. no.	
2	no route to transit ntwk	
3	no route to dest	
4	send special info tone	
5	misdialled trunk prefix	
8	preemption	
9	preemption cc’t reserved	
16	normal call clearing	
17	user busy	
18	no user responding	
19	no answer	
20	subscriber absent	
21	call rejected	
22	no. changed	

	ITU-T	ANSI
23	redirect	unalloc. dest no.
24	———————————	unknown business group
25	———————————	exchange routing error
26	———————————	misrouted call to ported no.
27	out of order	LNP QoR no. not found
28	address incomplete	
29	facility rejected	
31	normal unspecified	
34	no cc't available	
38	ntwk out of order	
41	temporary failure	
42	switching equip cong	
43	access info discarded	
44	cc't unavailable	
45	———————————	resource preemption
46	precedence call blocked	
47	resource unavailable	
50	not subscribed	
51	———————————	call type incompatible
53	og call barred in CUG	
54	———————————	group restrictions
55	ic call barred in CUG	
57	bearer cap not authorized	
58	bearer cap not available	
62	inconsistency	
63	service opt not available	
65	bearer cap not impl.	
69	facility not impl.	
70	restricted bearer cap only	
79	service opt not impl.	
87	user not member of CUG	
88	incompatible dest	
90	non-existent CUG	
91	invalid transit ntwk selection	
95	invalid message	
97	message type not impl.	
99	parameter not impl.	
102	recovery on timer expiry	
103	parameter passed on	
110	message discarded	
111	protocol error	
127	interworking	
	all other values reserved	

3.10.1.14. Optional Parameters

The *Optional 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 = 0x050E          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                                                               \
/                      Optional Parameters                      /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Optional Parameters* parameter contains the following fields:

Optional Parameters field: (ISUP Optional Parameters)

The Optional Parameters field is formatted according to the format of the ISUP Optional Parameters Part [Q.763, T1.113] of the ISUP message, starting with the first byte of the first optional parameter in the ISUP Optional Parameters Part of the message and continuing through and including the ISUP End of Optional Parameters parameter [Q.763, T1.113].

The ISUP Optional Parameters from the ISUP message **MUST** be placed transparently in this fashion into the ISUA *Optional Parameters* parameter.

3.10.2. Parameters used in CS Messages

The sections (below) provide the format of the parameters used in ISUA Circuit Supervision (CS) messages.

3.10.2.1. Circuit Status

The *Circuit Status* parameter indicates the state of a circuit. The state of a circuit is maintained and obtained by the SG and communicated to the ASP.

The *Circuit 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 = 0x0510          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|         Circuit Id #1         |        Circuit State #1       |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                               .                               \
/                               .                               /
\                               .                               \
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|         Circuit Id #n         |        Circuit State #n       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Circuit Status* parameter contains (a list of) the following fields:

Circuit Id field: 16 bits (unsigned integer)

The Circuit Id field contains the circuit identifier for one circuit. This is the least significant bit aligned Circuit Identification Code (CIC) [Q.763, T1.113] associated with the circuit. Unused bits are coded zero (0).

For example, a 12-bit Circuit Identification Code (CIC) is formatted into the Circuit Id field as follows:

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

Circuit State field: 32-bits (integer)

The Circuit State field contains the least significant bit aligned Circuit State Indicator (CSI) [Q.763, T1.113] indicating the status of the circuit. Unused bits are coded zero (0).

For example, the ITU-T Circuit State Indicator (CSI) is formatted into the Circuit State field as follows:

```
 1       2                   3
 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|    CSI    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                    |MSB-----LSB|
```

The ITU-T Circuit State Indicator (CSI) [Q.763] can take on the following values:

CSI	**State**		**Blocking State**	
	Maint	**Call Proc**	**Hardware**	**Maint**
XX 00 00	transient	–	–	–
XX 00 11	unequipped			
00 01 00	equipped	ic busy	active	active
00 01 01				local
00 01 10				remote
00 01 11				both
00 10 00		og busy		active
00 10 01				local
00 10 10				remote
00 10 11				both
00 11 00		idle		active
00 11 01				local
00 11 10				remote
00 11 11				both
01 11 00			local	active
01 11 01				local
01 11 10				remote
01 11 11				both
10 11 00			remote	active
10 11 01				local
10 11 10				remote
10 11 11				both
11 11 00			both	active
11 11 01				local
11 11 10				remote
11 11 11				both

3.10.3. Other Parameters

3.10.3.1. Circuit Id

The *Circuit Id* parameter is used in the ISUA CP and CS Message Header to identify one or more circuits within the Application Server indicated by the *Routing Context* parameter (in the ISUA Message Header).

The *Circuit 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 = 0x0520          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Circuit Id                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Circuit Id field: 32-bit (unsigned integer)

The Circuit Id field contains the circuit identifier for the circuit within an Application Server that the sending ASP ro SGP is configured or registered to control and manage. This is the least significant bit aligned Circuit Identification Code (CIC) [Q.763, T1.113] associated with the circuit. Unused bits are coded zero (0).

For example, a 12-bit Circuit Identification Code (CIC) is formatted into the Circuit Id field 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|          CIC          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                        |MSB-----------------LSB|
```

a list of one or more 32-bit unsigned integers indexing the circuits within an Application Server that the sending ASP is configured or registered to control and manage.

If the *Circuit Id* parameter is present, it **SHOULD** be the first parameter in the message following the *Routing Context* as it defines the format and/or interpretation of the parameters which follow.

3.10.3.2. 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 = 0x0521          |          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 format used, and the ISUP and Call Control protocol (type, variant and version) used within the specific SS7 network.

3.10.3.3. 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 = 0x0522          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                 Local Routing Key Identifier                  |
+- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
\                                                               \
/                       Key parameter(s)                        /
\                                                               \
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

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

Local Routing Key Identifier: TLV

The *Local Routing Key Identifier* parameter 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		
Traffic Mode Type	Optional	
Network Appearance	Conditional	*1
Local Point Code	Mandatory	
Remote Point Code	Mandatory	
Circuit Id	Conditional	*2
Circuit Range	Conditional	*2

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.

Note 2: One of the *Circuit Id* or *Circuit Range* parameters **MUST** be present in the *Key* parameters.

3.10.3.4. Circuit Range

The *Circuit 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 = 0x0523          |            Length             |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|      Circuit Id Beg #1        |       Circuit Id End #1       |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
\                               .                               \
/                               .                               /
\                               .                               \
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|      Circuit Id Beg #n        |       Circuit Id End #n       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Circuit Range* parameter can contain (a list of) the following fields:

Circuit Id Beg field: 16-bits (unsigned integer)

The Circuit Id Beg field contains the circuit identifier for the circuit at the beginning of the range (inclusive). This is the least significant bit aligned Circuit Identification Code (CIC) [Q.763, T1.113] associated with the first circuit in the range. Unused bits are coded zero (0). The first and last circuit in the range **MAY** be the same circuit.

For example, a 12-bit Circuit Identification Code (CIC) is formatted into the Circuit Id Beg field as follows:

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

Circuit Id End field: 16-bits (unsigned integer)

The Circuit Id End field contains the circuit identifier for the circuit at the end of the range (inclusive). This is the least significant bit aligned Circuit Identification Code (CIC) [Q.763, T1.113] associated with the last circuit in the range. Unused bits are coded zero (0). The first and last circuit in the range **MAY** be the same circuit.

For example, a 12-bit Circuit Identification Code (CIC) is formatted into the Circuit Id End field as follows:

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

3.10.3.5. Local Point Code

The *Local Point Code* parameter appears in the *Routing Key* parameter in the *REG REQ* message. It is used in conjunction with an implied or specified *Network Appearance* parameter which also appears in the *Routing Key* to identify the local ISUP switch for which an ASP is registering.

The *Local 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 = 0x0524          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Point Code                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Local 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, ETSI 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|
```

3.10.3.6. Remote Point Code

The *Remote Point Code* parameter appears in the *Routing Key* parameter in the *REG REQ* message. It is used in conjunction with an implied or specified *Network Appearance* parameter which also appears in the *Routing Key* to identify the ISUP switch at the remote end of the ISUP circuits for which an ASP is registering.

The *Remote 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 = 0x0525          |          Length = 8           |
+- - - - - - - - - - - - - - - -+- - - - - - - - - - - - - - - -+
|                          Point Code                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Remote 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. For examples of point codes, see the *Local Point Code* parameter description.

4. Procedures

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

4.1. Procedures to Support Call Control

4.1.1. Receipt of Primitives from Call Control

Upon receiving a ISUP request or response primitive from the upper layer at an ASP or IPSP, the ISUA layer sends a corresponding ISUA Call Processing (CP) message (see Section 3) to its ISUA peer. The ISUA peer receiving the CP message delivers the corresponding ISUP primitive to Call Control at the IPSP or Nodal Interworking Function at the SG as illustrated in *Figure 4*. The mapping of ISUP primitives to ISUA CP Messages is listed in *Table 2* (see Section 1.6.1).

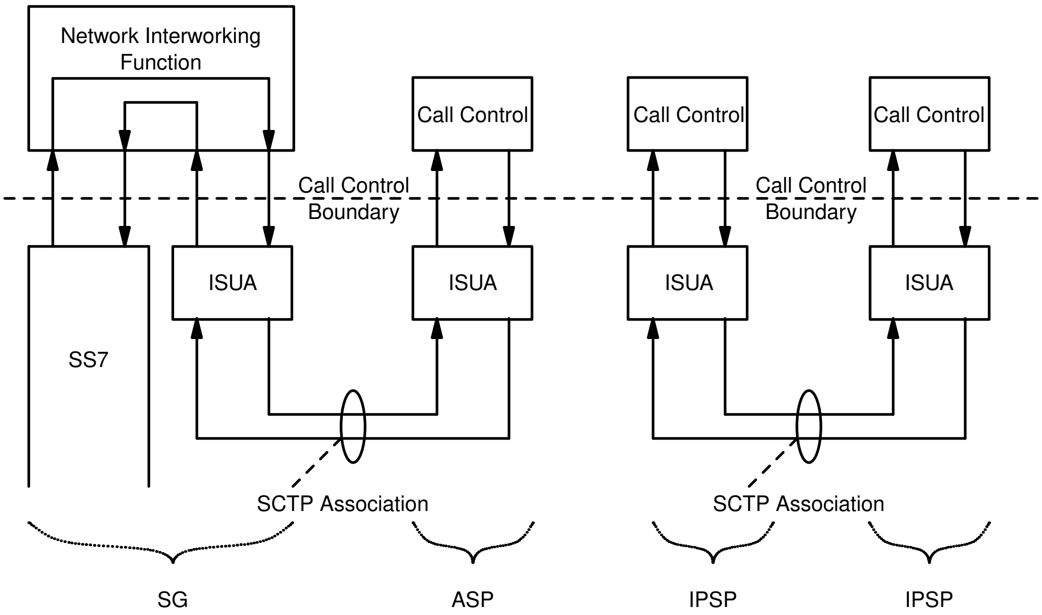

Figure 4. ISUA Layer Model

4.1.2. Receipt of Primitives from ISUP

Upon receiving a ISUP indication or confirmation primitive from ISUP at an SG, the Nodal Interworking Function passes the primitive to ISUA. The ISUA layer sends a corresponding ISUA Call Processing (CP) message (see Section 3) to its ISUA peer at the ASP.

The ISUA peer receiving the CP message delivers the corresponding ISUP primitive to Call Control at the ASP as illustrated in *Figure 5*. The mapping of ISUP primitives to ISUA CP Messages is listed in *Table 2* (see Section 1.6.1).

The ISUA Circuit Mapping Function (see Section 1.5.1.4)

For SETUP indications, the ISUA Circuit Mapping Function (CMF) determines the Application Server (AS) based on comparing the circuit 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 *CSET* message is constructed and issued on the corresponding SCTP association. The ISUA at the SG is also responsible for assigning and managing a Circuit Identifier which is sent to the ASP in the *CSET* message to identify the newly created call 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 ISUA messages with the correct Dialogue at the SG. The SG will have to access this stored information to continue processing the dialogue.

The ISUA Circuit Mapping Function (CMF) 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 ISUP

When ISUP Circuit Management indications are received (RESET, BLOCKING, UNBLOCKING, CCT GROUP QUERY), ISUP Management determines whether there are concerned local Call Control. When these local Call Control are in fact Application Servers, serviced by ASPs, ISUA circuit supervision is transparently informed with the RESET, BLOCKING, UNBLOCKING and CCT GROUP QUERY indication primitive upon which it formats and transfers the applicable CS message (*CRES*, *CBLO*, *CUBL* or *CQRY*) to the list of concerned ASPs.

The ISUA message distribution function determines the Application Server (AS) based on comparing the information in the ISUP 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 Call Processing (CP) 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. Call Processing (CP) 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 ISUA 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 ISUA layer will attempt to establish an SCTP association with the remote ISUA 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 ISUA layer. At the SGP or IPSP that initiated the request, the ISUA layer will send an M-SCTP_ESTABLISH confirm primitive to Layer Management when the association setup is complete. At the peer ISUA 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 ISUA 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 ISUA layer. At the ISUA Layer that initiated the request, the ISUA layer will send an M-SCTP_RELEASE confirm primitive to Layer Management when the association shutdown is complete. At the peer ISUA 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 ISUA layer simply maps the M-SCTP_STATUS request primitive to an SCTP-STATUS primitive to the SCTP layer. When the SCTP responds, the ISUA layer maps the association status information to an M-SCTP_STATUS confirm primitive. No peer protocol is invoked.

Similar LM-to-ISUA-to-SCTP and SCTP-to-ISUA-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 ISUA *Notify (NTFY)* or *Error (ERR)* message respectively. These indications can also be generated based on local ISUA events.

An M-ASP_STATUS request primitive supports a Layer Management query of the status of a particular local or remote ASP. The ISUA layer responds with the status in an M-ASP_STATUS confirm primitive. No ISUA peer protocol is invoked. An M-AS_STATUS request supports a Layer Management query of the status of a particular AS. The ISUA responds with an M-AS_STATUS confirm primitive. No ISUA 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 ISUA 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 (ASPUP)*, *ASP Down (ASPDN)*, *ASP Active (ASPAC)* and *ASP Inactive (ASPIA)* messages to the remote ISUA peer at an SGP or IPSP.

4.2. Procedures to Support the Management of SCTP Associations

4.2.1. Receipt of ISUA Peer Management Messages

Upon successful state changes resulting from reception of *ASP Up (ASPUP)*, *ASP Down (ASPDN)*, *ASP Active (ASPAC)* and *ASP Inactive (ASPIA)* messages from a peer ISUA, the ISUA 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 ISUA *Notify (NTFY)* or *Error (ERR)* message. These indications can also be generated based on local ISUA events.

All MGMT, ASPSM, ASPTM and RKM messages, except *BEAT*, *BEAT ACK* and *NTFY*, **SHOULD** be sent with sequenced delivery to ensure ordering. All MGMT, ASPSM and RKM messages, with the exception of *BEAT*, *BEAT ACK* and *NTFY* 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*, *BEAT ACK*, and *NTFY* messages **MAY** be sent using out-of-order delivery, and **MAY** be sent on any stream.

4.3. AS and ASP State Maintenance

The ISUA 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 ISUA message distribution function. Similarly, where IPSPs use ISUA in a point-to-point fashion, the ISUA 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 ISUA layer in the SGP. The state of a particular ASP in a particular AS changes due to events. The events include: .bu reception of messages from the peer ISUA layer at the ASP;

- reception of some messages from the peer ISUA 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 ISUA 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 ISUA messages, with the exception of *Heartbeat (BEAT)*, *ASP Down Ack (ASPDN ACK)* and *Error (ERR)* messages.

ASP-INACTIVE:
The remote ISUA 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 CP or CS messages for the AS for which the ASP is inactive.

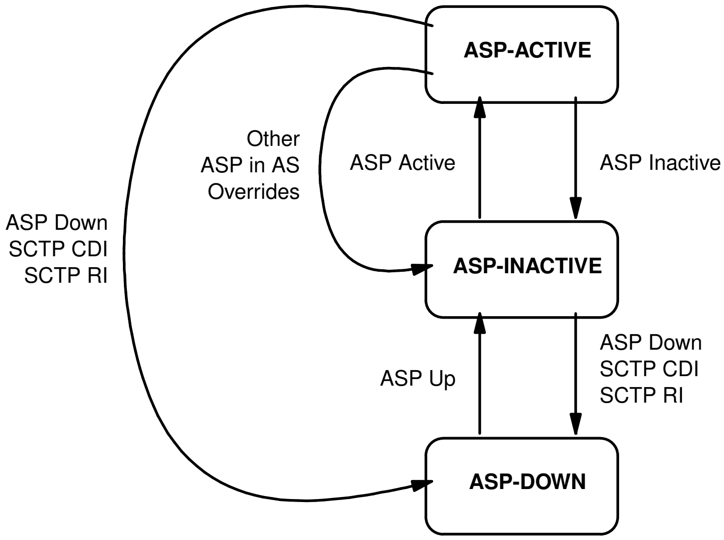

Figure 5. ASP State Transition Diagram (Per AS)

ASP-ACTIVE: The remote ISUA 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 (ISUA) on an SGP. The local SCTP layer will send this indication when it detects the loss of 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 (ISUA) 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 ISUA 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.

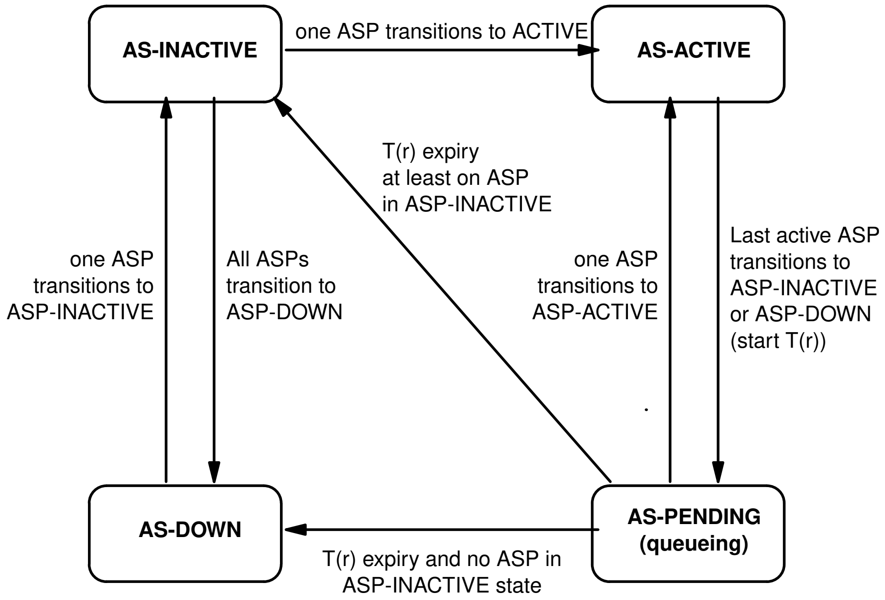

Figure 6. AS State Transition Diagram

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. ISUA 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 local Call Control is ready, the local ISUA 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).

If the ISUA 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, Call Control will be informed of the status of any affected ISUP circuit through the use of RESET, BLOCKING and UNBLOCKING 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 ISUA 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 ISUA 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 (ASPUP)* message, indicating that the ASP ISUA peer is available. The ASP is always the initiator of the *ASP Up (ASPUP)* 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 ISUA management function.

When an *ASP Up (ASPUP)* 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 (ASPUP)* message contains an ASP Identifier, the SGP should save the ASP Identifier for that ASP. The SGP **MUST** send an *ASP Up Ack (ASPUP ACK)* message in response to a received *ASP Up (ASPUP)* 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 (ASPUP ACK)* message, the SGP responds to an *ASP Up (ASPUP)* message with an *Error (ERR)* message with Reason "Refused - Management Blocking".

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

When the ASP sends an *ASP Up (ASPUP)* message it starts timer T(ack). If the ASP does not receive a response to an *ASP Up (ASPUP)* message within T(ack), the ASP **MAY** restart T(ack) and resend *ASP Up (ASPUP)* messages until it receives an *ASP Up Ack (ASPUP ACK)* message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of *ASP Up (ASPUP)* 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 (ASPUP ACK)* message before sending any other ISUA messages (e.g, ASP Active or REG REQ). If the SGP receives any other ISUA messages before *ASPUP* message is received (other than ASPDN - see Section 4.3.4.2), the SGP **SHOULD** discard them.

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

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

4.3.4.1.1. ISUA Version Control

If an *ASP Up (ASPUP)* message with an unsupported version is received, the receiving end responds with an *Error (ERR)* 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 (ERR)* 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 (ASPUP)* message with an *ASP Up Ack (ASPUP ACK)* message, it responds to an *ASP Up (ASPUP)* message with an *Error (ERR)* 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 (ASPDN)* 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 CP, CS 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 ISUA 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 (ASPDN)* 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 (ASPDN ACK)* message to the ASP.

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

At the ASP, the *ASP Down Ack (ASPDN 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 (ASPDN)* 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 (ASPDN)* message it starts timer T(ack). If the ASP does not receive a response to an *ASP Down (ASPDN)* message within T(ack), the ASP **MAY** restart T(ack) and resend *ASP Down (ASPDN)* messages until it receives an *ASP Down Ack (ASPDN ACK)* message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of *ASP Down (ASPDN)* 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 (ASPUP ACK)* message from the SGP or IPSP, the ASP **MAY** send an *ASP Active (ASPAC)* 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 ISUA management function. Whenever an ASP wishes to process the traffic for more than one Application Server across a common SCTP association, the *ASP Active (ASPAC)* message(s) **SHOULD** contain a list of one or more Routing Contexts to indicate for which Application Servers the *ASP Active (ASPAC)* message applies. It is not necessary for the ASP to include all Routing Contexts of interest in a single *ASP Active (ASPAC)* message, thus requesting to become active in all Routing Contexts at the same time. Multiple *ASP Active (ASPAC)* messages **MAY** be used to activate within the Application Servers independently, or in sets. Whenever an *ASP Active (ASPAC)* 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 (ASPAC ACK)* messages, including the associated Routing Context(s) and reflecting any Traffic Mode Type values present in the related *ASP Active (ASPAC)* message. The Routing Context parameter **MUST** be included in the *ASP Active Ack (ASPAC ACK)* message(s) if the received *ASP Active (ASPAC)* message contained any Routing Contexts. Depending on any Traffic Mode Type request in the *ASP Active (ASPAC)* 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 CP messages before an *ASP Active (ASPAC)* message is received, the SGP or IPSP **MAY** discard them. By sending an *ASP Active Ack (ASPAC 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 CP messages for the related Routing Context(s) before receiving an *ASP Active Ack (ASPAC ACK)* message, or it will risk message loss.

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

Whenever an "out-of-the-blue" *ASP Active (ASPAC* 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 (ASPAC ACK)* message in response to a received *ASP Active (ASPAC)* 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 (ASPAC 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 CP message(s) before the *ASP Active Ack*

(ASPAC ACK) message as the ASP Active Ack and CP 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 (ASPAC ACK)* message.

When the ASP sends an *ASP Active (ASPAC)* message it starts timer T(ack). If the ASP does not receive a response to an *ASP Active (ASPAC)* message within T(ack), the ASP **MAY** restart T(ack) and resend *ASP Active (ASPAC)* messages until it receives an *ASP Active Ack (ASPAC ACK)* message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of *ASP Active (ASPAC)* 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 ISUA layer: Override, Load-share and Broadcast. When included, the Traffic Mode Type parameter in the *ASP Active (ASPAC)* 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 (ASPAC)* message is unsupported or incompatible with the mode currently configured for the AS, the SGP responds with an *Error (ERR)* 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 (ASPAC)* 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 (ASPAC)* message at an SGP causes the (re)direction of all traffic for the AS to the ASP that sent the *ASP Active (ASPAC)* 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 (NTFY)* 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 (ASPAC)* message at an SGP or IPSP causes the direction of traffic to the ASP sending the *ASP Active (ASPAC)* 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 CCmessage.

An SGP or IPSP, upon reception of an *ASP Active (ASPAC)* 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 CP 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 (ASPAC)* message at an SGP or IPSP causes the direction of traffic to the ASP sending the *ASP Active (ASPAC)* 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 (ASPAC)* 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 CP 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 (ASPAC ACK)* message. An ASP receiving an *ASP Active Ack (ASPAC 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 (ASPIA)* 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 ISUA management function. Whenever an ASP is processing the traffic for more than one Application Server across a common SCTP association, the *ASP Inactive (ASPIA)* message contains one or more Routing Contexts to indicate for which Application Servers the *ASP Inactive (ASPIA)* message applies. Whenever an

ASP Inactive (ASPIA) 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 (ASPAC)* message, the ASP that sends the *ASP Inactive (ASPIA)* message is already considered by the SGP to be in state ASP-INACTIVE. An *ASP Inactive Ack (ASPIA 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 (NTFY)* message ("Insufficient ASP resources active in AS") **MAY** be sent to all inactive ASPs, if required. An *ASP Inactive Ack (ASPIA 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 (NTFY)* message ("Insufficient ASP resources active in AS") **MAY** be sent to all inactive ASPs, if required. An *ASP Inactive Ack (ASPIA 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 (ASPIA ACK)* messages **MAY** be used in response to an *ASP Inactive (ASPIA)* 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 (ERR)* ("Invalid Routing Context") message for each invalid or not configured Routing Context value in a received *ASP Inactive (ASPIA)* message.

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

At the ASP, the *ASP Inactive Ack (ASPIA 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 (ASPIA)* 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 (ASPIA)* message it starts timer T(ack). If the ASP does not receive a response to an *ASP Inactive (ASPIA)* message within T(ack), the ASP **MAY** restart T(ack) and resend *ASP Inactive (ASPIA)* messages until it receives an *ASP Inactive Ack (ASPIA ACK)* message. T(ack) is provisionable, with a default of 2 seconds. Alternatively, retransmission of *ASP Inactive (ASPIA)* 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 (NTFY)* 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 (ASPAC)* 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 (ASPIA 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 (NTFY)* 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 (NTFY)* 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 (NTFY)* 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 (NTFY)* ("AS-PENDING") message is sent by an SGP that now has no ASPs active to service the traffic, or where a *Notify NTFY* ("Insufficient ASP resources active in AS") message **MUST** be sent in the Load-share or Broadcast mode, the *Notify (NTFY)* 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 (NTYF)* 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 ISUA peer may optionally send *Heartbeat (BEAT)* messages periodically, subject to a provisionable timer T(beat). Upon receiving a *Heartbeat (BEAT)* message, the ISUA peer **MUST** respond with a *Heartbeat Ack (BEAT ACK)* message.

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

The *Heartbeat (BEAT)* message may optionally contain an opaque Heartbeat Data parameter that **MUST** be echoed back unchanged in the related *Heartbeat Ack (BEAT ACK)* message. The sender, upon examining the contents of the returned *Heartbeat Ack (BEAT ACK)* message, **MAY** choose to consider the remote ISUA 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 (REG RSP)* 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 (REG RSP)* 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 (ERR)* 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 (REG RSP)* 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 (REG RSP)* 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 (REG RSP)* 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 (REG RSP)* 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 Circuit and Call State

4.5.1. At an SGP

Upon receiving a RESET, BLOCKING, UNBLOCKING, CCT GROUP QUERY indication primitive from the nodal interworking function at an SGP, the SGP ISUA layer will send a corresponding ISUA Circuit Supervision (CS) *CRES*, *CBLO*, *CUBL* or *CQRY* message (see Section 3) to the ISUA peers at concerned ASPs. The ISUA layer must fill in various fields of the CS messages consistently with the information received in the primitives.

CS 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 ISUA Circuit Supervision (CS) message from the remote ISUA Peer, the ISUA layer invokes the appropriate primitive indications to the resident Call Control. Local management is informed.

Whenever a local event has caused the change in state of ISUP circuits, the ISUA layer at the ASP **SHOULD** pass up appropriate indications in the primitives to the ISUA User, as though equivalent CS messages were received. For example, the loss of an SCTP association to an SGP may cause the software blocking of a set of ISUP circuits. BLOCKING indication primitives to the ISUA User are appropriate.

4.5.2.2. Multiple SG Configurations

At an ASP, upon receiving an ISUA Circuit Supervision (CS) message from the remote ISUA Peer, the ISUA layer updates the status of the affected circuit(s) via the originating SG and determines, whether or not the overall status of the affected circuits(s) has changed. If so, the ISUA layer invokes the appropriate primitive indications to the resident Call Control [5]. Local management is informed.

4.5.3. ASP Auditing

An ASP may optionally initiate an audit procedure to inquire of an SG the status of a circuit or circuit(s). A *Circuit Query (CQRY)* message is sent from the ASP to the SGP requesting the current status of one or more circuits.

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

- Periodic: A Timer originally set upon reception of a *CBLO* message has expired without a subsequent *CUBL* or *CQRY* message updating the circuit status of the affected circuits. The Timer is reset upon issuing a *CQRY*. In this case the *CQRY* is sent to the SGP that originally sent the CS message.
- Isolation: The ASP is newly ASP-ACTIVE or has been isolated from an SG for an extended period. The ASP **MAY** request status of one or more ISUP circuits for which it expects to communicate.

The SGP **SHOULD** either respond to a *CQRY* messages with CS messages indicating the status of the circuit, or **SHOULD** respond with an *ERR* ("Circuit Status Unknown") or *ERR* ("Call Reference Status Unknown") message for each *Circuit Id* or *Call Reference* requested in the *CQRY* message.

The status of each ISUP circuit requested is indicated in a *CQRY* response message. If the SGP cannot return information on the status of the ISUP circuit or call reference, the SGP responds with an ERR ("Circuit Status Unknown") or ERR ("Call Reference Status Unknown") with a list of all the *Circuit Ids* and *Call References* for which the SGP cannot provide information.

In some cases, the SGP **MAY** chose not to respond to a *CQRY* message or a component of a *CQRY* message on the basis of policy [6].

Any *CQRY* message in response to a *CQRY* message **MAY** contain a list of *Call References*.

4.5.4. ISUP − ISUA Interworking at the SG

On the SG, the ISUP routing or interworking function determines that the message must be sent to an AS via the ISUA stack, based on information in the incoming message. The ISUA 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 ISUA 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 Call Control

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

The ISUA needs to setup and maintain the appropriate SCTP association to the selected endpoint. ISUA also manages the usage of SCTP streams. The address information passed by the ISUA-user at an ASP must contain:

(1) a valid circuit identifier to specify an ISUP circuit in the SS7 network via the appropriate SCTP association to a SG

(2) a valid IP address or host name to reach another ASP in the IP network via the appropriate SCTP association.

5. Examples of ISUA 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 ISUA 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 ISUA 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 (ASPAC)* 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 ISUA 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 ISUA 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 ISUA peer (ISUA heartbeat loss or detection of SCTP failure), the initial SG-ASP1 *ASP Inactive (ASPIA)* 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 (NTFY)* 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 ISUA peer (ISUA heartbeat loss or detection of SCTP failure), the first SG-ASP1 *ASP Inactive (ASPIA)* message exchange would not occur.

5.1.3. ISUP/CC Service Translation Examples

When the ISUA layer on the ASP has a CP 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 CP message, fill ISUA Message Header, fill Common Header

(5) Send the CP message to the remote ISUA peer in the SG, over the SCTP association

When the ISUA layer on the SG has a CP 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 CP message, fill in ISUA Message Header, fill in Common Header

(5) Send the CP message to the remote ISUA 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 ISUA connectivity

The following shows an example establishment of ISUA 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 ISUA 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 ISUA 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.

+-------------------------------ISUP_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 (NTFY)* messages declaring the availability of the backup are not sent.

6. Security

6.1. Introduction

ISUA is designed to carry signalling messages for telephone services. As such, ISUA 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, ISUA has the following security objectives:

- Availability of reliable and timely user data transport.
- Integrity of user data transport.
- Confidentiality of user data.

ISUA 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 ISUA 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" [RFC 2196] should be consulted for guidance.

SS7 networks have a different security model that IP networks. Traditionally, the PSTN has been a private and closed network, where in many cases, to get connectivity, one would need to be a service provider and negotiate physical connections to the PSTN.

The Internet has a slightly different security mode, on which connectivity is a primary goal. When signalling protocols are run over IP, one must be aware that it is impossible to guarantee that the IP network will be physically separate from another IP network. Firewalls and gateways may create an illusion of separateness, but do not guarantee this. One mis-configured parameter in a firewall could leave a dangerous security hole.

The most reasonable security model for ISUA is to assume a virtual private network (VPN) type of security, where TLS of IPsec are used to encrypt traffic between nodes.

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.

6.4. IPsec Usage

All ISUA implementations **MUST** support IPsec ESP [RFC 2406] in transport mode with non-null encryption and authentication algorithms to provide per-packet authentication, integrity protection and confidentiality, and **MUST** support the replay protection mechanisms of IPsec.

ISUA implementations **MUST** support IKE for peer authentication, negotiation of security associations, and key management, using IPsec DOI [RFC 2407]. ISUA implementations **MUST** support peer authentication using a pre-shared key, and **MAY** support certificate-authentication using the public key encryption methods outlined in IKE sections 5.2 and 5.3 [RFC 2409] **SHOULD NOT** be used.

Conforming implementations **MUST** support both IDE Main Mode and Aggressive Mode. When pre-shared keys are used for authentication, IKE Aggressive Mode **SHOULD** be used, and IKE Main Mode **SHOULD NOT** be used. When digital signatures are used for authentication, either IKE Main Mode or IKE Aggressive Mode **MAY** be used.

When digital signatures are used to achieve authentication, an IKE negotiator **SHOULD** use IKE Certificate Request Payload(s) to specify the certificate authority (or authorities) that are trusted in accordance with it local policy. IKE negotiators **SHOULD** use pertinent certificate revocation checks before accepting a PKI certificate for use in IKE's authentication procedures.

The Phase 2 Quick Mode exchanges used to negotiate protection for ISUA connections **MUST** explicitly carry the Identity Payload fields (IDci and IDcr). The DOI provides for several types of implementations, each ID Payload **MUST** carry a single IP address and a single non-zero port number, and **MUST NOT** use the IP Subnet or IP Address Range formats. This allows the Phase 2 security association to correspond to specific TCP and SCTP connections.

Since IPsec acceleration hardware may only be able to handle a limited number of active IKE Phase 2 SAs, Phase 2 delete messages may be sent for idle SAs, as a means of keeping the number of active Phase 2 SAs to a minimum. The receipt of an IKE Phase 2 delete message **SHOULD NOT** be interpreted as a reason for tearing down a ISUA connection. Rather, it is preferable to leave the connection up, and if additional traffic is sent on it, to bring up another IKE Phase 2 SA to protect it. This avoids the potential for continually bringing connections up and down.

6.5. TLS Usage

A ISUA peer that initiates a connection to another ISUA peer acts as a TLS client according to TLS [RFC 2246, RFC 3436], and a ISUA peer that accepts a connection acts as a TLS server. ISUA peers implementing TLS for security **MUST** mutually authenticate as part of TLS session establishment. To ensure mutual authentication, the ISUA node acting as TLS server must request a certificate from the ISUA node acting as TLS client, and the ISUA node acting as TLS client **MUST** be prepared to supply a certificate on request.

ISUA peers supporting TLS **MUST** be able to negotiate the following TLS cipher suites:

TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_3DES_EDE_CBC_SHA

ISUA nodes **MAY** negotiate other TLS cipher suites.

6.6. Peer-to-Peer Considerations

As with any peer-to-peer protocol, proper configuration of the trust model within a ISUA peer is essential to security. When certificates are used, it is necessary to configure the root certificate authorities trusted by the ISUA peer. These root CAs are likely to be unique to ISUA usage and distinct from the root CAs that might be trusted for other purposes such as Web browsing. In general, it is expected that those root CAs will be configured to reflect the business relationships between the organization hosting the ISUA peer and other organizations. Therefore, a ISUA peer will typically not be configured to allow connectivity with any arbitrary peer. With certificate authentication, ISUA peers might not be known beforehand, and therefore peer discovery may be required.

Note that IPsec is considerably less flexible than TLS when it comes to configuring root CAs. Since use of Port identifiers is prohibited within IKE Phase 1, within IPsec it is not possible to uniquely configure trusted root CAs for each application individually; the same policy must be used for all applications. This implies, for example, that a root CA trusted for use with ISUA must also be trusted to protect SNMP. These restrictions can be awkward at best. Since TLS supports application-level granularity in certificate policy, TLS **SHOULD** be used to protect ISUA connections between administrative domains. IPsec is most appropriate for intra-domain usage when pre-shared keys are used as a security mechanism.

When pre-shared key authentication is used with IPsec to protect ISUA, unique pre-shared keys are configured with ISUA peers, who are identified by their IP address (Main Mode), or possibly their FQDN (Aggressive Mode). As a result, it is necessary for the set of ISUA peers to be known beforehand. Therefore, peer discovery is typically not necessary.

The following is intended to provide some guidance on the issue.

It is recommended that a ISUA peer implement the same security mechanism (IPsec or TLS) across all its peer-to-peer connections. Inconsistent use of security mechanisms can result in redundant security mechanisms being used (e.g. TLS over IPsec) or worse, potential security vulnerabilities. When IPsec is used with ISUA, a typical security policy for outbound traffic is "Initiate IPsec, from me to any, destination port ISUA"; for inbound traffic, the policy would be "Require IPsec, from any to me, destination port ISUA".

This policy causes IPsec to be used whenever a ISUA peer initiates a connection to another ISUA peer, and to be required whenever an inbound ISUA connection occurs. This policy is attractive, since it does not require policy to be set for each peer or dynamically modified each time a new ISUA connection is created; an IPsec SA is automatically created based on a simple static policy. Since IPsec extensions are typically not available to the sockets API on most platforms, and IPsec policy functionality is implementation dependent, use of a simple static policy is the often the simplest route to IPsec-enabling a ISUA implementation.

One implication of the recommended policy is that if a node is using both TLS and IPsec, there is not a convenient way in which to use either TLS or IPsec, but not both, without reserving an additional port for TLS usage. Since ISUA uses the same port for TLS and non-TLS usage, where the recommended IPsec policy is put in place, a TLS- protected connection will match the IPsec policy, and both IPsec and TLS will be used to protect the ISUA connection. To avoid this, it would be necessary to plumb peer-specific policies either statically or dynamically.

If IPsec is used to secure ISUA peer-to-peer connections, IPsec policy **SHOULD** be set so as to require IPsec protection for inbound connections, and to initiate IPsec protection for outbound connections. This can be accomplished via use of inbound and outbound filter policy.

7. IANA Considerations

7.1. SCTP Payload Protocol ID

IANA has assigned a ISUA value for the Payload Protocol Identifier in the SCTP DATA chunk. The following SCTP Payload Protocol Identifier is registered:

ISUA "5"

The SCTP Payload Protocol Identifier value "5" **SHOULD** be included in each SCTP DATA chunk, to indicate that the SCTP is carrying the ISUA 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.

> **EDITOR'S NOTE:–** The value shown above as "5" is to be assigned by IANA an may change in future versions of this document.

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 ISUA. 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. Timer Values

Following are the **RECOMMENDED** timer values for ISUA timers:

Timer	Description	Value
Ta	–	2 seconds
Tr	–	2 seconds
T(ack)	Inactivity Send Timer	7 minutes
T(ias)	Inactivity Receive Timer	15 minutes
T(beat)	Heartbeat Timer	30 seconds

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 ISUA and was included here.

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:–** 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:

- 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.
- The data parameter will normally be the final one in the message.
- The receiver **SHOULD** accept parameters in any order, except where explicitly mandated.

[3] **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 ISUP 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.

[4] **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.

For application of the *Correlation Id* parameter see *CORID* [CORID].

[5] **IMPLEMENTATION NOTE:–** To accomplish the handling of CS messages from multiple SGs in a multiple SG configuration, the ISUA layer at an ASP maintains the status of circuits via each SG.

[6] **IMPLEMENTATION NOTE:–** For example, an SGP **MAY** chose to not respond to a request for the circuit status of a specific circuit in the *CQRY* message because the ASP that issued the *CQRY* message is not authorized to obtain information concerning the status of the circuit as requested.

References

RFC 2960.
R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, H. Kalla, L. Zhang and V. Paxson, “Stream Control Transmission Protocol (SCTP),” RFC 2960, The Internet Society (February 2000). [Normative]

Q.761. ITU, “Signalling System No. 7 – Functional Description of the ISDN User Part,” ITU-T Recommendation Q.761, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). [Informative]

T1.113.
ANSI, “Signalling System No. 7 – ISDN User Part,” ANSI T1.113, American National Standards Institue (1992). [Informative]

RFC 2719.
L. Ong, I. Rytina, M. Holdrege, L. Coene, M.-A. Garcia, C. Sharp, I. Juhasz, H. P. Lin and HannsJ. Schwarzbauer, “Framework Architecture for Signaling Transport,” RFC 2719, The Internet Society (October, 1999). [Informative]

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). [Informative]

T1.111.
ANSI, “Signalling System No. 7 – Message Transfer Part,” ANSI T1.111, American National Standards Institue (1992). [Informative]

Q.724. ITU, “Signalling System No. 7 – Telephone User Part – Signalling Procedures,” ITU-T Recommendation Q.724, ITU-T Telecommunication Standardization Sector of ITU, Geneva (November 1988). (Previously "CCITT Recommendation")

Q.764. ITU, “Signalling System No. 7 – ISDN User Part Signalling Procedures,” ITU-T Recommendation Q.764, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

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). [Informative]

M3UA. G. Sidebottom, K. Morneault and J. Pastor-Balbas, (eds), “Signaling System 7 (SS7) Message Transfer Part 3 (MTP3) - User Adaptation Layer (M3UA),” RFC 3332, Internet Engineering Task Force - Signalling Transport Working Group (September, 2002).

RFC 2119.
S. Bradner, “Key words for use in RFCs to Indicate Requirement Levels,” RFC 2119 - BCP 14, Internet Engineering Task Force (March 1997). [Normative]

Q.763. ITU, “Signalling System No. 7 – Formats and Codes of the ISDN User Part,” ITU-T Recommendation Q.763, ITU-T Telecommunication Standardization Sector of ITU, Geneva (March 1993). (Previously "CCITT Recommendation")

RFC 2279.
F. Yergenau, “UTF-8, a transformation format of ISO 10646,” RFC 2279, Internet Engineering Task Force (January 1998). [Normative]

CORID.
B. Bidulock, “Correlation Id and Heartbeat Procedures Supporting Lossless Fail-Over,” <draft-bidulock-sigtran-corid-01.txt>, Internet Engineering Task Force - Signalling Transport Working Group (January 2, 2003). [Informative]

RFC 2196.
B. Y. Frazer, “Site Security Handbook,” RFC 2196, Internet Engineering Task Force (September 1997). [Normative]

RFC 2406.
S. Kent, R. Atkinson, “IP Encapsulating Security Payload (ESP),” RFC 2406, Internet Engineering Task Force (November 1998). [Normative]

RFC 2407.
D. Piper, “The Internet IP Security Domain of Interpretation for ISAKMP,” RFC 2407, Internet Engineering Task Force (November 1998). [Normative]

RFC 2409.
D. Harkins, D. Carrel, “The Internet Key Exchange (IKE),” RFC 2409, Internet Engineering Task Force (November 1998). [Normative]

RFC 2246.
T. Dierke, C. Allen, “The TLS Protocol - Version 1.0,” RFC 2246, The Internet Society (January 1999). [Normative]

RFC 3436.
A. Jungmaier, E. Rescorla and M. Tuxen, “Transport Layer Security over Stream Control Transmission Protocol,” RFC 3436, The Internet Society (December 2002). [Normative]

RFC 2434.
T. Narten, H. T. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” RFC 2434, The Internet Society (October, 1998). [Normative]

Author's Addresses

Brian Bidulock
OpenSS7 Corporation
1469 Jeffreys Crescent
Edmonton, AB T6L 6T1
Canada

Phone: +1-780-490-1141
Email: bidulock@openss7.org
URL: http://www.openss7.org/

This draft expires July 2003.

Appendices

A. Operational Considerations

A.1. Signalling Network Architecture

A Signalling Gateway is used to support the transport of Call Control signalling traffic received from the SS7 network to multiple distributed ASPs (e.g., MGCs and IP Databases). Clearly, the ISUA 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 ISUA 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*.

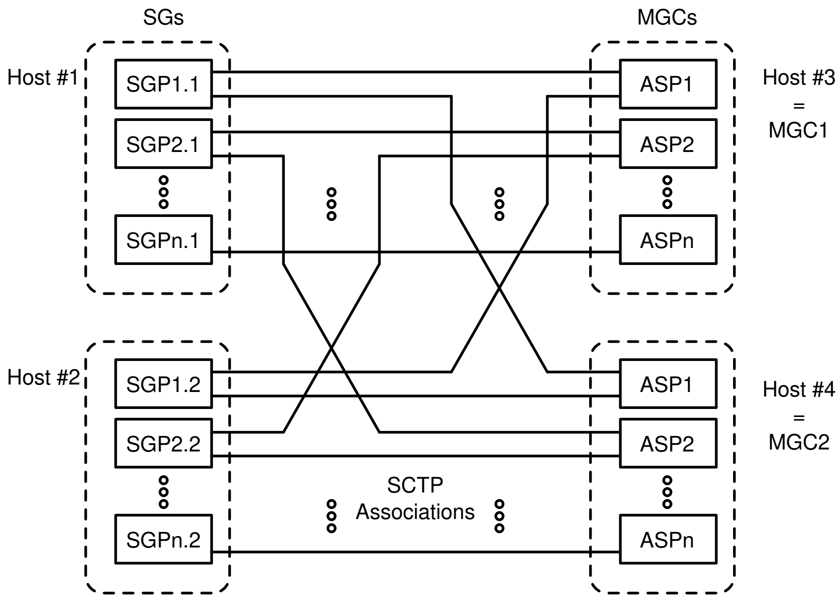

Figure 7. Physical Model

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.

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 ISUA 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, load-sharing, 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. ISUA 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, load-sharing 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 load-share 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 Circuit Identification Codes (CICs), the Operator is implicitly splitting up control of the related circuit groups. Some CIC value range assignments may interfere with ISUP circuit supervision procedures.

In the process of fail-over, 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, load-share 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 load-sharing model is possible, where the signalling messages are load-shared 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 Call Control messages over the SGPs should be done in such a way to minimize message mis-sequencing, 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 load-shared between them. Other models are also possible, subject to the limitations of the local SS7 network provisioning guidelines.

From the perspective of the ISUA 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 acknowledgment to the ASP to indicate that the ASP is actively handling traffic for that destination, subsystem 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, load-sharing and broadcast models, in the event of failures, recovery and maintenance activities. The ASP ISUA may also use this information for congestion avoidance purposes. The distribution of the Call Control messages over the SGPs should be done in such a way to minimize message mis-sequencing, as required by the some ISUP applications.

List of Tables

List of Illustrations

Table of Contents

Copyright Statement