Network Working Group
Request for Comments: 3094
Category: Informational

D. Sprague
R. Benedyk
D. Brendes
J. Keller
Tekelec

April 2001

Tekelec's Transport Adapter Layer Interface

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

IESG Note:

Readers should note that this memo presents a vendor's alternative to standards track technology being developed by the IETF SIGTRAN Working Group. The technology presented in this memo has not been reviewed by the IETF for its technical soundness or completeness. Potential users of this type of technology are urged to examine the SIGTRAN work before deciding to use the technology described here.

Abstract

This document proposes the interfaces of a Signaling Gateway, which provides interworking between the Switched Circuit Network (SCN) and an IP network. Since the Gateway is the central point of signaling information, not only does it provide transportation of signaling from one network to another, but it can also provide additional functions such as protocol translation, security screening, routing information, and seamless access to Intelligent Network (IN) services on both networks.

The Transport Adapter Layer Interface (TALI) is the proposed interface, which provides TCAP (Transaction Capability Application Part), ISUP (ISDN User Part), and MTP (Mail Transport Protocol) messaging over TCP/IP. In addition, TALI provides SCCP (Signalling Connection Control Part) Management (SCMG), MTP Primitives, dynamic registration of circuits, and routing of call control messages based on circuit location.

Table of Contents

1. Introduction

This document is organized into the following 6 sections:

- Introduction to the document
- Overview of the TALI Protocol
- TALI Version 1.0
- TALI Version 2.0
- Success/Failure Codes
- Security Considerations

The following terms are used throughout this document.

Circuit Identification Code (CIC):
A field identifying the circuit being setup or released. Depending on SI and MSU Type, this field can be 12, 14 or 32 bits.

Changeover/Changeback (co/cb):
SS7 MTP3 procedure related to link failure and re-establishment.

Far End (FE):
The remote endpoint of a socket connection.

Far End Allowed (FEA):
The FE is ready to use the socket for service PDUs.

Far End Prohibited (FEP):
The FE is not ready to use the socket for service PDUs.

Intelligent Network (IN):
A network that allows functionality to be distributed flexibly at a variety of nodes on and off the network and allows the architecture to be modified to control the services.

Management ATM Adaptation Layer (MAAL):
This layer is a component of SAAL. This layer maps requests and indications between the System Management for the SG and the other SAAL layers. MAAL includes interfaces to/from SSCOP, SSCF, and system management. More information can be found in T1.652.

Media Gateway (MG):
A MG terminates SCN media streams, packetizes the media data, if it is not already packetized, and delivers packetized traffic to the packet network. It performs these functions in reverse order for media streams flowing from the packet network to the SCN.

Media Gateway Controller (MGC):
An MGC handles the registration and management of resources at the MG. The MGC may have the ability to authorize resource usage based on local policy. For signaling transport purposes, the MGC serves as a possible termination and origination point for SCN application protocols, such as SS7 ISDN User Part and Q.931/DSS1.

MTP3 Framing (MTP3F):
TALI does not require full MTP3 procedures support but rather uses the MTP3 framing structure (ie: SIO, Routing Label, etc)

Near End (NE):
The local endpoint of a socket connection.

Near End Allowed (NEA):
The NE is ready to use the socket for service PDUs.

Near End Prohibited (NEP):
The NE is not ready to use the socket for service PDUs.

Q.BICC ISUP:
An ISUP+ variant that uses 32 bit CIC codes instead of 14/12 bit CIC codes. ISUP+, or Q.BICC ISUP, is based on the Q.765.BICC specification currently being developed in ITU Study Group 11.

Signaling ATM Adaptation Layer (SAAL):
This layer is the equivalent of MTP-2 for ATM High Speed Links carrying SS7 Traffic as described in GR-2878-CORE [8]. SAAL includes SSCF, SSCOP and MAAL.

Signaling Gateway (SG):
An SG is a signaling agent that receives/sends SCN native signaling at the edge of the IP network. The SG function may relay, translate or terminate SS7 signaling in an SS7-Internet Gateway. The SG function may also be co-resident with the MGC/MG functions to process SCN signaling associated with line or trunk terminations controlled by the MG (e.g., signaling backhaul).

Service Specific Coordination Function (SSCF):
This layer is a component of SAAL. This layer maps the services provided by the lower layers of the SAAL to the needs of a specific higher layer user. In the case of the STP, the higher layer user is the MTP-3 protocol, and the SSCF required is that as defined by T1.645: SSCF for Support of Signaling at the Network Node Interface (SSCF at the NNI). More information can be found in T1.645. SSCF provides the interface between SSCOP and MTP3 and includes the following functions:

- Local Retrieve of messages to support link changeover procedures
- Flow control with four levels of congestion

Switched Circuit Network (SCN):
The term SCN is used to refer to a network that carries traffic within channelized bearers of pre-defined sizes. Examples include Public Switched Telephone Networks (PSTNs) and Public Land Mobile Networks (PLMNs). Examples of signaling protocols used in SCN include Q.931, SS7 MTP Level 3 and SS7 Application/User parts.

Service Specific Connection Oriented Protocol (SSCOP):
This layer is a component of SAAL. This layer provides reliable point to point data transfer with sequence integrity and error recovery by selective retransmission. Protocol layer interfaces are described in T1.637. Aspects of the protocol include flow control, connection control, error reporting to layer management, connection maintenance in the prolonged absence of data transfer, local data retrieval by the user of the SSCOP, error detection of protocol control information and status reporting. SSCOP provides the link layer functions that are:

- In-Sequence Delivery
- Flow Control
- Error Detection/Correction
- Keep Alive
- Local Data Retrieval
- Connection Control
- Protocol Error Detection and Recovery

Signaling Transfer Point (STP):
Packet switches that provide CCS message routing and transport. They are stored programmed switches that use information contained in the message in conjunction with information stored in memory to route the message to the appropriate destination signaling point.

2. Overview of the TALI Protocol

2.1 Traditional PSTN SS7 Networks

The traditional PSTN SS7 network consists of 3 types of devices connected via dedicated SS7 signaling links.

The 3 primary device types for PSTN networks are:

* SSP: Signaling Service Point. These nodes act as endpoints in the SS7 network, originating SS7 messages as users attempt to place phone calls. These nodes contain interfaces into the SS7 data network and the SS7 voice network.

* STP: Signaling Transfer Point. These nodes act primarily as switches, switching SS7 traffic from node to node throughout the network until it reaches another endpoint. An important feature of each STP is to provide SS7 network management functionality that allows messages to be delivered even when links and devices fail. STPs also sometimes provide database type services, such as Global Title Translations and Local Number Portability.

* SCP: Signaling Control Point. These nodes act as databases. These nodes contain stored data that is used to turn SS7 Queries into SS7 Replies.

There are 3 primary types of dedicated SS7 signaling links:

* 56Kbps SS7 (DS0, V35, OCU) links. These links implement the MTP-1 and MTP-2 protocols as defined in [1].

* DS1 High Speed Links. These links use the SAAL protocol to provide an alternative to 56Kbps SS7 links that is based on newer, faster technology. These links implement the SS7 protocol as defined in [8].

* E1 Links.

Figure 1 provides an overview of the traditional PSTN network. In this network, any of the links can be implemented via either 56 Kbps, DS1, or E1 links.

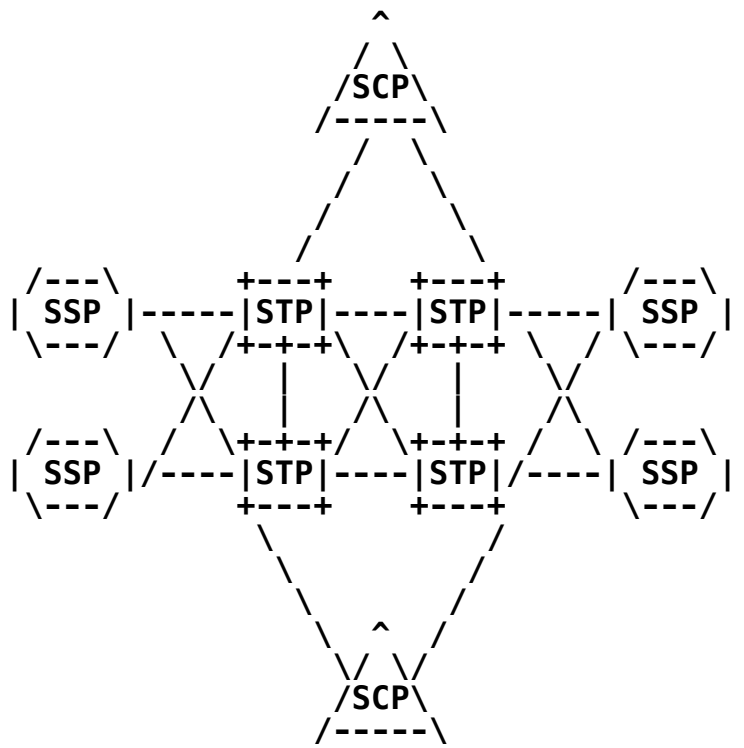

Figure 1: The Traditional PSTN Network

2.2 Converged SS7 Networks

In the converged SS7 network, SS7 devices will reside on both the traditional PSTN network (with dedicated 56 Kbps and DS1 links) and on the IP network (with Ethernet links based on IP protocol). The services of SSPs, STPs, and SCPs can be provided by new types of devices that reside on IP networks. The IP network is not intended to completely replace the PSTN, rather devices on the 2 types of networks must be able to communicate with one another and convert from 1 lower layer protocol to the other.

Signaling Gateways are new devices that may also function as an STP in the converged network. SGs provide interfaces to:

* devices on the SCN (traditional SSPs, STPs, and SCPs)

* other SGs

* new devices on the IP network

SGs also continue to perform STP functions such as SS7 network management and some database services (such as GTT and LNP).

New devices on the IP network include:

* Media Gateway Controllers. In addition to other functions, these devices control Media Gateways and perform call processing.

* Media Gateways. In addition to other functions, these devices control voice circuits that are used to carry telephone calls. MGs + MGCs combine to provide the functionality of traditional SSPs.

* IP based SCPs. The database services that are related to SS7 can be moved onto devices on the IP network.

Figure 2 provides an overview of the converged SS7 network.

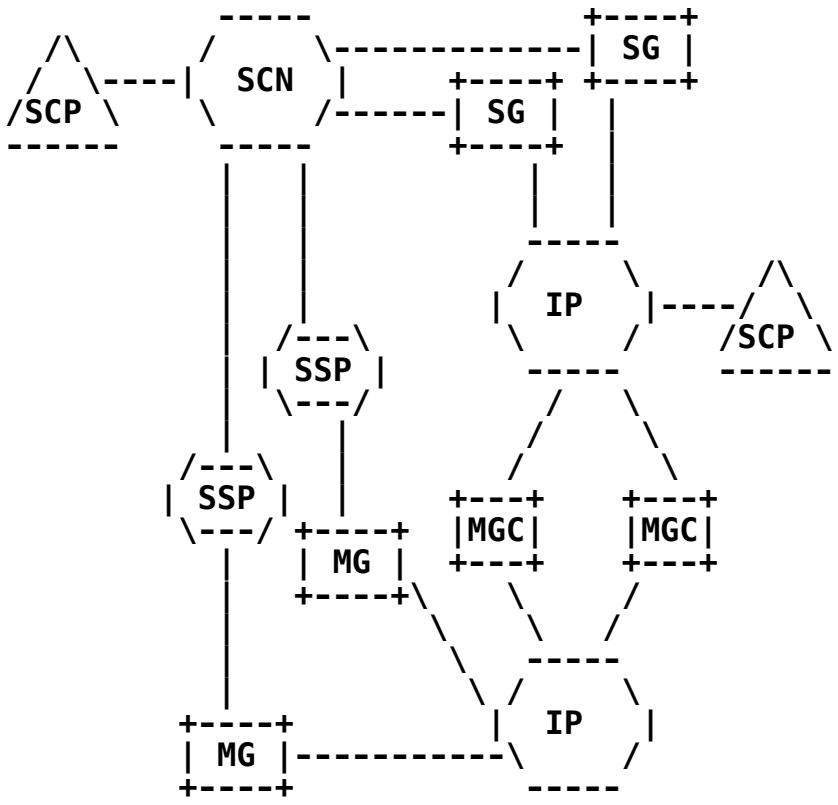

Figure 2: The Converged SS7 Network

In theory, the TALI protocol can be used between 2 nodes to carry SS7 traffic across TCP/IP. Some of the areas that TALI could be used include:

- For SG to SG communication across IP
- For SG to MGC communication across IP
- For SG to IP based SCP communication across IP
- For communication between multiple IP based SCPs
- For communication between multiple MGCs
- For communication between MGCs and MGs
- For other IP devices such as DNS, Policy Servers, etc.

In reality, the communication between MGCs, or between MGC and MG is probably better suited to using other protocols. With respect to the Signaling Gateway implementation, the TALI protocol is used to carry SS7 traffic:

- For SG to SG communication
- For SG to MGC communication
- For SG to IP based SCP communication

2.3 TALI Protocol Stack Overview

The Transport Adapter Layer Interface is the proposed interface that provides SCCP, ISUP, and MTP messaging encapsulation within a TCP/IP packet between two switching elements. In addition, TALI provides SCCP Management (SCMG), MTP Primitives, dynamic registration of circuits, and routing of call control messages based on circuit location.

The major purpose of the TALI protocol is to provide a bridge between the SS7 Signaling Network and applications that reside within an IP network. Figure 3 provides a simple illustration that highlights the protocol stacks used for transport of SS7 MSUs on both the SS7 side and the IP side of the SG.

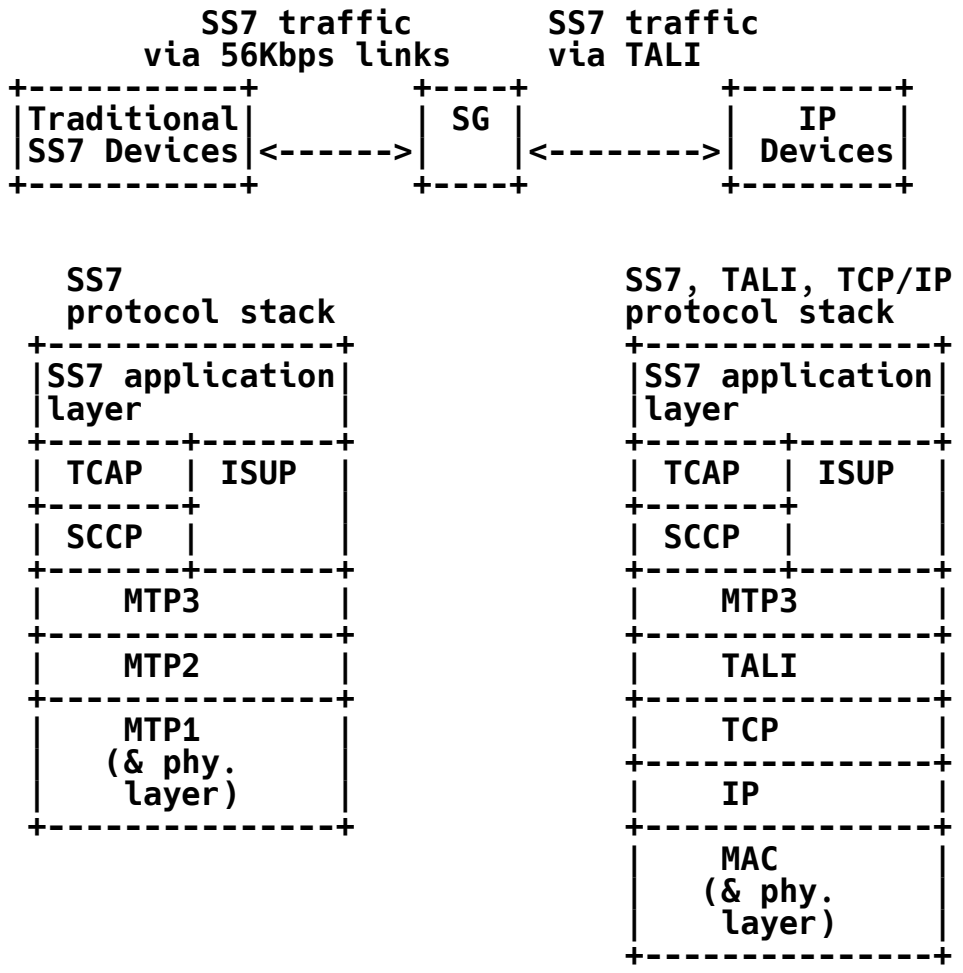

Figure 3: TALI Protocol to carry SS7 over TCP/IP

From Figure 3, several observations can be made:

* The TALI layer is used when transferring SS7 over IP.

* When SS7 traffic is carried over a IP network, the MTP2 and MTP1 layers of a traditional 56 Kbps link are replaced by the TALI, TCP, IP, and MAC layers

* The TALI layer sits on top of the TCP layer.

* The TALI layer sits below the various SS7 layers (MTP3, SCCP/TCAP, ISUP, and applications). The data from these SS7 layers is carried as the data portion of TALI service data packets.

Some of the facts concerning the TALI protocol which are important to understanding how TALI works that are not evident from Figure 3 include the following:

* Each TALI connection is provided over a single TCP socket.
    * The standard Berkeley sockets interface to the TCP is used by the TALI layer to provide connection oriented service from endpoint to peer endpoint.
    * TCP sockets are based on a Client/Server architecture; one end of the TALI connection must be defined as the 'server side', the other end is a 'client'.
    * The client/server roles are important only in bringing up the TCP connection between the 2 endpoint, once the connection is established both ends use the same Berkeley sockets calls (send, recv) to transfer data.
    * The TCP socket must be connected before the 2 TALI endpoints can begin communicating.
* TALI provides user control over each TALI connection that is defined. This control:
    * Allows the user to control when each TALI connection will be made
    * Allows the user to control when each TALI connection is allowed to carry SS7 traffic
    * Allows the user to control the graceful shutdown of each socket
* TALI provides Peer to Peer messages. These messages originate from the TALI layer of one endpoint of the connection and are terminated at the TALI layer of the other endpoint. Peer to Peer messages are used:
    * To provide test and watchdog maintenance messages
    * To control the ability of each socket to carry SS7 service messages
* TALI provides Service messages. These messages originate from the layer above the TALI layer of one endpoint of the connection and are transferred to and terminated at the layer above the TALI layer of the other endpoint.

- The service messages provide several different ways to encapsulate the SS7 messages (SCCP/TCAP, ISUP, and other MTP3 layer data) across the TCP/IP connection.
- As we will see later, different Service opcodes are used to communicate across the TALI socket exactly how each SS7 message has been encapsulated.

* A set of TALI timers is defined. These timers are used to correctly implement the TALI state machine.

2.3.1 An Alternate TALI Protocol Stack using the SAAL Layer

This section presents a different, slightly more complex, TALI protocol stack that can be used in place of the protocol stack in the previous section.

Figure 3 in the previous section provided a simple illustration that highlighted the basic TALI protocol stack that can be used to transport SS7 MSUs between 56 Kbps links on the SS7 side of an SG and the IP devices.

Figure 4 below illustrates an alternate TALI protocol stack that includes the SAAL layer as part of the data transferred across the TCP/IP connection.

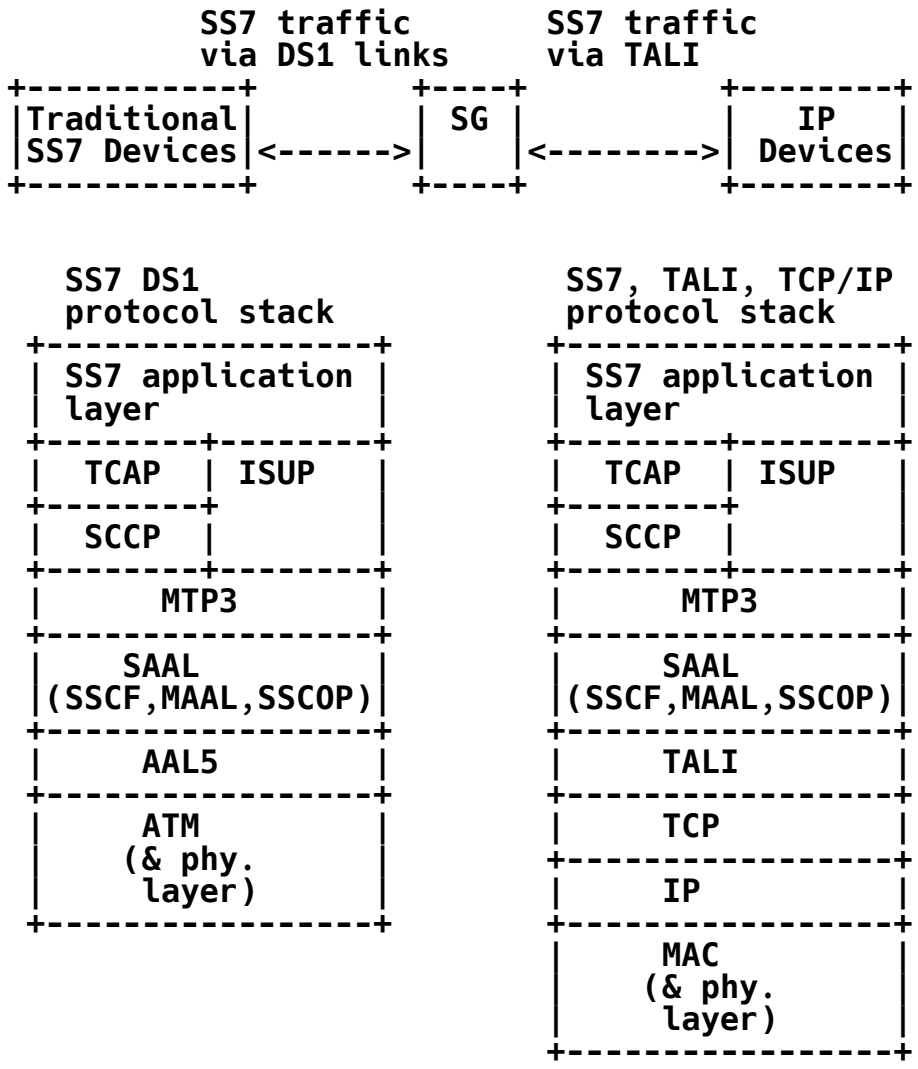

Figure 4: An Alternate TALI Protocol Stack with SAAL

The following bullets provide a discussion regarding the differences between these 2 protocol stacks, the reasons for having 2 protocol stacks, and the advantages of each:

* When the TALI protocol stack is implemented without the SAAL layer, as in Figure 3, the SEQUENCE NUMBER of the SS7 MSU is NOT part of the data transferred across the TCP/IP connection. In 56 Kbps SS7 links, the MTP2 header contains an 8 bit sequence number for each MSU. The sequence number is used to preserve message sequencing and to support complex SS7 procedures involving MSU retrieval during link changeover and changeback. As indicated in Figure 3, the MTP2 header is NOT part of the data transferred

across the TCP/IP connection. The TALI protocol stack without SAAL still guarantees correct sequencing of SS7 data (this sequencing is provided by sequence numbers in the TCP layer), however that protocol stack can not support SS7 changeover and changeback procedures.

* When the TALI protocol stack is implemented with the SAAL layer, as in Figure 4, the SEQUENCE NUMBER of the SS7 MSU IS part of the data transferred across TCP/IP. In SS7 DS1 links, the SSCOP trailer contains a 24 bit sequence number for each MSU. This 24 bit sequence number serves the same purposes as the 8 bit SS7 sequence number. As indicated in Figure 4, the SSCOP trailer IS part of the data transferred across the TCP/IP connection. The protocol stack in Figure 4 can support SS7 changeover and changeback procedures.

* Implementing the TALI protocol with SAAL therefore provides support for SS7 co/cb and data retrieval and can help to minimize MSU loss as SS7 links are deactivated. However, implementing SAAL is not a trivial matter. The SAAL layer consists of 3 sublayers (SSCF, SSCOP, and MAAL), one of which (SSCOP) is quite involved. It is envisioned that most SS7 to TCP/IP applications will NOT choose to implement SAAL.

2.3.2 An Alternate TALI Protocol Stack using SCTP

The TALI protocol is dependent on a reliable transport layer below it. At the initial design of TALI, TCP was the only reliable, proven transport layer. Simple Control Transport Protocol (SCTP) is currently being designed as a transport later specifically for signalling. Once SCTP is a proven and accepted transport protocol, SCTP can then be used in place of TCP as shown in Figures 3 and 4.

2.4 Inputs to the TALI Version 1.0 State Machine

Figure 5 illustrates the inputs that affect the TALI State Machine. Inputs to the state machine include:

* Management events (ie: requests from the human user of the TALI connection) to control the operation of a particular TALI session.

* TALI messages received from the Peer. These messages include peer to peer messages as well as service data messages.

* Events from the User of the TALI layer. The user is the layer above TALI in the protocol stack, either the SS7 or SAAL layer.

- Implementation Dependent Events. Each implementation must provide inputs into the TALI state machine such as:
  - Socket Events
  - TALI protocol violations. The TALI state machine must detect protocol violations and act accordingly.
  - Timer events.

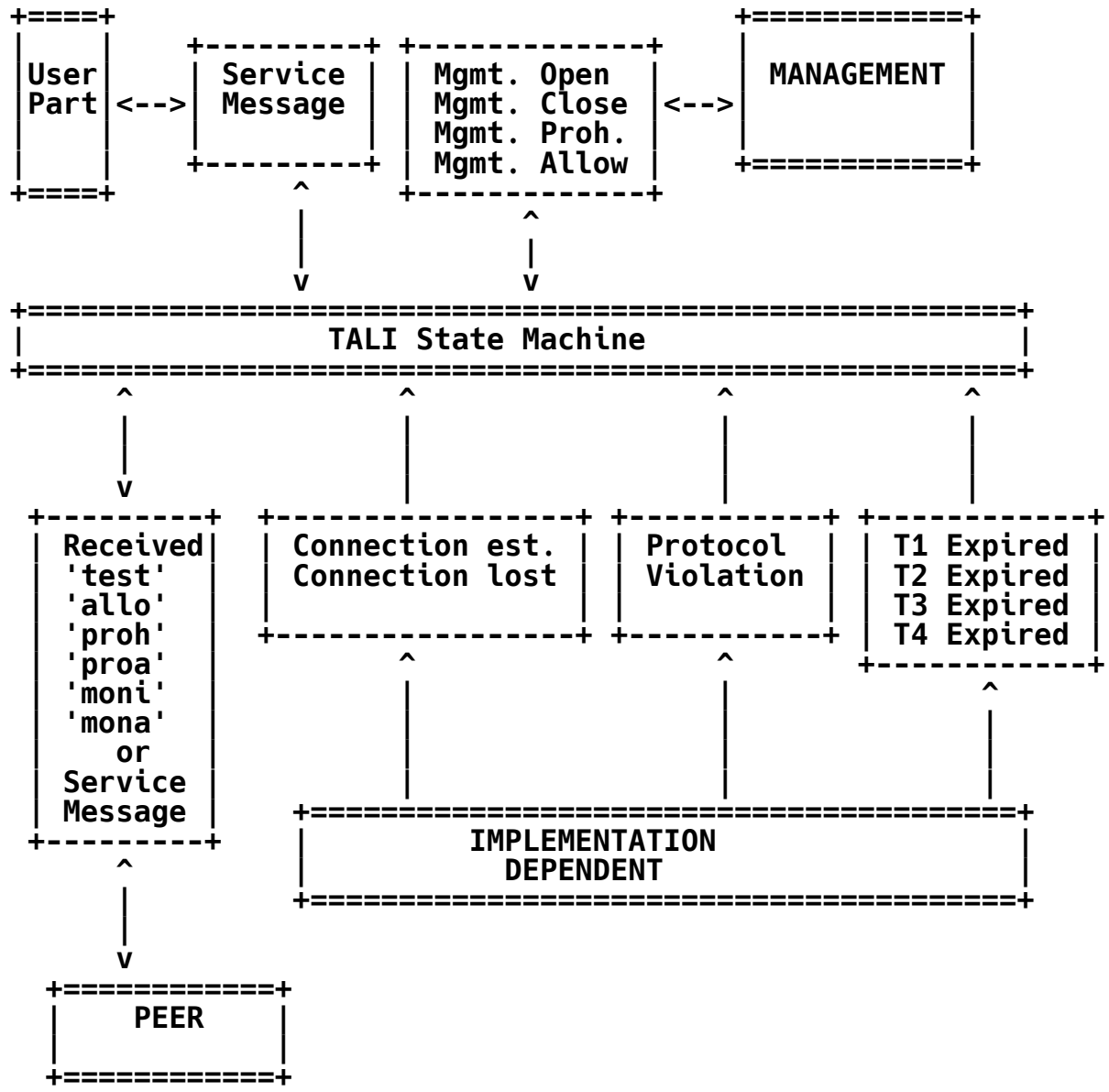

Figure 5: Overview of Inputs to the TALI 1.0 State Machine

3. TALI Version 1.0

This chapter provides the states, messages, message exchange rules and state machine that must be implemented to provide a TALI version 1.0 protocol layer.

3.1 Overview of the TALI Message Structure

Table 2 provides a summary of the messages and message structure used in TALI version 1.0.

OCTET	DESCRIPTION	SIZE	VALUE	TYPE
0..3	SYNC	4 Octets		4 byte ASCII
	TALI		'TALI'	
4..7	OPCODE	4 Octets		4 byte ASCII
	Test Service		'test'	
	Allow Service		'allo'	
	Prohibit Service		'proh'	
	Prohibit Service Ack		'proa'	
	Monitor Socket		'moni'	
	Monitor Socket Ack		'mona'	
	SCCP Service		'sccp'	
	ISUP Service over TALI		'isot'	
	MTP3 Service over TALI		'mtp3'	
	Service over SAAL		'saal'	
8..9	LENGTH (least significant byte first) non-0 if Service or Socket monitor message	2 Octets		integer
10..X	DATA PAYLOAD	variable		variable

Table 2: Message Structure for TALI 1.0

Table 3 indicates the valid values of the LENGTH field for each version 1.0 opcode. The LENGTH field is always an indication of the # of bytes contained in the DATA PAYLOAD portion of a general TALI message.

OPCODE	VALID LENGTH VALUES	COMMENTS
test	0 bytes	
allo	0 bytes	
proh	0 bytes	
proa	0 bytes	
moni	0-200 bytes	A maximum length is provided so that the maximum ethernet frame size is not exceeded.
mona	0-200 bytes	Mona reply length and content must match the original moni (with the exception of the opcode)
sccp	12-265 bytes	These are the valid sizes for the SCCP-ONLY portions of SCCP UDT MSUs
isot	8-273 bytes	The length is the number of octets in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte, the MTP3 routing label, the CIC code, and the ISUP Message Type field, and any other bytes that may exist as part of the SIF (Service Information Field)
mtp3	5-280 bytes	The length is the number of octets in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte and the MTP3 routing labeld, and any other bytes that may exist as part of the SIF (Service Information Field)
saal	11-280 bytes	The length is the number of octets in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte and all bytes in the SIF (Service Information Field) field. The MTP3 routing label is part of the SIF field. Seven (7)

		octets of SSCOP trailer is added to the message. The SSCOP trailer bytes are also included in the length.

Table 3: Valid Length Fields for Each Opcode in TALI 1.0

3.1.1 Types of TALI Fields

Several field types are used in the general TALI message structure.

Field Type	Implementation Notes for that Type
4 byte ASCII text	* 4 byte ASCII text strings are used to define the sync code and the opcode of the basic TALI message. * These fields are case sensitive, the coding for each sync and opcode literal needs to match the case specified in Table 2. * The standard ASCII conversion table is used to transform each character into a byte. * The order of the ASCII characters is important. The first character in the string must be the first character transmitted across the wire. * For example, if the string being encoded is 'abCD', the order of the bytes as they are transferred over the wire must be: 1st byte: 0x61 ('a') 3rd byte: 0x43 ('C') 2nd byte: 0x62 ('b') 4th byte: 0x44 ('D') * The software for each implementation should be written in a manner that accounts for the required byte order of transmission (ie: the Big Endian/ Little Endian characteristics of the processor need to be dealt with in the software.
Integer	* A 1, 2 or 4 byte field to be treated as an integer value. Integer fields should be transmitted Least Significant Byte first across the wire. * The software for each implementation should be written in a manner that accounts for the required byte order of transmission (ie: the Big Endian/ Little Endian characteristics of the processor need to be dealt with in the software.
Variable	* The definition of the message structure for this field is governed by other specifications. * For example, when transferring MTP3 service data

	via a 'mtp3' opcode, the DATA PAYLOAD begins with the SIO byte of the MTP3 routing label. The structure for the entire DATA PAYLOAD is governed by the MTP3 message structure defined in [1].
X byte ASCII text	* ASCII text fields of sizes other than 4 bytes should be supported according to the same rules presented for the 4 byte ASCII text fields. For instance, an 8 byte string such as 'ab01cd23' could be used, where the 'a' would be the first byte of the field transmitted out the wire.

Table 4: Implementation Notes for each Type of TALI field

3.2 Detailed TALI Message Structure

3.2.1 TALI Peer to Peer Messages

The following subsections provide more information regarding the TALI Peer to Peer messages that are implemented in version 1.0. The TALI peer to peer messages originate at the TALI layer of 1 end of the socket connection (the near end) and are terminated at the TALI layer of the far end of the connection.

3.2.1.1 Test Message (test)

The 'test' message is used by a TALI implementation to query the remote end of the TALI connection with respect to the willingness of the remote end to carry SS7 service data. This message asks the other end: are you ready to carry service data? This message is sent periodically by each TALI implementation based on a T1 timer interval. Upon receiving 'test', a TALI implementation must reply with either 'proh' or 'allo' to indicate the nodes willingness to carry SS7 service data over that TALI connection.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'test'
8..9	LENGTH	Length = 0

3.2.1.2 Allow Message (allo)

The 'allo' message is sent in reply to a 'test' query, or in response to some internal implementation event, to indicate that a TALI implementation IS willing to carry SS7 service data over the TALI session. This message informs the far end that SS7 traffic can be transmitted on the socket. 'allo' is one of the 2 possible replies to a 'test' message. Before SS7 traffic can be carried over a socket, both ends of the connection need to send 'allo' messages.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'allo'
8..9	LENGTH	Length = 0

3.2.1.3 Prohibit Message (proh)

The 'proh' message is sent in reply to a 'test' query, or in response to some internal implementation event, to indicate that a TALI implementation is NOT willing to carry SS7 service data over the TALI session. This message informs the far end that SS7 traffic can not be transmitted on the socket. 'proh' is one of the 2 possible replies to a 'test' message. As long as 1 end of the connection remains in the 'prohibited' state, SS7 traffic can not be carried over the socket.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'proh'
8..9	LENGTH	Length = 0

3.2.1.4 Prohibit Acknowledgement Message (proa)

The 'proa' message is sent by a TALI implementation each time a 'proh' is received from the far end. This message is sent to indicate to the far end that his 'prohibit' message was received correctly and will be acted on accordingly.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'proa'
8..9	LENGTH	Length = 0

3.2.1.5 Monitor Message (moni)

The 'moni' message provides a generic ECHO capability that can be used by each TALI implementation as that implementation sees fit. A TALI version 1.0 implementation does not have to originate a 'moni' message to be compliant with the 1.0 specification. The primary intent of this message is to provide a way for the TALI layer to test the round-trip message transfer time on a socket. A 'mona' message must be sent in reply to each received 'moni' message. The DATA portion of a 'moni' message is vendor implementation dependent. The DATA portion of each 'mona' reply must exactly match the DATA portion of the 'moni' that is replied to. Regardless of whether an implementation chooses to send 'moni' or not, 'mona' must be sent in response to each 'moni' in order to remain compliant with the TALI protocol.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'moni'
8..9	LENGTH	Length
10..X	DATA PAYLOAD	Vendor Dependent

3.2.1.6 Monitor Acknowledge Message (mona)

As mentioned above, the 'mona' must be sent in reply to each received 'moni'. The contents of the 'mona' DATA area must match the DATA area of the received 'moni' message.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'mona'
8..9	LENGTH	Length
10..X	DATA PAYLOAD	Vendor Dependent

3.2.2 Service Messages

The following subsections provide more information regarding the TALI Service messages that are implemented in version 1.0. TALI Service messages are used to carry SS7 MSUs across the IP network. The information in this section includes details with respect to how to encapsulate SS7 MSUs into TCP/IP frames using each of the TALI service opcodes. The TALI service messages originate at the layer above TALI, are transported across the IP network via a TALI service message, and are delivered to the layer above TALI at the far end of the TALI connection.

3.2.2.1 SCCP Service Message (sccp)

The 'sccp' opcode is used to deliver SS7 MSUs with a Service Indicator of 3 (SCCP) over a TALI connection. This opcode is only used on TALI protocol stacks that are implemented without SAAL. The MTP3 layer of the SS7 MSU is NOT part of the data transferred across TCP/IP for this opcode; the data portion of the TALI 'sccp' message begins with the first byte of the SCCP data area in the SS7 MSU (after the MTP3 routing label). The first byte in the SCCP data area is an SCCP message type field.

Several restrictions on the SCCP messages that this TALI opcode can carry exist. These restrictions are as follows:

* SCCP messages contain an SCCP message type field. The SCCP messages that are supported by TALI 1.0 implementations are limited to Class 0 and Class 1 SCCP messages with a message type field of either:
  * UDT
  * UDTS
  * XUDT
  * XUDTS

* SCCP messages must contain a Point Code in the 'calling party' area in order to be transferred across the TCP/IP connection as a 'sccp' message. An implementation may choose to modify the original SCCP MSU to add an appropriate calling party point code before transmission across TALI if desired.

* SCCP messages must contain a Point Code in the 'called party' area in order to be transferred across the TCP/IP connection as a 'sccp' message. An implementation may choose to modify the original SCCP MSU to add an appropriate called party point code before transmission across TALI if desired.

* The encoding of the SS7 SCCP MSUs, as they are transmitted across TALI via 'sccp', should remain compliant with the ANSI specifications (T1.112 and T1.114) that apply to the SCCP and TCAP portions of the message respectively.

NOTE 1: SCCP Subsystem Management for the IP based SCP's is supported via this 'sccp' opcode. SS7 SCCP Management messages are controlled by an SCMG SS7 process. SCMG sends the management messages via SCCP UNITDATA (UDT) messages. Therefore, the SCMG messages can be sent across the TALI connection.

NOTE 2: 'sccp' TALI messages will not include the MTP3 header and therefore will not retain the original DPC/OPC of the SS7 MSU. Each TALI implementation needs to consider if/how to provide this DPC/OPC information in the SCCP portion of the message. For example the DPC can be replicated to the point code in the SCCP Called Party Address area and the OPC can be replicated to the point code in the SCCP Calling Party Address area.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'sccp'
8..9	LENGTH	Length
10..X	SCCP Data	SCCP data starting at the first byte after the Layer 3 Routing Label (data does not include the SIO or Routing Label)

3.2.2.1.1 SCCP Encapsulation using TALI

When an SCCP MSU arrives at an SG from a 56 Kbps or DS1 link and is routed within the SG for transmission to an IP device, the SG performs the following processing on the SS7 MSU:

* discards the MTP Layer 2 information, CRC and flags

* places the DPC from MTP Layer 3 into the Called Party Address field of the SCCP layer; the Calling Party Address field is created if it does not exist and then filled

* places the OPC from MTP Layer 3 into the Calling Party Address field of the SCCP layer if there is no Calling Party Point Code

* places the modified SCCP and unchanged TCAP data in the service payload area of the TALI packet

* The SYNC field is set

* The OPCODE is set to 'sccp'

* The LENGTH is set to the number of octets in the SERVICE field

Once the fully formed 'sccp' TALI packet is created, it is handed to the TCP socket layer and transmitted. The transmission process will add TCP, IP and MAC header information.

Since the routing information from MTP Layer 3 is placed in the SCCP part of the outgoing message, no routing information needs to be saved by the SG.

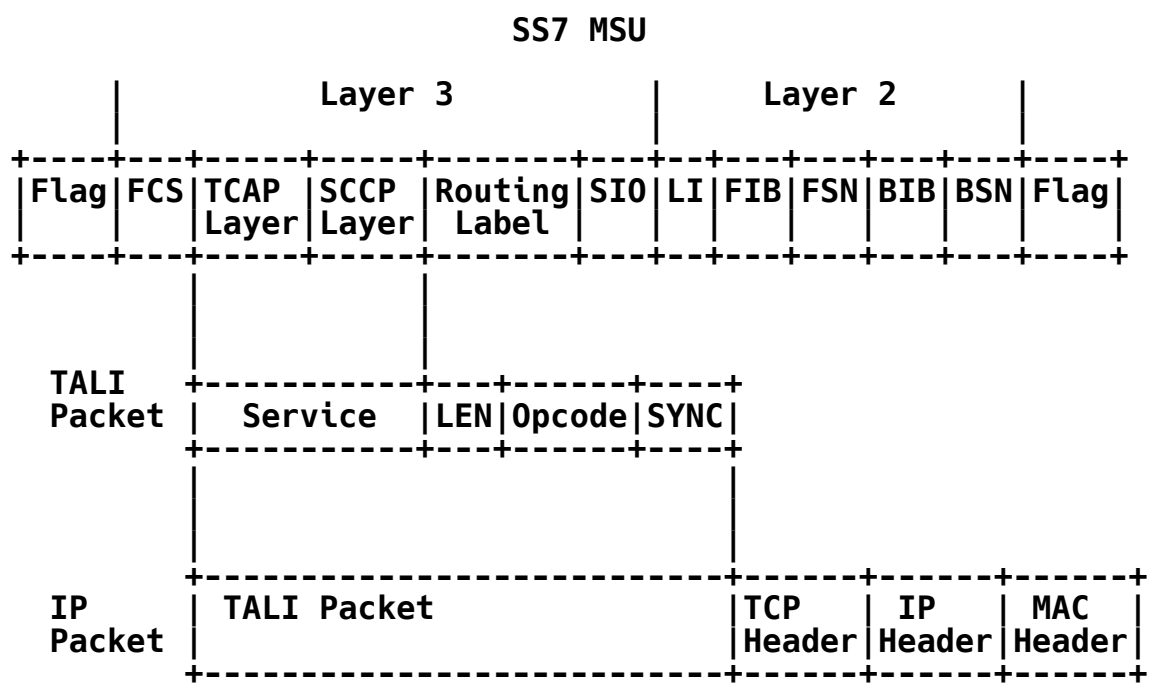

Figure 6: Encapsulation of SCCP MSUs using the TALI 'sccp' opcode

When an 'sccp' TALI packet is received on by an SG from an IP device, the SG performs the following processing on the 'sccp' packet:

* validates the TALI header

* Allocates space for a new SS7 message

* Regenerates the SIO with the Sub-Service Field set to National Network, priority of zero (0), Service Indicator set to SCCP

* extracts the SCCP/TCAP data from the SERVICE area and places it in the new SS7 message

* sets the DPC to the SCCP Called Party Point Code

* sets the OPC to the SCCP Calling Party Point Code

* randomly generates the SLS

Once the 'sccp' packet is transformed back into a normal SS7 MSU, the MSU is routed within the SG according to the normal SS7 routing procedures.

3.2.2.2 ISUP Service Message (isot)

The 'isot' opcode is used to deliver SS7 MSUs with a Service Indicator of 5 (ISUP) over a TALI connection. This opcode is only used on TALI protocol stacks that are implemented without SAAL. The MTP3 layer of the SS7 MSU IS part of the data transferred across TCP/IP for this opcode; the data portion of the TALI 'isot' message begins with the SIO byte of the MTP3 header in the SS7 MSU.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'isot'
8..9	LENGTH	Length
10..X	ISUP Data	Raw ISUP data starting at the Layer 3 SIO field.

3.2.2.2.1 ISUP Encapsulation using TALI

When an ISUP MSU arrives at an SG from a 56 Kbps or DS1 link and is routed within the SG to a IP device, the SG performs the following processing on the SS7 MSU:

* discards the MTP Layer 2 information, CRC and flags
* places MTP Layer 3 into the SERVICE payload area of the TALI packet
* The SYNC field is set
* The OPCODE is set to 'isot'
* The LENGTH is set to the number of octets in the SERVICE field

Once the fully formed 'isot' TALI packet is created, it is handed to the TCP socket layer and transmitted. The transmission process will add TCP, IP and MAC header information.

Since the routing information is placed in the TALI Packet, no routing information needs to be saved by the SG.

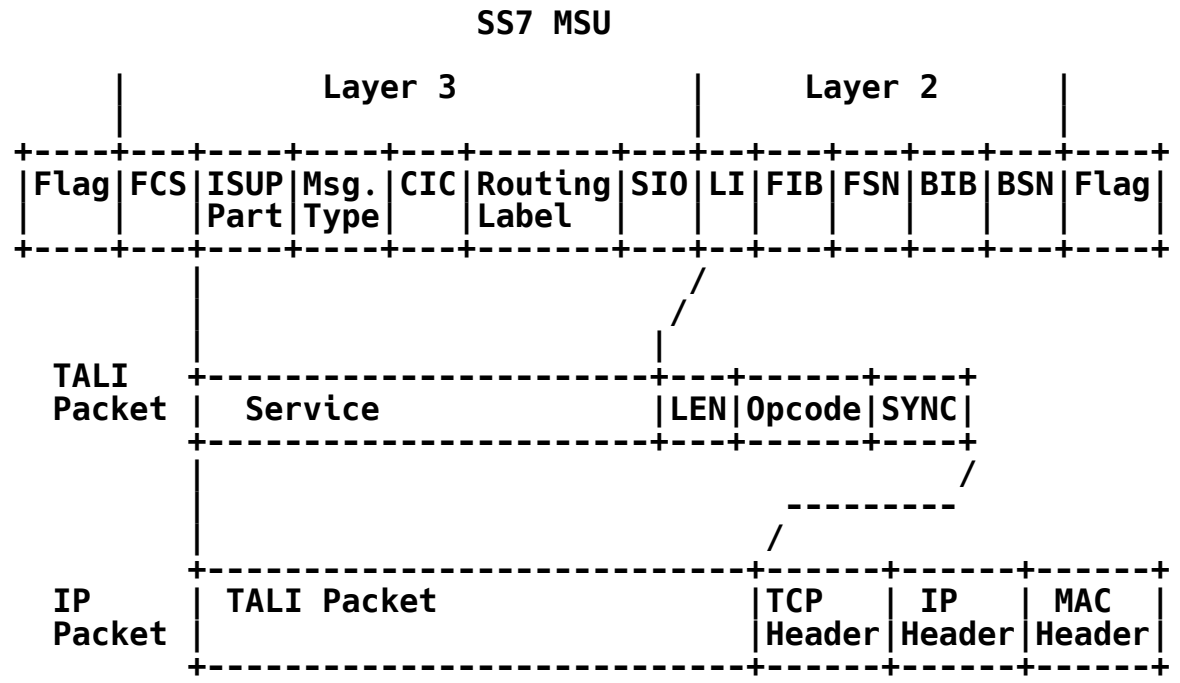

Figure 7: Encapsulation of ISUP MSUs using the TALI 'isot' opcode

When an 'isot' TALI packet is received on an SG from an IP device, the SG performs the following processing on the 'isot' packet:

* validates the TALI header
* Allocates space for a new SS7 message
* extracts the MTP Layer 3 data from the SERVICE area and places it in the new SS7 message

Once the 'isot' packet is transformed back into a normal SS7 MSU, the MSU is routed within the SG according to the normal SS7 routing procedures.

3.2.2.3 MTP3 Service Message (mtp3)

The 'mtp3' opcode is used to deliver SS7 MSUs with a Service Indicator of 0-2, 4, 6-15 (non-SCCP, non-ISUP) over a TALI connection. This opcode is only used on TALI protocol stacks that are implemented without SAAL. The MTP3 layer of the SS7 MSU IS part of the data transferred across TCP/IP for this opcode; the data portion of the TALI 'mtp3' message begins with the SIO byte of the MTP3 header in the SS7 MSU.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'mtp3'
8..9	LENGTH	Length
10..X	Layer 3 MSU Data	Raw MSU data starting at the Layer 3 SIO field.

3.2.2.3.1 MTP3 Encapsulation using TALI

When an SS7 MSU with SI=0-2,4,6-15 arrives at an SG from a 56 Kbps or DS1 link and is routed within the SG to an IP device, the SG performs the following processing on the SS7 MSU:

* discards the MTP Layer 2 information, CRC and flags

* places MTP Layer 3 into the SERVICE payload area of TALI packet

* The SYNC field is set

* The OPCODE is set to 'mtp3'

* The LENGTH is set to the number of octets in the SERVICE field

Once the fully formed 'mtp3' TALI packet is created, it is handed to the TCP socket layer and transmitted. The transmission process will add TCP, IP and MAC header information.

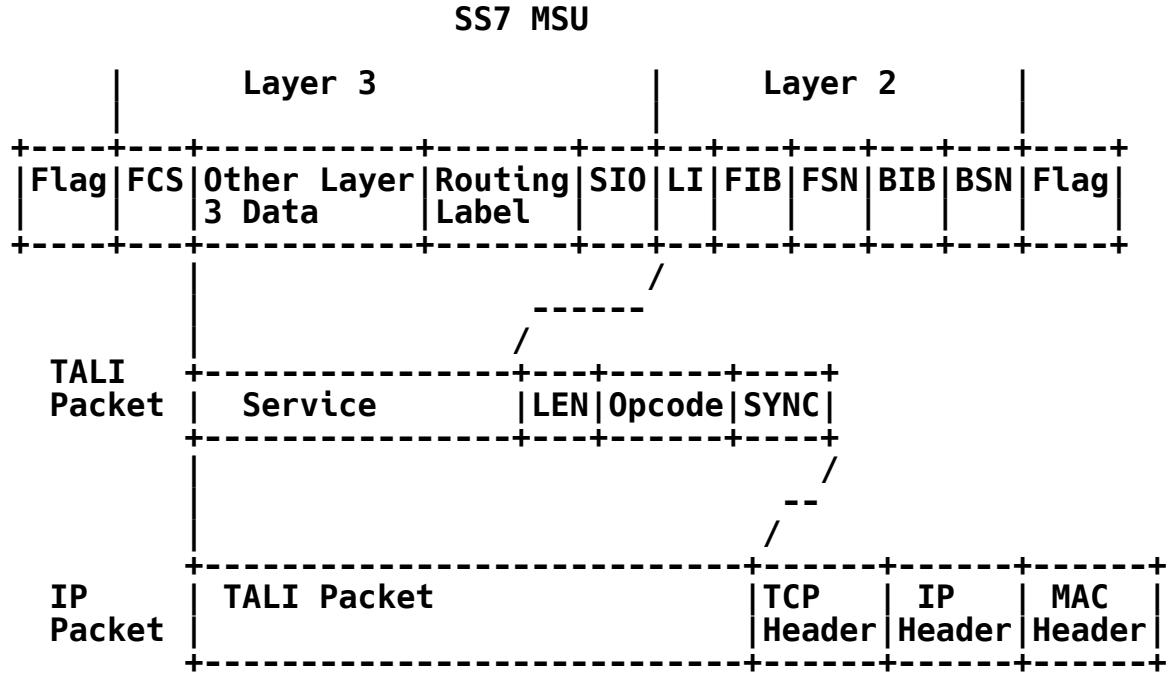

Figure 8: Encapsulation of SS7 MSUs with SI!=3,5,13 using 'mtp3'

When an 'mtp3' TALI packet is received by an SG from an IP device, the SG performs the following processing on the 'mtp3' packet:

* validates the TALI header

* Allocates space for a new SS7 message

* extracts the MTP Layer 3 data from the SERVICE area and places it in the new SS7 message

Once the 'mtp3' packet is transformed back into a normal SS7 MSU, the MSU is routed within the SG according to the normal SS7 routing procedures.

3.2.2.4 SAAL Service Message (saal)

The 'saal' opcode is used to deliver SS7 MSUs with any Service Indicator over a TALI connection. This opcode is only used on TALI protocol stacks that are implemented with SAAL. The 'saal' opcode is also used to transmit SAAL peer to peer packets (SSCF peer to peer packets and SSCOP peer to peer packets other than SS7 service data) over a TALI connection.

When used to transfer SS7 MSUs, the MTP3 layer of the SS7 MSU IS part of the data transferred across TCP/IP for this opcode; the data portion of the TALI 'saal' message begins with the SIO byte of the MTP3 header in the SS7 MSU and ends with the last byte of the SSCOP trailer.

When used to transfer SSCF/SSCOP peer to peer messages the data portion of the TALI 'saal' message includes the entire SSCOP PDU.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'saal'
8..9	LENGTH	Length
10..X	Layer 3 Data	Raw MSU data starting at the Layer 3 SIO field.
(X+1) ..Y	SSCOP Trailer	Zero (0) to three (3) octets of padding plus 4 octets for the trailer data. The total length of the Layer 3 Data and the SSCOP trailer must be a multiple of 4.

or

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'saal'
8..9	LENGTH	Length
10..X	SAAL Peer to Peer message	Raw SSCF/SSCOP peer to peer packets are also transferred over the TALI connection using this 'saal' opcode.

3.2.2.4.1 MTP3 and SAAL Peer to Peer Encapsulation using TALI

When an SS7 MSU (with any SI) arrives at an SG from a 56 Kbps or DS1 link and is routed within the SG for transmission to an IP device, the SG performs the following processing on the SS7 MSU:

* discards the MTP Layer 2 information, CRC and flags

* the MSU is passed from an MTP3 processing software layer to the SSCF and SSCOP layers (the SAAL layers). These layers convert the SS7 MSU into an SSCOP PDU. Part of this conversion includes adding an SSCOP trailer.

* the SSCOP PDU (whether it is a peer to peer SAAL message or SS7 MSU in an SSCOP PDU) is copied into the SERVICE payload area of the TALI packet

* The SYNC field is set

* The OPCODE is set to 'saal'

* The LENGTH is set to the number of octets in the SERVICE field

Once the fully formed 'saal' TALI packet is created, it is handed to the TCP socket layer and transmitted. The transmission process will add TCP, IP and MAC header information.

Since the routing information is placed in the TALI Packet, no routing information needs to be saved by the SG.

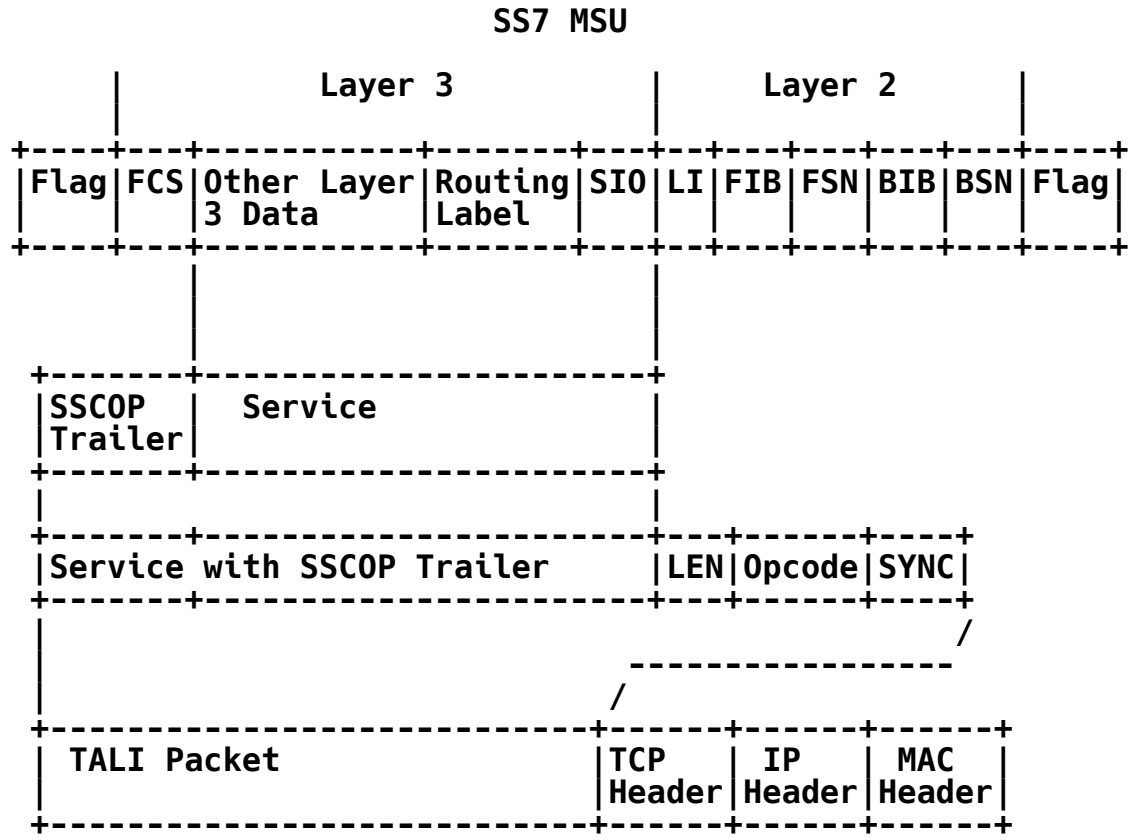

Figure 9: Encapsulation of SAAL PDUs using the TALI 'saal' opcode

When an 'saal' TALI packet is received at the SG from an IP device, the SG performs the following processing on the 'saal' packet:

* validates the TALI header

* Allocates space for a new SSCOP PDU message

* extracts the SSCOP PDU data from the SERVICE area and places it in the new SSCOP PDU message

Once the 'saal' packet is transformed back into a normal DS1 SSCOP PDU, the SSCOP PDU is passed to the SAAL layer for receive processing. If the SSCOP PDU is a peer to peer pdu, it is processed completely in the appropriate SAAL layer. If the SSCOP PDU is an SS7 MSU, the MSU is transformed back to a normal SS7 MSU and is routed within the SG according to the normal SS7 routing procedures.

3.3 TALI Timers

Version 1.0 of the TALI specification defined 4 TALI timers that are used as part of the TALI state machine. These timers are generically named 'T1' through 'T4'. Brief descriptions of each timer are provided in the following subsections. Timer expiration events for each of the T1-T4 timers appear as inputs to the TALI state machine. For exact processing of each timer (when to start/stop, how to process timer expirations), refer to the TALI state machine.

Both ends of the TALI connection have there own T1-T4 timers. The T1-T4 timer values can be set on each end of the connection independent of the settings on the far end. For each timer, a default value and range is recommended in the following sections.

3.3.1 T1 Timer

The T1 timer represents the time interval between the origination of a 'test' message at each TALI implementation. Each time T1 expires, the TALI implementation should send a 'test'.

3.3.2 T2 Timer

The T2 timer represents the amount of time that the Peer has to return an 'allo' or a 'proh' in response to a 'test'. If the far end fails to reply with 'allo' or 'proh' before T2 expires, the sender of the 'test' treats the T2 expiration as a protocol violation. Note that T2 must be < T1 in order for these timers to work as designed.

3.3.3 T3 Timer

The T3 timer controls how long the near end should continue to process Service Data that is received from the far end after a Management Prohibit Traffic Event has occurred (at the near end). This timer is used when a transition from NEA-FEA (both ends allowed to send service data) to NEP-FEA (only far end willing to send service data) occurs. On that transition, it is reasonable to expect that the far end needs some amount of time to adjust its TALI state machine and divert service data traffic away from this socket. The T3 timer controls the amount of time the far end has to divert traffic.

3.3.4 T4 Timer

The T4 timer represents the time interval between the origination of a 'moni' message at each TALI implementation. Each time T4 expires, the TALI implementation should send a 'moni'.

3.3.5 Recommended Defaults and Ranges for the TALI Timers

The following table provides the recommended default and configurable range for each TALI timer.

Name	Min	Max	Default	Description
T1	100ms	60sec	4 sec	Send test PDU timer
T2	100ms	60sec	3 sec	Response timer for an allo or proh response to test message.
T3	100ms	60sec	5 sec	Timer controls how long to process rcvd serv data after an NE transition from NEA to NEP. System is waiting for a proa response to the first proh send when NE transitions from NEA to NEP.
T4	100ms	60sec	10 sec	Send moni PDU timer

Table 5: Timers

NOTE: The value of T1 must be at least one (1) millisecond greater than T2. This is to prevent the system from a lockup in the T1 expired condition. If T1 is equal or less than T2, it will expire and restart T2 and not enforce responses to the test message.

Enforcement of minimum and maximum timer values is implementation dependent.

3.4 TALI User Events

Each TALI implementation must provide several user event controls over the behavior of the TALI state machine for each TALI connection. The user interface to provide these capabilities is implementation specific.

3.4.1 Management Open Socket Event

The 'mgmt open socket' event, together with the 'mgmt close socket' event, allows the user to control when each defined TALI connection will form a TCP socket connection. When 'open socket' for a particular TALI connection occurs, the TALI connection should begin trying to form a TCP socket connection to the peer.

The steps that are taken to connect are dependent on the client/server role of that end of the TALI connection. The exact steps to perform these tasks are implementation dependent and may differ based on the TCP stack being used.

In general, TALI clients form socket connections by using the BSD sockets calls:

```
Socket()
Bind()
Connect()
```

In general, TALI servers form socket connections by using the BSD sockets calls:

```
Socket()
Bind()
Listen()
Accept()
```

3.4.2 Management Close Socket Event

The 'mgmt close socket' event can be issued by the user when it is desired that the TCP socket for a TALI socket, be closed immediately, or discontinue its attempts to connect to the peer. After acting on 'close socket', the TALI connection will not be established until 'mgmt open socket' is issued.

3.4.3 Management Allow Traffic Event

The 'mgmt allow traffic' event, together with the 'mgmt prohibit traffic' event, allows the user to control when each defined TALI connection will be willing to carry SS7 service data over that particular TALI connection. When 'mgmt allow traffic' is issued, the TALI implementation becomes willing to carry service data. The TALI state for the near end should transition to NEA (near end allowed) if the connection is already established.

3.4.4 Management Prohibit Traffic Event

The 'mgmt prohibit traffic' event is the opposite of 'allow traffic'. When 'mgmt prohibit traffic' is issued, the TALI implementation becomes un-willing to carry SS7 service data over that particular TALI connection. The TALI state for the near end should transition to NEP (near end prohibited) if the connection is already established.

3.5 Other Implementation Dependent TALI Events

In addition to timers, each TALI implementation needs to be able to detect, and react accordingly, for the following events:

* Connection Established. When the TCP socket connection is initially established the TALI state machine must be notified.

* Connection Lost. When the TCP socket connection is lost, due to socket errors during reads/writes, the TALI state machine must be notified.

* Protocol Violations. Any violation of the TALI protocol as discussed in 3.7.1.3.

3.6 TALI States

The TALI version 1.0 specification is based on a state machine that considers 6 TALI states. Each end of the TALI connection maintains its own TALI state.

Name	Description
OOS	The TALI connection is out of service. This usually corresponds to a user event to 'close' the socket, or a user event to 'deactivate the SS7 link'.
Connecting	The TALI layer is attempting to establish a TCP socket connection to the peer. Servers are 'accepting', clients are 'connecting'.
NEP-FEP	The TCP socket connection is established. Neither side of the connection is ready to use the socket for service PDUs.
NEP-FEA	The TCP socket connection is established. The NE is not ready to use the socket for service PDUs. The FE is ready to use the socket for service PDUs.
NEA-FEP	The TCP socket connection is established. The NE is ready to use the socket for service PDUs. The FE is not ready to use the socket for service PDUs.
NEA-FEA	The TCP socket connection is established. Both sides are ready to use the socket for service PDUs. This is the only state where normal bi-directional SS7 data transfer occurs.

Table 6: TALI States

3.7 TALI Version 1.0 State Machine

This section provides the state machine that must be followed by each TALI implementation in order to be compliant with this specification.

3.7.1 State Machine Concepts

Before presenting the actual state machine, several concepts are discussed.

3.7.1.1 General Protocol Rules

1. Neither side can send service data unless both sides are allowed.

2. Each side initializes to the prohibited state for both near end and far end.

3. State changes between the NEx-FEx states are signaled with either an 'allo' or 'proh'.

4. Each side can poll the far end's state with a 'test'. Upon sending 'test', T1 and T2 should always be restarted.

5. Each side polls the far end with a 'test' every T1 expiration.

6. The reply to a 'test' is based on the state of the near end only.

7. The reply to a 'test' is either 'allo' or 'proh'.

8. A far end signals the last service PDU has been transmitted with either a 'proh' or a 'proa'.

9. Upon receiving a 'proh', the receiver must always reply with 'proa'.

10. The NE cannot gracefully close a socket unless a 'proh' is sent and 'proa' is received.

11. On the transition from NEA to NEP, after sending a 'proh', the near end must continue to process received service data until a 'proa' is received or until a T3 timer expires.

3.7.1.2 Graceful Shutdown of a Socket

The state table treats a management request to close the socket as a 'hard' shutdown. That is, it will close the socket immediately regardless of the current state. Therefore, the correct steps to ensure a graceful shutdown of a socket (from the NEA_FEP or NEA_FEA states) is:

1. Management issues a Management Prohibit Traffic Event on the socket.

2. Management will wait for T3 to expire.

3. Management can then issue a Close Socket Event on the socket.

3.7.1.3 TALI Protocol Violations

Each TALI implementation must detect when violations of the TALI protocol have occurred and react accordingly. Protocol violations include:

* Invalid sync code in a received message

* Invalid opcode in a received message

* Invalid length field in a received message

* Not receiving an 'allo' or 'proh', in response to the origination of a 'test' , before the T2 timer expires

* Receiving Service Messages on a prohibited socket.

* TCP Socket errors - Connection Lost

In the state machine that follows, State/Event combinations that should be treated as protocol violations are indicated via a 'PV' in the state/event cell. All of the 'PV' events are then processed as per the 'Protocol Violation' row in the table.

3.7.2 The State Machine

Internal Data required for State Machine:

boolean sock_allowed. This flag indicates whether the NE is allowed to carry Service Messages.

Initial Conditions:
sock_allowed = FALSE
state = OOS
no timers running

State / Event	OOS	Connecting	NEP-FEP	NEP-FEA	NEA-FEP	NEA-FEA
T1 Exp.			Send test Start T1 Start T2	Send test Start T1 Start T2	Send test Start T1 Start T2	Send test Start T1 Start T2
T2 Exp.			PV	PV	PV	PV
T3 Exp.			PV	PV		
T4 Exp.			Send moni Start T4	Send moni Start T4	Send moni Start T4	Send moni Start T4
Rcv test			Send proh	Send proh	Send allo	Send allo
Rcv allo			Stop T2 NEP-FEA	Stop T2	Stop T2 NEA-FEA	Stop T2

Rcv proh			Stop T2 Send proa	Stop T2 Send proa NEP-FEP	Stop T2 Send proa	Stop T2 Flush or reroute Send proa NEA-FEP
Rcv proa			Stop T3	Stop T3		
Rcv moni			Convert to mona Send mona	Convert to mona Send mona	Convert to mona Send mona	Convert to mona Send mona
Rcv mona			Implemen- tation dependent	Implemen- tation dependent	Implemen- tation dependent	Implemen- tation dependent
Rcv Service			PV	If T3 run Process Else PV	PV	Process
Connect. Estab.		Start T1 Start T2 Start T4 (if non-0) if sock_ allowed = TRUE send allo send test NEA-FEP else send proh send test NEP-FEP				
Connect. Lost			PV	PV	PV	PV
Protocol Violat.			Stop all timers Close the socket Connect- ing	Stop all timers Close the socket Connect- ing	Stop all timers Close the socket Connect- ing	Stop all timers Close the socket Connect- ing

Mgmt. Open Socket	Open socket Conne- cting					
Mgmt. Close Socket		Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS
Mgmt. Prohibit Socket	sock_ allow- ed = FALSE	sock_allo- wed=FALSE	sock_all- owed= FALSE	sock_all- owed= FALSE	sock_all- owed= FALSE send proh start t3 NEP-FEP	sock_all- owed= FALSE send proh start t3 NEP-FEA
Mgmt. Allow Traffic	sock_ allow- ed = TRUE	sock_allo- wed=TRUE	sock_all- owed= TRUE send allo NEA-FEP	sock_all- owed= FALSE send allo NEA-FEA	sock_all- owed= TRUE	sock_all- owed= TRUE
User Part Msgs.	reject data	reject data	reject data	reject data	reject data	send data

Table 7: TALI 1.0 State Machine

3.8 TALI 1.0 Implementation Notes

Several aspects of the expected TALI 1.0 implementation have not been specifically addressed in the state machine or previous text (or else they were presented but will be reiterated here). These implementation notes in some cases have to do with the expected behavior of the software layer above the TALI layer.

3.8.1 Failure on a TCP/IP Socket

* The failure to read or write from a TCP socket shall be detected and generate a connection lost event.

3.8.2 Congestion on a TCP/IP Socket

* Message streams can be monitored for congestion via implementation dependent methods.

* One possible definition of congestion for the previous requirement might be when a TCP socket is blocked.

3.9 TALI 1.0 Limitations

Several limitations with the TALI 1.0 specification and implementation are identified:

* For SCCP traffic, only UDT and XUDT Class 0 and Class 1 traffic should be managed by this protocol.

* When the MTP3 Routing Label is not part of the data transmitted across the wire, priority zero (0) traffic is used for all traffic when the SIO is regenerated.

4. TALI Version 2.0

Version 2.0 of the TALI specification provides several additions to the Version 1.0 specification. The 2.0 additions are provided by introducing three new TALI opcodes. The basic functionality and most of the details of the TALI 1.0 implementation are NOT changed by the 2.0 additions.

The table below provides a summary of the messages and message structure used in TALI version 2.0.

OCTET	DESCRIPTION	SIZE	VALUE	TYPE
0..3	SYNC	4 Octets		4 byte ASCII
	TALI		'TALI'	
4..7	OPCODE	4 Octets		4 byte ASCII
	Test Service		'test'	
	Allow Service		'allo'	
	Prohibit Service		'proh'	
	Prohibit Service Ack		'proa'	
	Monitor Socket		'moni'	
	Monitor Socket Ack		'mona'	
	SCCP Service		'sccp'	
	ISUP Service o/TALI		'isot'	
	MTP3 Service o/TALI		'mtp3'	
	Service o/SAAL		'saal'	
	Management Message		'mgmt'	
	Extended Service Msg		'xsrv'	
	Special Message		'spcl'	
8..9	LENGTH (least significant byte first) non-0 if Service or Socket monitor msg	2 Octets		integer
10..X	DATA PAYLOAD	variable		variable

Due to the minimal amount of change from 1.0, this chapter will only provide:

* Detailed information regarding how a TALI implementation can identify itself as a 2.0 vs. a 1.0 implementation

* Detailed information regarding how to provide backward compatibility for a connection to a far end that is only TALI 1.0 capable

* Detailed information regarding the new 2.0 opcodes

* Detailed information regarding any other changes to the information presented in previous sections that need to be implemented in order to be 2.0 compatible.

Therefore, readers of this chapter should read this from the point of view of modifying an existing TALI 1.0 implementation to support the new 2.0 features.

4.1 Overview of TALI Version 2.0 Features

A small number of changes to a 1.0 TALI implementation are required to support 2.0. Figure 10 illustrates the inputs that affect the 2.0 TALI State Machine. The reader may notice that the only differences from the inputs for 1.0 are as follows:

Three new TALI opcodes can be sent/received between a TALI node and its peer. The new opcodes are:

* 'mgmt'
* 'xsrv'
* 'spcl'

Three new User Part capabilities need to be supported by the layer of code above the TALI layer in each implementation. The user part needs to provide support for 'mgmt', 'xsrv', and 'spcl' data.

More information about the 3 new opcodes is provided in individual sections in this chapter. However, a brief description of the purpose of each of these opcodes is as follows:

* 'mgmt' - This opcode is intended to allow MANAGEMENT data, or data that will manage the operation of the device, to pass between the TALI endpoints. Examples of this management data include:

  * configuration data, such as which SS7 traffic streams a peer would like to receive over a specific socket

  * SS7 Network Management data, such as information regarding point code (un)availability and congestion.

  * Enabling/disabling various socket options, such as options regarding which messages are supported, or how to format data.

* 'xsrv' - Extended Service Opcodes. It is envisioned that the TALI protocol could be extended to carry other types of traffic that are not covered by the 1.0 service data opcodes ('sccp', 'isot', 'mtp3', or 'saal'). By defining a new 'xsrv' service opcode, the TALI protocol is opened up to the possibility of being used for other types of data transport.

* 'spcl' - Special services. It is envisioned that vendors may want to build special services into their TALI implementations that are only activated when the implementation is connected to other equipment implementing the same special services. This opcode is intended to provide a general means to discover more information regarding who the TALI session is connected to, and a means to enable special features based on the vendor/implementation on the far end.

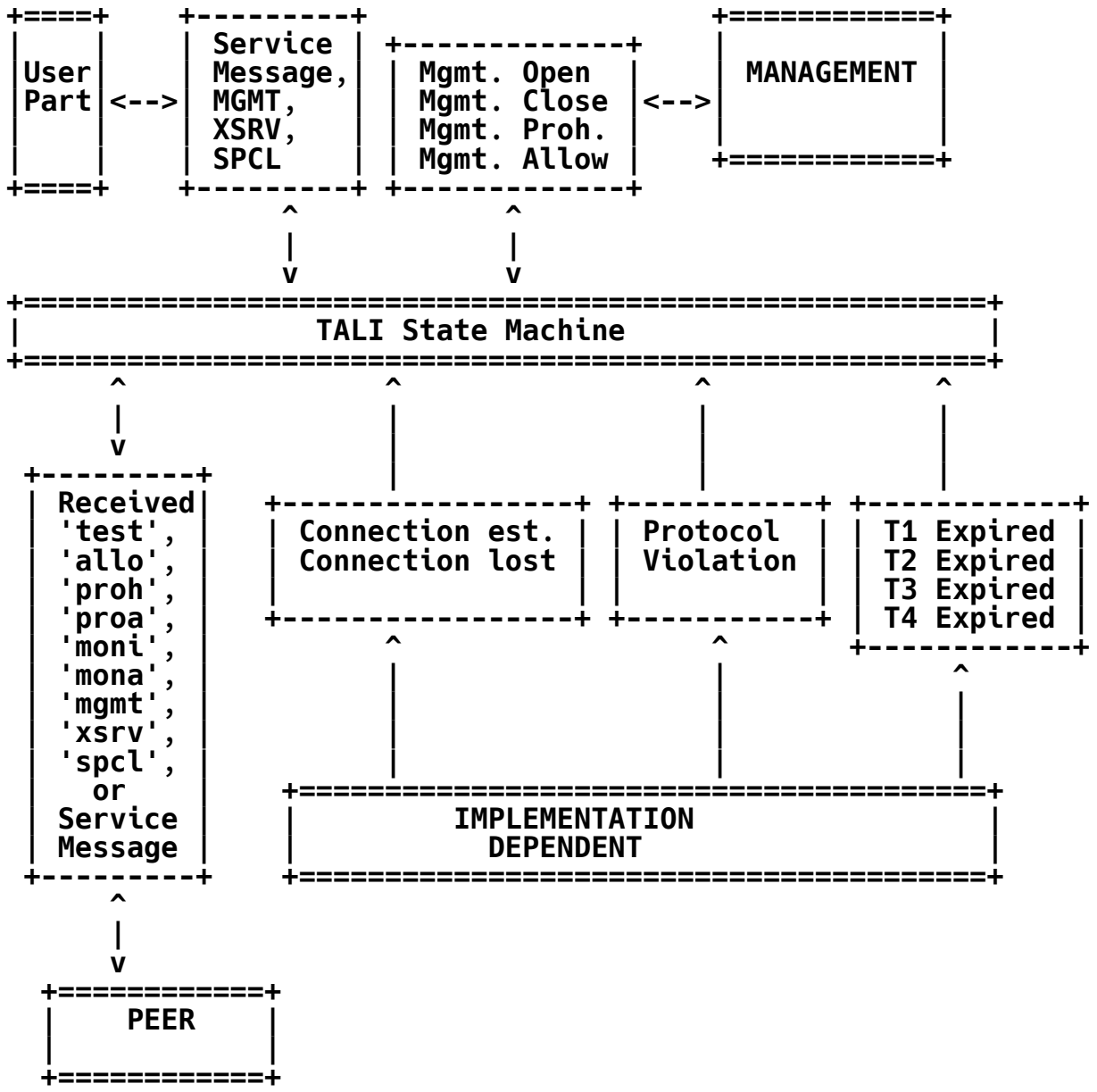

Figure 10: Overview of Inputs to the TALI 2.0 State Machine

4.2 TALI Version Identification

The TALI 1.0 specification did not provide a simple means to perform TALI version identification. However, the general purpose 'moni' message from 1.0 can be used to solve this problem in 2.0.

Recall from 1.0 that the 'moni' message was very loosely defined in the 1.0 spec:

* The primary purpose of the 'moni' message was to provide a general purpose ECHO capability. It was envisioned that an important task that the ECHO capability could provide would be to measure Round Trip TALI/TALI processing time.

* The data portion of the 'moni' message could be from 0-200 bytes long. The use of the data area was completely implementation specific.

* There were no requirements that an implementation ever send a 'moni'.

* If an implementation did send 'moni', it should use the T4 timer to control the frequency of the outgoing 'moni'.

* The receiver of the 'moni' should not make any assumptions as to the data portion of the 'moni'. The receiver should simply convert the 'moni' into a 'mona' and return the message with the same data portion.

TALI 2.0 implementations should use the 'moni' message to provide version identification as per the following bullets:

* The primary purpose of the 'moni' message is now twofold:

* To provide version identification

    * To continue to provide a general purpose ECHO capability that can be used to measure Round Trip time or perform other implementation specific tasks.

* The data portion of the 'moni' message is now divided into 2 portions

    * A portion dedicated to version identification, 12 bytes long, with a specific format that must be followed

    * Followed by a free format section that can be used in a completely implementation specific manner.

* The overall length of the data portion for a 'moni' should still not exceed 200 bytes. This is required to maintain backward compatibility with 1.0 implementations that may check for a maximum length of 200 bytes on the 'moni' opcode.

* If a TALI implementation wants to identify itself as a version 2.0 node, it must send a 'moni' encoded as per Table 8. Every 'moni' it sends should conform to the encoding in Table 8. The version label should not change from 'moni' to 'moni'. The data following the version label can change from 'moni' to 'moni' and can continue to be used for RTT calculations, or other purposes.

* If a TALI implementation is trying to determine if the far end of the TALI connection has implemented version 2.0, the implementation must examine any received 'moni' messages that arrive from the far end and see if they conform to the new stricter 'moni' encoding in Table 8. On receiving 'moni', a TALI 2.0 node will compare the 12 bytes of data in the VER LABEL field with a list of predetermined strings to determine the functionality of the TALI node it is connected to. If the data doesn't match any of the predetermined strings, the Far End is assumed to be a TALI 1.0 node.

* Each TALI implementation must assume that the far end of the connection is a 1.0 implementation until an arriving 'moni' announces that the far end supports TALI version 2.0. If a 'moni' never arrives, the implementation knows the far end has implemented version 1.0 of the specification.

* TALI 1.0 implementations can receive newly encoded 'moni' messages and simply ignore the data. The 1.0 implementations will continue to operate as if the far end is always a 1.0 node (ignore the data portion of the 'moni', convert 'moni' to 'mona', and return the 'mona').

* The next section provides more information regarding backwards compatibility (2.0 implementations connected to devices that implemented version 1.0 of the specification).

Octets	Field Name	Description	Field Type
0..3	SYNC	'TALI'	4 byte ASCII
4..7	OPCODE	'moni'	4 byte ASCII
8..9	LENGTH	Length (includes the version label and data fields)	Integer
10..21	Ver. Label See note	'vers xxx.yyy'	12 byte ASCII
22..X	DATA	Vendor Dependent Maximum length of this message (as coded in octets 8 -9, and stored in bytes 10-X) should not exceed 200 bytes.	Variable

Table 8: Version Control 'moni' Message

NOTE: xxx.yyy = provides the Major and Minor release number of the TALI specification being implemented.

```
001.000 = Tali version 1.0
002.000 = Tali version 2.0     // this specification.
002.001 = Tali version 2.1     // a minor change to 2.0
003.000 = Tali version 3.0
and so on.
```

The 'vers 002.000' field is an 12 byte field of field type 'ascii text'. As such, 'v' should be the first byte of the field that is transmitted out the wire.

4.3 Backwards Compatibility

As part of adding new functionality to the TALI specification, backwards compatibility from TALI version 2.0 to version 1.0 is required. Backwards compatibility is important since TALI 2.0 nodes may be connected to far ends that only support version 1.0; it is important that these 2 implementations continue to inter-operate, and that the 2.0 node falls back to supporting only 1.0 opcodes in this situation.

The previous section described how a TALI 2.0 implementation can use the 'moni' it sends to identify itself as a 2.0 node and how it can use the 'moni' it receives to determine if the far end is also a 2.0

node. In addition to the discussion in the previous section, the following bullets provide details regarding how backwards compatibility must be achieved:

* As documented in the version 1.0 specification, TALI 1.0 implementations that receive TALI messages with 'mgmt', 'xsrv', and 'spcl' opcodes will treat the message as a Protocol Violation (invalid opcode received). The Protocol Violation will cause the socket to be dropped immediately.

* It is therefore required that a 2.0 implementation only send 'mgmt', 'xsrv', and 'spcl' opcodes, after it has used a received 'moni' message to determine that the far end is a 2.0 (or later) implementation and has identified itself as a 2.0 (or later) implementation.

* Each TALI 2.0 implementations must use the 'moni' as described in the previous section to identify themselves as 2.0, and to learn if the far end is 2.0.

* Each TALI 2.0 implementation should maintain a variable as part of its state machine, 'far_end_version'. The 'far_end_version' should be initialized to 1.0 when the socket is established. Each time a 2.0 implementation receives 'moni', it should update the 'far_end_version' variable. If the 'moni' did not contain a version label, the 'far_end_version' should be reset to 1.0. If the 'moni' did contain a version label for 2.0 (or a later version), the 'far_end_version' should be set accordingly.

* Each time a 2.0 implementation receives a new 2.0 opcode ('mgmt', 'xsrv', and 'spcl') from the far end, it should examine the ' far_end_version'. If the 'far_end_version' indicates the far end is a 1.0 implementation, the received TALI message should be treated as a Protocol Violation (invalid opcode). If the 'far_end_version' is 2.0 (or later), the 2.0 implementation should process the received 'mgmt/xsrv/spcl' according to that nodes capabilities for that opcode.

* Each time a 2.0 implementation receives a request to send a TALI message with a 2.0 opcode ('mgmt/xsrv/spcl') from a higher layer of software, it should examine the 'far_end_version'. If the 'far_end_version' indicates the far end is a 1.0 implementation, the request to send the 2.0 opcode should be denied or ignored (an implementation decision) and the 2.0 opcode must NOT be sent to the far end. If the 'far_end_version' indicates the far end is 2.0 (or later), the request can be satisfied and the TALI message with the 2.0 opcode can be sent to the far end.

* Each TALI 2.0 implementation can provide a varying level of support for each of the three new 2.0 opcodes ('mgmt/xsrv/spcl'). In other words, an implementation may wish to only support SOME OF the primitives within the new opcodes. The level of support for each 2.0 opcode ('mgmt/xsrv/spcl') is independent of the other two 2.0 opcodes.

* The basic message structure for TALI messages using the new 2.0 opcodes is presented in Table 9.

* The minimal level of support that is required for each of the 2.0 opcodes (mgmt/xsrv/spcl) is to be able to receive TALI messages with these opcodes, recognize the new opcode, and ignore the message without affecting the state machine. The TALI state should not change. The socket connection should stay up. In other words, a 2.0 implementation can elect to ignore any received 'mgmt/xsrv/spcl' messages, if that implementation does not care to support the capability intended by that particular opcode.

* A partial level of support for a 2.0 opcode could be implemented. Partial support may consist of understanding the data structure for the 2.0 opcode, but only supporting some of the variants of the opcode. The message structure for each of the new 2.0 opcodes consists of an extra 'Primitive' field that follows the TALI opcode and message length fields. Each 'Primitive' is used to differentiate a variant of the opcode. It is envisioned that each new 2.0 opcode can be extended by adding new 'Primitives', as more capabilities are defined for the opcode, without having to add new TALI opcodes. A 2.0 implementation may understand and be willing to act on some of the 'Primitives' for an opcode, but not others. Receiving variants of a 2.0 opcode that an implementation does not understand need to be ignored and not affect the 2.0 state machine.

* The full level of support for a 2.0 opcode could be implemented. This support would consist of understanding and fully supporting every 'Primitive' within the 2.0 opcode.

Octets	Field Name	Description	Field Type
0..3	SYNC	'TALI'	4 byte ASCII
4..7	OPCODE	'mgmt', 'xsrv' or 'spcl'	4 byte ASCII
8..9	LENGTH	Length (length of the rest of this packet)	Integer
10..13	Primitive See note	'wxyz', or a 4 byte text that is appropriate for the given opcode	4 byte ASCII
14..X	DATA	The content of the data area is dependent on the opcode/ primitive combination	Variable

Table 9: Basic Message Structure for New 2.0 TALI Opcodes

NOTE: The Primitive field acts as a modifier for each opcode. Within an opcode, different operations or groups of operations can be defined and supported. The Primitive identifies each different operation or set of operations.

4.3.1 Generating Protocol Violations based on Received Messages

As implied by some of the bullets before Table 9, it is a goal of the 2.0 TALI specification to relax some of the error checking associated with the processing of received TALI messages.

Version 1.0 of this specification was very strict in detailing the fields that were checked for each received message. As each received message was processed, the SYNC code, opcode and length field of the message was checked; if any of these fields were invalid an internal protocol violation was generated. The processing of the protocol violation caused the socket to go down. In addition to the 3 specific checks (sync, opcode, length), the overall philosophy of version 1.0 was to treat any received data that the receiver did not understand, or which the receiver deemed to contain incorrectly coded fields as protocol violations.

Version 2.0 introduces the possibility of partial support for opcodes, partial support for primitives, and partial support for various fields (such as support for ANSI Pt Codes, but not ITU Pt Codes). Thus, the overall philosophy of how to treat received data that the receiver does not support needs to be relaxed from the

strict treatment in version 1.0. Version 2.0 implementations should be more tolerant when they receive messages they do not support (or which they believe contain incorrectly coded fields). This tolerance should include NOT treating these receives as protocol violations.

Version 2.0 implementations should perform the following level of strict/loose checks on the received messages:

* Checks against the sync codes, opcodes and lengths for version 1.0 and version 2.0 opcodes should be performed (against Table 3 and Table 11). Invalid data in these fields should be treated as cause for protocol violations.

* Checks against the opcode field for messages with new 2.0 opcodes (mgmt/xsrv/spcl) should be performed to determine whether the message can be processed by the implementation. If an implementation chooses to NOT support a particular 2.0 opcode, the received message should be discarded, internal data (such as measurements, logs of messages, user notifications) could record the event, and the TALI state should NOT be affected.

* Checks against the primitive field for messages with new 2.0 opcodes (mgmt/xsrv/spcl) should be performed to determine whether the message can be processed by the implementation. If an implementation does not understand a particular primitive, or has chosen NOT to implement the features for a particular primitive, the received message should be discarded, internal data (such as measurements, logs of messages, user notifications) could record the event, and the TALI state should NOT be affected.

* Checks against other field types in messages with new 2.0 opcodes (such as checking the encoding of a Point Code field for a valid Pt Code type) should also be performed in a 'soft' manner. Errors found in individual fields should cause the received message to be discarded, internal data (such as measurements, logs of messages, user notifications) could record the event, and the TALI state should NOT be affected.

The goals behind introducing this gentler treatment of errors in received data are as follows:

* To keep the socket up in order to perform the primary purpose of the connection (ie: to continue to transport SS7 data) in spite of improperly formatted/unsupported TALI messages related to other features/extensions/etc.

* To allow applications to support and use some of the features for a particular TALI revision without requiring the application to implement all of the functionality for the TALI revision.

* To increase the extensibility of the protocol. Looser receive checks are preferred in order to be able to add new primitives and new primitive operations as they are defined.

4.4 Overview of the TALI Message Structure

The basic message structure for all TALI messages is unchanged with the addition of new 2.0 opcodes. The base TALI header still consists of SYNC + OPCODE + LENGTH, as described in Table 2.

The message structure for the new 2.0 opcodes was shown in Table 9. These messages define an extra required field, PRIMITIVE, that follows the LENGTH field of Table 2.

4.4.1 Types of TALI Fields

Table 4 in the version 1.0 specification provided implementation notes for all the 'types of fields' found in the 1.0 specification. Version 2.0 of TALI continues to use all of the types provided in Table 4, and also defines some new fields that are used in TALI messages that use the new 2.0 opcodes. The following table introduces the new field types that are introduced with version 2.0. The types in Table 10 are used in addition to the types in Table 4 to implement the 2.0 TALI protocol.

Field Type	Implementation Notes for that Type
SS7 Point Code	Used to transmit point code information for ANSI or ITU variants of point codes across the TALI interface * The point code structure is 4 bytes. Byte 3 is used to identify the TYPE of point code. The actual point code is then encoded in bytes 0-2 (w/byte 0 being the least significant byte and the first byte transmitted across the wire) * Byte 3: encoding of the type of point code (PC) 0 = an ANSI Full PC 1 = an ITU International Full PC w/ a 3/8/3 coding scheme for zone/area/identifier 2 = an ITU National Full PC w/ a raw 14 bit PC 3 = unused 4 = an ANSI Cluster PC * For ANSI Full PC w/byte 3=0. These point codes are 24 bit point codes as follows: Byte 2 = Network Byte 1 = Cluster Byte 0 = Member * For ITU International Full PC (3/8/3) w/byte 3=1. These point codes use 14 bits (stored in the 14 least significant bits in bytes 0&1). Byte 2 is unused. The 14 bits should be interpreted as 3 bits of zone, 8 bits of area and 3 bits of signaling point identifier. The 3 bits of signaling point identifier are the 3 least significant bits. * For ITU National Full PC w/byte 3=2. These point codes use 14 bits (stored in the 14 least significant bits in bytes 0&1). Byte 2 is unused. The 14 bits represent a single 14-bit quantity that constitutes the point code. * For unused w/byte 3=3. Bytes 0 through 2 are undefined. * For ANSI Cluster PC, w/byte 3=4. These point codes are 24 bit point codes as follows: Byte 2 = Network Byte 1 = Cluster Byte 0 = 0. This field is ignored and should be coded as 0...all members of the cluster are implied * Byte 0 is the first byte that is transmitted across the wire, followed by byte 1, byte 2, then byte 3.

Bit-Field	* Field containing an array of N bits, where N is a multiple of 8. Bit-Field types should be transmitted such that the byte containing bits 0 through 7 is transmitted across the wire first, followed by the byte containing bits 8 through 15, etc. * The software for each implementation should be written in a manner that accounts for the required byte order of transmission (ie: the Big Endian/ Little Endian characteristics of the processor need to be dealt with in the software).
Version Label	A TALI version label is a 12 byte ASCII text field. The label is of a format 'vers xxx.yyy', where xxx.yyy are used to identify the version such as 002.000. As with other ASCII text fields, the first byte of the text field (the 'v') should be the first byte transmitted out the wire.
Primitive	Messages that use the new TALI 2.0 opcodes all have a 4 byte text ASCII field referred to as a Primitive. The Primitive acts as a modifier for the opcode. This allows a single opcode to be used to perform multiple actions.
Primitive Operation	A Primitive can be used to specify either a specific action or a set of actions. When the Primitive field is used to specify a set of actions, an operation field is used to pick a specific operation within that group of actions. Operation fields are 4 byte integers
Private Enterprise Code (PEC)	Various RFC documents have detailed a set of assigned numbers (RFC 1700, Assigned Numbers) and defined data structures (RFC 1155, Structure and Identification of Management Information for IP-based Internets) that are used on IP networks to provide network management information. Network Management Object Identifiers (OID) are used to recognize specific organizations, companies, protocols, and so on, in a manner that all vendors can agree on. An Object Identifier exists which uniquely describes each company that does business in the data/telecomm industry. That OID is referred to as an 'SMI Network Management Private Enterprise Code', which we are shortening to Private Enterprise Code of PEC in this document for simplicity. Each PEC is assumed to have

	a defined prefix of 'iso.org.dod.internet.private.enterprise' or (1.3.6.1.4.1). The PEC for each company can be found via a file at: ftp://ftp.isi.edu/in-notes/iana/assignments/ enterprise-numbers To encode the PEC for a vendor in each implementation of TALI, a 2 byte integer field is used. The contents of the integer field should match the PEC code for that company in the file mentioned above. For example, Tekelec, which has a PEC of 323, will code this 2 byte field as '0x0143'. Like other integer fields, the PEC value is transmitted Least Significant Byte first across the ethernet wire.

Table 10: Implementation for new field types introduced in TALI 2.0

4.5 Detailed TALI Message Structures for New 2.0 Opcodes

The message structures for opcodes defined in version 1.0 of TALI are unchanged from the information presented earlier, with the exception of the 'moni' message. The 2.0 format for the 'moni' message was described earlier.

Detailed message structures, and discussion of the capabilities, for each of the new 2.0 opcodes is provided in the following sections. Before discussing each opcode individually, Table 11 provides the minimum and maximum value of the LENGTH field that should be supported for each new opcode (as well as 'moni/mona'). Table 11 additionally shows the impact of ITU support that was added in 2.0. The routing label for ITU point codes only uses 4 octets instead of 7 octets as ANSI requires.

Opcode	Valid Length Field Values	Comments
moni	0-200 bytes	The overall length of the data portion for 'moni' on TALI 2.0 implementations is unchanged from version 1.0 of the specification and remains at 200 bytes to provide backwards compatibility.
mona	0-200 bytes	The overall length of the data portion for 'mona' on TALI 2.0 implementations is unchanged from version 1.0 of the specification and remains at 200 bytes to provide backwards compatibility.
mgmt	4-4096 bytes	The minimum length of 4 bytes is required to provide space for the Primitive field. The maximum length allows large TCP packets to be supported if desired.
xsrv	4-4096 bytes	The minimum length of 4 bytes is required to provide space for the Primitive field. The maximum length allows large TCP packets to be supported if desired.
spcl	4-4096 bytes	The minimum length of 4 bytes is required to provide space for the Primitive field. The maximum length allows large TCP packets to be supported if desired.
sccp	9-265 bytes	These are the valid sizes for the SCCP-ONLY portions of SCCP UDT MSUs.
isot	8-273 bytes	The length is the number of octets that in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte and all bytes in the SIF (Service Information Field) field. The MTP3 routing label is part of the SIF field.
mtp3	8-280 bytes	The length is the number of octets that in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte and all bytes in the SIF (Service Information Field) field. The MTP3 routing label is part of the SIF field.

saal	8-280 bytes	The length is the number of octets that in the MTP3 and higher layer(s) of the SS7 MSU. This length includes the SIO byte and all bytes in the SIF (Service Information Field) field. The MTP3 routing label is part of the SIF field. Seven (7) octets of SSCOP trailer is added to the message. The SSCOP trailer bytes are also included in the length.

Table 11: Valid Length Fields for Opcodes Affected by TALI 2.0

4.5.1 Management Message (mgmt)

The 'mgmt' opcode is intended to allow Management data, or data that will manage the operation of the device, to pass between the TALI endpoints over the socket connection. 'mgmt' messages can be received and processed in any of the TALI NEx-FEx states. Three PRIMITIVES are defined for use with this opcode:

* 'rkrp' - Routing Key Registration Primitive. This primitive allows the nodes to configure the SS7 traffic streams that they wish to receive over each socket. This 'routing key registration' is performed in-band, via TALI messages.

* 'mtpp' - MTP3 Primitives. This primitive allows SS7 MTP3 network management messages regarding the (un)availability and congestion states of SS7 devices to be passed to the IP devices SG.

* 'sorp' - Socket Options Registration Primitive. This primitive allows various socket options to be enabled/disabled by each TALI device.

As of version 2.0, the only defined primitives for the 'mgmt' opcode are 'rkrp', 'mtpp', and 'sorp'. In the future, more primitives can be added to this opcode to extend the Management capabilities of the SG or IP devices. The basic message structure for the 2.0 'mgmt' messages for all 3 of these primitives is as follows:

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'mgmt'
8..9	LENGTH	Length
10..13	Primitive	'rkrp', 'mtpp' or 'sorp' Each of these primitives specify a group of applicable management operations.
14..17	Primitive Operation	The operation field specifies the one operation within the group of operations identified by the primitive.
18..	Message Data	The content of the message data area is dependent on the combination of opcode/ primitive/operation fields. Each of those combinations could use a different message data structure.

Table 12: Message Structure for 'mgmt' opcode

4.5.1.1 Routing Key Registration Primitive (rkrp)

The 'rkrp' primitive allows IP nodes to modify the application routing key table in the SG by sending TALI messages to configure the SS7 traffic streams that they wish to receive over each socket. This 'routing key registration' is performed in-band, via TALI messages, as an alternative to using the SG user interface to configure the routing keys.

Recall from earlier discussion in this document that the specification supports five (5) types of fully specified routing keys:

* A key for SCCP traffic, where key = DPC-SI-SSN, where SI=3.

* A key for ISUP traffic, where key = DPC-SI-OPC-CIC Range, where SI=5. The CIC values for traditional ISUP are 14 bit quantities in ANSI networks and 12 bit quantities in ITU networks.

* A key for TUP traffic, where key = DPC-SI-OPC-CIC Range, where SI=4. This key is only supported for ITU networks. The CIC values for TUP keys are 12 bit quantities in ITU networks.

* A key for QBICC traffic (an extension of ISUP which uses 32 bit CIC codes), where key = DPC-SI-OPC-CIC, where SI=13. The CIC values for QBICC keys are 32 bit quantities for ANSI and ITU networks.

* A key for OTHER-MTP3-SI (non-SCCP, non-ISUP, non-QBICC for ANSI and non-SCCP, non-ISUP, non-QBICC, non-TUP for ITU) traffic, where key = DPC-SI

Each of these keys is fully specified key where the exact content of the MSU to be routed must match the data in the routing key.

Extensions to the routing keys have been added that will support 'partial match' or 'default' routing keys. The purpose of these extensions is to improve the handling of MSU traffic when no fully specified routing key exists that matches the MSU. Partial match and default routing keys are used when the SG can not find a fully specified routing key that can be used to route an MSU. Partial match keys can be used to provide closest-match routing such as 'ignore the CIC' for ISUP/QBICC/TUP traffic, or 'ignore the SSN' for SCCP traffic. Default keys are used when no full or partial routing key has been found as a last resort destination to route the MSU to.

The types of partial and default keys defined by the protocol are discussed in the following table. The 4th column in the table indicates the data structure that is used in the TALI rkrp message to perform operations on each partial/default key type. Note: The order of the keys in the table (from top to bottom) matches the hierarchical search order that an SG will use to attempt to find a routing key to use for an MSU. The partial and default keys are only used after attempting to find a fully specified key that matches the MSU.

Key Type	Key Attributes	Comments	Cross Reference
Partial	DPC-SI-OPC	Used as backup routes for CIC based traffic (but ignoring the CIC field).	4.5.1.1.2
Partial	DPC-SI	Used as backup routes for CIC based or SCCP traffic (but ignoring the OPC-CIC or SSN). Routes traffic based solely on DPC and SI of the MSU.	4.5.1.1.4
Partial	DPC	Used as a backup route for any MSU type. Routes traffic based solely on the DPC field.	4.5.1.1.4
Partial	SI	Used as a backup route for any MSU type. Routes traffic based solely on the SI field.	4.5.1.1.4
Default	-	If no other type of routing key for an MSU can be found, use this one.	4.5.1.1.5

Table 13: Partial and Default Routing Keys (in hierarchical order)

The specific capability requested in each 'rkrp' message is indicated via an 'RKRP Operation' field. These capabilities include:

* ENTER: The ENTER operation creates an association between a specific socket and a specific application routing key. The socket of the association is always the socket that the 'rkrp' was received on. The application routing key identifies an SS7 traffic stream that should be carried over that socket. Multiple sockets can be associated with the same application routing key, if so, they all receive traffic in a 'load sharing' mode. An override field can be used to remove any other socket associations for a particular routing key and add a single socket association. The ENTER operation is applicable for fully specified SCCP keys, CIC based keys (ISUP, Q.BICC, and TUP), OTHER-MTP3-SI keys, and all types of partial keys and to the default routing key.

* DELETE: The DELETE operation deletes an association between a specific socket and a specific application routing key. The socket of the association is always the socket that the 'rkrp' was

received on. Other socket associations for the same application routing key are NOT affected by the deletion. When the last socket association for a routing key is deleted, the entire routing key entry is removed from the database. The DELETE operation operation is applicable for fully specified SCCP keys, CIC based keys (ISUP, Q.BICC, and TUP), OTHER-MTP3-SI keys, and all types of partial keys and to the default routing key.

* SPLIT: The SPLIT operation is used to convert a single application routing key into 2 application routing keys that together cover the same SS7 traffic stream as the original key. Immediately after a split is performed, both of the resulting entries retain the same socket associations as the original routing key. When the split is completed, the socket associations can be modified for each of the 2 resulting ranges independent of the other range. The split operation is only applicable to fully specified CIC based keys (ISUP, QBICC, and TUP). Each fully specified CIC based key is uniquely identified by the combination of DPC/SI/OPC/CIC range. The CIC range is a continuous set of numbers from CICS(start) to CICE(end); the CIC range in the application routing key corresponds to the CIC value in a CIC based MSU.

* RESIZE: The RESIZE operation is used to modify the CIC range in for a single application routing key. The resize operation is only applicable to fully specified CIC based routing keys. The resize operation replaces the CICS/CICE values for a routing key with a new CIC range (NCICS/NCICE). A wide variety of NCICS/NCICE ranges can be supported based on the existing CICS/CICE; just about the only restriction is that the new range can not already exist in the database and can not overlap any other entry in the database. The socket associations for the routing key are NOT affected by the change in CICS/CICE. The SPLIT operation is applicable only to fully specified CIC based keys (ISUP, Q.BICC, and TUP).

The list of RKRP Operations (and their encodings) that are supported for TALI version 2.0 is as follows:

```
0x0001 - ENTER ISUP KEY
0x0002 - DELETE ISUP KEY
0x0003 - SPLIT ISUP KEY
0x0004 - RESIZE ISUP KEY
0x0005 - ENTER Q.BICC ISUP KEY
0x0006 - DELETE Q.BICC ISUP KEY
0x0007 - SPLIT Q.BICC ISUP KEY
0x0008 - RESIZE Q.BICC ISUP KEY
0x0009 - ENTER SCCP KEY
0x000A - DELETE SCCP KEY
```

0x000B - ENTER OTHER-MTP3-SI KEY
0x000C - DELETE OTHER-MTP3-SI KEY
0x000D - ENTER TUP KEY (ITU only)
0x000E - DELETE TUP KEY (ITU only)
0x000F - SPLIT TUP KEY (ITU only)
0x0010 - RESIZE TUP KEY (ITU only)
0x0011 - ENTER DPC-SI-OPC PARTIAL KEY
0x0012 - DELETE DPC-SI-OPC PARTIAL KEY
0x0013 - ENTER DPC-SI PARTIAL KEY
0x0014 - DELETE DPC-SI PARTIAL KEY
0x0015 - ENTER DPC PARTIAL KEY
0x0016 - DELETE DPC PARTIAL KEY
0x0017 - ENTER SI PARTIAL KEY
0x0018 - DELETE SI PARTIAL KEY
0x0019 - ENTER DEFAULT
0x001A - DELETE DEFAULT KEY
0x001B - MULTIPLE REGISTRATION SUPPORT

The message data area of the 'rkrp' messages will differ based on which RKRP Operation is specified. Several different structures are used, the correct structure can be identified by the RKRP Operation field.

In order to simplify the implementation, each of these structures will define a structure that will support all of the operations required for the key type. This means that based on the rkrp operation, some of the fields will be required, and some of the fields will not be applicable for each RKRP message. Unused fields should be initialized to 0 by the sender and ignored by the receiver.

4.5.1.1.1 RKRP Data Structures

4.5.1.1.1.1 Common Fields in all RKRP Messages

In the following subsections several different data structures to be used for various RKRP operations are presented. It should be noted that each of these data structures has the following fields in common. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer
2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings from in section 5.	Integer

Table 14: Common Fields in ALL 'rkrp' Data Structures

The primary purpose of requiring the data structures for all RKRP operations to begin with these same fields, is to provide a means for a receiver to reply to unknown RKRP messages in a consistent manner. When an implementation receives an RKRP request message it does not understand, it should turn the request into a reply and use the success/failure code to indicate that the operation is not supported (with an RKRP Reply Code of Unsupported rkrp Operation).

It is a requirement that these common fields continue to be used as new RKRP operations are added to this specification. This will ensure that the capability described in the previous paragraph will always exist.

4.5.1.1.1.2 CIC Based Routing Key Operations

The data structure used for 'rkrp' messages related to MSUs which are CIC based (ISUP, Q.BICC ISUP, and TUP (ITU only)) is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Note 1: The number of bits used in each CIC field will vary based on the SI and network type.

* ISUP operations (0x0001 - 0x0004) are assumed to use 14 bit CIC values from the corresponding fields in the structure when DPC/OPC indicate an ANSI network (12 bits used in ITU networks). Only the 14(12) least significant bits of the 32 bit CIC field will be used.

* Q.BICC ISUP operations (0x0005 - 0x0008) are assumed to use 32 bit CIC values from the corresponding fields in the structure.

* TUP operations (0x000d - 0x0010) are assumed to use 12 bit CIC values from the corresponding fields in the structure when DPC/OPC indicate an ITU network. Only the 12 least significant bits of the 32 bit CIC field will be used. TUP operations are not supported for ANSI networks.

Note 2: This same structure should be used to specify the partial key = DPC-SI-OPC(ignoreCIC). When specifying a DPC-SI-OPC partial key, the CIC fields in this structure should be set to 0 by the sender.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer

2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings listed in section 5.	Integer
2	RKRP flags	This is a 2 byte bit-field that provides 16 possible flags that can control various aspects of the operation. Bit 0 - An Override bit is used on the ENTER operation to control how the socket associations for a routing key should be manipulated. This flag determines if the ENTER is to add the given socket association in a 'load-sharing' mode or if the new association should replace (Override) all existing associations. This flag is only examined on ENTER operations. Bit 0=0, Load Sharing Mode Bit 0=1, Override Mode Bits 1-15, currently undefined	Bit-field
1	SI	Service Indicator. The SI field in an SS7 MSU identifies the type of traffic being carried by the MSU (0=SNM, 3=SCCP, 5=ISUP, etc). Each application routing key must specify a specific SI value that it relates to. SI should be 5 for ISUP keys. SI should be 13 for Q.BICC ISUP keys. SI should be 4 for TUP keys.	Integer

4	DPC	Destination Point Code. Each SS7 MSU contains a DPC that identifies the destination for the MSU. Each application routing key must specify a specific DPC value that it relates to.	SS7 Point Code
4	OPC	Origination Point Code. Each SS7 MSU contains a OPC that identifies the source of the MSU. ISUP routing keys must each specify a single OPC that the application routing key relates to.	SS7 Point Code
4	CICS	Circuit Identification Code Start. Each SS7 ISUP MSU contains a CIC code. Each ISUP/QBICC/TUP routing key identifies a range of CIC values that are applicable for the routing key. The CICS value is the low end of the CIC range.	Integer
4	CICE	Circuit Identification Code End. Each SS7 ISUP MSU contains a CIC code. Each ISUP/QBICC/TUP routing key identifies a range of CIC values that are applicable for the routing key. The CICE value is the high end of the CIC range.	Integer
4	SPLIT CIC	The SPLIT field is used on the SPLIT operation to specify where in the existing CIC range (given by CICS/ CICE) an existing routing key should be split into 2 routing keys. To be valid, the following relationship must be true before the SPLIT is performed: CICS < SPLIT <= CICE.	Integer

		After the SPLIT is performed, the 2 routing keys are as follows: CICS to SPLIT-1 SPLIT to CICE	
4	NCICS	The NCICS and NCICE fields are used on the RESIZE operation to specify how the CIC range for existing routing key should be modified. NCICS specifies the new value that should replace the existing CICS value in the routing key.	Integer
4	NCICE	The NCICS and NCICE fields are used on the RESIZE operation to specify how the CIC range for existing routing key should be modified. NCICE specifies the new value that should replace the existing CICE value in the routing key.	Integer

Table 15: Message Data Structure CIC based Routing Key Operations

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure in Table 15 based on the RKRP Operation field. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation / Field	ENTER (ISUP, QBICC, TUP)	DELETE (ISUP, QBICC, TUP)	SPLIT (ISUP, QBICC, TUP)	RESIZE (ISUP, QBICC, TUP)	ENTER/DELETE PARTIAL DPC SI OPC KEY
Request/Reply	R	R	R	R	R
Success/Failure	R	R	R	R	R
RKRP Flags	R	R	R	R	R
SI	R	R	R	R	R
DPC	R	R	R	R	R
OPC	R	R	R	R	R
CICS	R	R	R	R	NA
CICE	R	R	R	R	NA
SPLIT CIC	NA	NA	R	NA	NA
NCICS	NA	NA	NA	R	NA
NCICE	NA	NA	NA	R	NA

Table 16: Required/Not Applicable Fields for CIC based Routing Keys

4.5.1.1.1.3 SCCP Routing Key Operations

The data structure used for 'rkrp' messages related to SCCP routing keys is presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer
2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings listed in section 5.	Integer
2	RKRP flags	This is a 2 byte bit-field that provides 16 possible flags that can control various aspects of the operation. Bit 0 - An Override bit is used on the ENTER operation to control how the socket associations for a routing key should be manipulated. This flag determines if the ENTER is to add the given socket association in a 'load-sharing' mode or if the new association should replace (Override) all existing associations. This flag is only examined on ENTER operations. Bit 0=0, Load Sharing Mode	Bit-field

		Bit 0=1, Override Mode Bits 1-15, currently undefined	
1	SI	Service Indicator. The SI field in an SS7 MSU identifies the type of traffic being carried by the MSU (0=SNM, 3=SCCP, 5=ISUP, etc). Each application routing key must specify a specific SI value that it relates to. SI should be 3 for SCCP keys.	Integer
4	DPC	Destination Point Code. Each SS7 MSU contains a DPC that identifies the destination for the MSU. Each application routing key must specify a specific DPC value that it relates to.	SS7 Point Code
1	SSN	SubSystem Number. Each SCCP MSU contains a subsystem number that identifies the SCCP subsystem that should process the MSU. SCCP routing keys must each specify a single SSN that the application routing key relates to.	Integer

Table 17: Message Data Structure SCCP Routing Key Operations

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure in Table 17 based on the RKRP Operation field. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation Field	ENTER SCCP	DELETE SCCP
Request/Reply	R	R
Success/Failure	R	R
RKRP Flags	R	R
SI	R	R
DPC	R	R
SSN	R	R

Table 18: Required/Not Applicable Fields for SCCP Routing Keys

4.5.1.1.1.4 DPC-SI, DPC and SI based Routing Key Operations

The data structure used for 'rkrp' messages related to DPC-SI based (either full keys for non-sccp, non-cic based traffic, or partial keys for CIC based or SCCP), DPC based (partial key), and SI based (partial key) operations is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer

2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings from section 5.	Integer
2	RKRP flags	This is a 2 byte bit-field that provides 16 possible flags that can control various aspects of the operation. Bit 0 - An Override bit is used on the ENTER operation to control how the socket associations for a routing key should be manipulated. This flag determines if the ENTER is to add the given socket association in a 'load-sharing' mode or if the new association should replace (Override) all existing associations. This flag is only examined on ENTER operations. Bit 0=0, Load Sharing Mode Bit 0=1, Override Mode Bits 1-15, currently undefined	Bit-field
1	SI	Service Indicator. The SI field in an SS7 MSU identifies the type of traffic being carried by the MSU (0=SNM, 3=SCCP, 5=ISUP, etc). Each application routing key must specify a specific SI value that it relates to.	Integer

4	DPC	Destination Point Code. Each SS7 MSU contains a DPC that identifies the destination for the MSU. Each application routing key must specify a specific DPC value that it relates to.	SS7 Point Code

Table 19: Message Data Structure DPC/SI, DPC and SI based Routing Key Operations

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure in Table 19 based on the RKRP Operation field. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation / Field	ENTER/ DELETE OTHER MTP3 SI	ENTER/ DELETE DPC-SI PARTIAL	ENTER/ DELETE DPC ONLY	ENTER/ DELETE SI ONLY
Request/Reply	R	R	R	R
Success/Failure	R	R	R	R
RKRP Flags	R	R	R	R
SI	R	R	NA	R
DPC	R	R	R	NA

Table 20: Required/Not Applicable Fields for DPC/SI, DPC and SI based Routing Keys

4.5.1.1.1.5 Default Routing Key Operations

The data structure used for 'rkrp' messages related to entering and deleting a default routing key is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer
2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings listed in section 5.	Integer
2	RKRP flags	This is a 2 byte bit-field that provides 16 possible flags that can control various aspects of the operation. Bit 0 - An Override bit is used on the ENTER operation to control how the socket associations for a routing key should be manipulated. This flag determines if the ENTER is to add the given socket association in a 'load-sharing' mode or if the new association should replace (Override) all existing associations. This flag is only examined on ENTER operations. Bit 0=0, Load Sharing Mode	Bit-field

		Bit 0=1, Override Mode Bits 1-15, currently undefined	

Table 21: Message Data Structure for Default Routing Keys

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure in Table 21 based on the RKRP Operation field. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation Field	ENTER DEFAULT	DELETE DEFAULT
Request/Reply	R	R
Success/Failure	R	R
RKRP Flags	R	R

Table 22: Required/Not Applicable Fields for Default Routing Keys

4.5.1.1.1.6 Support for Multiple RKRP Registration Operations

The intent of support for multiple RKRP operations within a single TALI message (opcode = 'mgmt', primitive = 'rkrp') is to decrease the message count and byte overhead on network transmission when performing massive registration sequences.

This functionality is added by 2 mechanisms:

* a new RKRP operation (0X001B, MULTIPLE REGISTRATIONS SUPPORT) is defined. This operation is meant to be used in a query/reply manner to determine if the far end supports multiple RKRP registrations per TALI message before using such capability.

* The basic 'rkrp' message structure is extended to allow multiple rkrp operations to follow one another in a tali message.

4.5.1.1.1.6.1 Multiple Registrations Support

A new RKRP operation and accompanying data structure are defined to determine if a far end device supports multiple RKRP registration operations per TALI message.

The data structure used for the 'multiple registrations support' operation is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	RKRP Operation	Identifies which 'rkrp' operation is desired.	Integer
2	Request/ Reply	Identifies whether the 'rkrp' message is a request (from an IP node to SG) for some type of 'rkrp' action, or a reply to a previous request (from the SG back to the IP node). This integer field uses the following encodings: 0x0000=Request 0x0001=Reply. See Success/ Failure code for more info.	Integer
2	Success/ Failure Code	Provides a success/failure indication as part of the reply back to the IP node for each processed request. This field is only used when the Request/Reply field is 0x0001. This field uses the encodings listed in section 5.	Integer
4	Operations Per Message	This field is used by the reply to tell the requester the maximum # of RKRP registration operations per TALI message that are supported by the implementation. * This field should be set to 0 when the request/ reply field is set to Request. * This field should be set to the Maximum # of operations per TALI message that a TALI implementation is	Integer

		willing to support when the request/reply field is set to Reply.	

Table 23: Message Data Structure for Multiple Registrations Support Operation

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure above. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation / Field	MULTIPLE REGISTRATIONS SUPPORT REQUEST	MULTIPLE REGISTRATIONS SUPPORT REPLY
Request/Reply	R	R
Success/Failure	R	R
Operations Per Message	R	R

Table 24: Required/Not Applicable Fields for Multiple Registrations Support Operation

4.5.1.1.1.6.2 Multiple RKRP Operations in a Single Message

After using the MULTIPLE REGISTRATIONS SUPPORT operation to determine that the far end supports multiple RKRP operations per TALI message, a device wishing to use this functionality can begin sending more than 1 registration request/reply per message. To do so, the basic message structure for an 'mgmt' opcode (presented in Table 12) can be extended so that each operation directly follows the previous operation in the TALI message. An example showing a TALI message with 3 RKRP operations in it would look as follows:

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'mgmt'
8..9	LENGTH	Length. The length should be set such that all (3 in this example) operations are accounted for.
10..13	Primitive	'rkrp'
14..17	Primitive Operation #1	The fisrt operation field identifies a specific rkrp operation to be performed.
18..x	Message Data for Operation #1	The length of the message data (and the interpretation of those bytes) for operation #1 depends on the message data required for rkrp operation #1
x+1.. x+4	Primitive Operation #2	The fisrt operation field identifies a specific rkrp operation to be performed.
x+5..y	Message Data for Operation #2	The length of the message data (and the interpretation of those bytes) for operation #2 depends on the message data required for rkrp operation #2
y+1.. y+4	Primitive Operation #3	The fisrt operation field identifies a specific rkrp operation to be performed.
y+5..z	Message Data for Operation #3	The length of the message data (and the interpretation of those bytes) for operation #3 depends on the message data required for rkrp operation #3

Table 25: Message Structure for 'mgmt' opcode with multiple 'rkrp' operations in 1 TALI Message

It should be reiterated that in order to avoid unpredictable behavior, a node using the 'multiple registrations per TALI msg' capability must be sure the far end device supports the capability. The only way to be sure of this is to successfully send a MULTIPLE REGISTRATION SUPPORT request and receive a MULTIPLE REGISTRATION SUPPORT reply.

4.5.1.2 MTP3 Primitive (mtpp)

The 'mtpp' primitive allows IP nodes to receive status regarding point code (un)availability and congestion levels. These messages provide information similar to the TFP/TFA (TransFer Prohibited and TransFer Allowed), TFC (TransFer Congested) and RCT (Route Congestion Test) messages that are encoded as SS7 SNM (Signaling Network Management) MSUs in traditional SS7 networks. The 'mtp3 primitives' allow this status information to be transferred in-band, via TALI messages, to the IP nodes.

The specific information provided in each 'mtpp' message is indicated via an 'MTPP Operation' field. These capabilities provided by the various MTPP Operation fields include:

* POINT CODE UNAVAILABLE: This primitive operation announces that an SS7 Point Code is Unavailable (ie: the SG has NO route available to send traffic for the destination). The PT CODE field indicates which SS7 Pt Code this operation is concerned with.

* POINT CODE AVAILABLE: This primitive operation announces that an SS7 Point Code is Available (ie: the SG has SOME route available to send traffic for the destination). The PT CODE field indicates which SS7 Pt Code this operation is concerned with.

* REQUEST FOR POINT CODE STATUS: This primitive operation provides a way for one end of the connection to poll the other end for the available/unavailable status of a specific SS7 pt code. For instance, the IP node can poll the SG - Can you send traffic successfully for the destination indicated? The receiver of the request will reply to the request with either a point code available or pt code unavailable primitive respectively.

* CLUSTER UNAVAILABLE: This primitive operation announces that an entire Cluster of SS7 Point Codes (ex: 10-10-*) are Unavailable (ie: the SG has NO route available to send traffic for any of the destinations in that cluster). The PT CODE field indicates which SS7 Cluster Pt Code this operation is concerned with.

* CLUSTER AVAILABLE: This primitive operation announces that at least 1 SS7 Point Code within a cluster is Available (ie: the SG

has SOME route available to send traffic for at least 1 of the destinations in that cluster). The PT CODE field indicates which SS7 Cluster Pt Code this operation is concerned with.

* REQUEST FOR CLUSTER STATUS: This primitive operation provides a way for one end of the connection to poll the other end for the available/unavailable status of a cluster of SS7 pt codes. For instance, the IP node can poll the SG - Can you send traffic successfully for any of the destinations in the cluster? The receiver of the request will reply to the request with either a cluster available or cluster unavailable primitive respectively.

* CONGESTED DESTINATION: This primitive operation announces that the path towards an SS7 Point Code is Congested. The PT CODE field indicates which SS7 Pt Code this operation is concerned with. The CONGESTION LEVEL field indicates the severity of the congestion.

* REQUEST FOR CONGESTION STATUS: This primitive operation provides a way for one end of the connection to poll the other end for the congestion status of an SS7 pt code. For instance, the IP node can poll the SG - Is the path to the specified destination still congested? This request is used to abate congestion towards an SS7 destination.

  * As an implementation note: Upon receiving this request, the SG will generate and send a Route Congestion Test (RCT), SS7 Network Management Message with a priority set to match the congestion level in the request. The RCT is sent towards the SS7 destination. If the SS7 destination is still congested, the RCT will result an SS7 Transfer Controlled (TFC) arriving back at the SG, which will be converted into a CONGESTED DESTINATION primitive and sent on to the IP node.

* USER PART UNAVAILABLE: SS7 nodes send User Part Unavailable messages when a user part that is mounted on a node is no longer available for service. This primitive operation provides a way for an IP Node to receive the same information as the SS7 UPU message.

In order to simplify the implementation, a single data structure is defined to be used for all of the 'mtpp' operations. Depending on the 'mtpp operation', some of the fields will be required, and some of the fields will not be applicable for each MTPP message. Unused fields should be initialized to 0 by the sender and ignored by the receiver. The data structure used for 'mtpp' messages is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	MTPP Operation	Identifies which 'mtpp' operation/capability is provided in this message. This integer field uses the following encodings: 0x0001 = PC Unavailable 0x0002 = PC Available 0x0003 = Request for PC Status 0x0004 = Cluster Unavailable 0x0005 = Cluster Available 0x0006 = Request for Cluster Status 0x0007 = Congested Destination, w/Cong Level 0x0008 = Request for Congestion Status 0x0009 = User Part Unavailable	Integer
4	Concerned Point Code	Identifies the SS7 Point Code that is relevant to the mtpp operation. The mtpp operation is concerning this point code (or cluster).	SS7 Point Code
4	Source Point Code	This field is only used on the 'Congested Destination' and 'Request for Congestion Status' operations. * When used in an 'Congestion Destination' operation, this field contains the Pt Code of the Source of the traffic that was experiencing congestion as it made its way to the Concerned Pt Code. In terms of the original SS7 MSUs (the TransFer Controlled MSU) that provided congestion information, the CPC of the TFC is the 'Concerned Point	SS7 Point Code

		Code' of the resulting MTPP primitive and the DPC of the TFC is the 'Source Point Code' of the resulting MTPP primitive. * When used in an 'Request for Congestion Status' operation, this field indicates which Source Pt Code is trying to abate the congestion of the concerned Pt Code. In terms of the original SS7 MSUs (the Route Congestion Test MSU) that is used to poll for congestion, the DPC of the RCT is the 'Concerned Point Code' of the MTPP primitive and the OPC of the RCT is the 'Source Point Code' of the MTPP primitive.	
2	Congestion Level	This field is used on the 'Congested Destination' and 'Request for Congestion Status' operations to indicate the congestion level of the destination. This integer field uses the following encodings: 0x0000 = Congestion Level 0 0x0001 = Congestion Level 1 0x0002 = Congestion Level 2 0x0003 = Congestion Level 3	Integer
2	Cause Code	This field is used on the 'User Part Unavailable' operation to indicate the Cause Code for why the UPU is being sent. This integer field uses the following encodings: 0x0000 = Cause Unknown 0x0001 = User Part Unequipped 0x0002 = User Part Inaccessible	Integer

2	User ID	This field is used on the 'User Part Unavailable' operation to indicate which user part is unavailable. The User ID field identifies the type of traffic that was unavailable (0=SNM, 3=SCCP, 5=ISUP, etc).	Integer

Table 26: Message Data Structure for use with the 'mtpp' Primitive

The following table indicates the Required (R), or Not Applicable (NA) status for each field of the message data structure in Table 26 based on the MTPP Operation field. As mentioned previously, unused fields (those marked NA) should be initialized to 0 by the sender and ignored by the receiver.

Operation \ Field	Concerned Point Code	Source Point Code	Congestion Level	Cause Code	User ID
PC Unavailable	R	NA	NA	NA	NA
PC Available	R	NA	NA	NA	NA
Request for PC Status	R	NA	NA	NA	NA
Cluster Unavailable	R	NA	NA	NA	NA
Cluster Available	R	NA	NA	NA	NA
Request for Cluster Status	R	NA	NA	NA	NA
Congested Destination w/ Cong. Level	R	R	R	NA	NA
Request for Congestion Status	R	R	R	NA	NA
User Part Unavailable	R	NA	NA	R	R

Table 27: Required/Not Applicable Fields for MTPP Operations

4.5.1.3 Socket Option Registration Primitive (sorp)

The 'sorp' primitive allows IP nodes to set various options on a socket by socket basis. This allows the IP node some control over the communication that will occur across the TALI connection. The 'sorp' primitives allows this socket option control to be transferred in-band, via TALI messages, to the IP nodes.

The SORP primitives capabilities that are available to the IP device in SG are as follows:

* Set SORP Flags: Used to set the flags bit field. The receiver of this message should store the bit settings indicated in the SORP Flag field.

* Request Current SORP Flags Settings: Used to poll for the status of the bit field options. The receiver of this message should send a Reply w/ Current SORP Flag settings.

* Reply w/ Current SORP Flag Settings: Used to reply to a poll, indicating the current bit field settings to the far end.

As of TALI 2.0, each socket option is stored as a bit in a 32 bit bit-field. Each bit in the field indicates the setting for 1 option. A bit field with a 0 value indicates the option is DISABLED. A bit field with a 1 value indicates the option is ENABLED. The following options are currently supported:

* ENABLE/DISABLE BROADCAST PHASE MTPP PRIMITIVES: Traditional STPs send Broadcast Phase TFPs and TFAs to all adjacent nodes when the point code availability changes for destinations in the STP's SS7 routing table. These Broadcast Phase TFA/TFP SS7 messages are converted into TALI mtpp primitives by SG nodes such as the SG. The ENABLE/DISABLE BROADCAST PHASE MTPP PRIMITIVES options allow each IP node to tell the remote end whether the IP node wants to receive the mtpp primitives that result from SS7 broadcast phase messages.

  * As an implementation note: In the SG, each defined socket has a flag, 'enable_broadcast_phase_primitives', which is initialized to FALSE each time the socket connects. The IP node should send the ENABLE BROADCAST PHASE MESSAGES operation to the SG to announce that it wants to receive unsolicited status changes for a particular socket. As the SG is determining where to send broadcast phase TFAs/TFPs, it will interrogate the 'enable_broadcast_phase_primitives' flag for each socket on that socket.

* ENABLE/DISABLE RESPONSE METHOD MTPP PRIMITIVES: Traditional STPs send Response Method TFPs to adjacent nodes when the adjacent nodes continue to send MSUs to the STP that can not be delivered (ie: the STP has told the adjacent node that a destination is Unavailable, but the adjacent node continues to send traffic destined for that unavailable DPC to the STP). These Response Method messages are sent in response to MSUs that are received at the STP. These Response Method TFP messages are converted into TALI mtpp primitives by SG nodes such as the SG. The ENABLE/DISABLE RESPONSE METHOD MTPP PRIMITIVES options allow each IP node to tell the remote end whether the IP node wants to

receive the mtpp primitives that result from SS7 response method messages. In addition to response method TFPs, 2 other SS7 Network Management messages, namely TFCs (transfer controlled) and UPUs (user part unavailable), fall into this RESPONSE METHOD grouping. TFCs and UPUs are similar to response method TFPs due to the fact that a previous action by the IP Node (sending traffic toward some destination) has caused a response method event back to the IP Node. The primary difference between response method TFPs versus response method TFCs/UPUs is that the response method TFP is converted to an MTPP primitive and sent back to only the original socket, while response method TFCs/UPUs may need to be replicated to multiple sockets (after being converted to mtpp primitives) since there is no way to tell which socket caused the response method event.

* As an implementation node: In the SG, each defined socket has a flag, 'enable_response_method_primitives', which is initialized to FALSE each time the socket connects. The IP node should send the ENABLE RESPONSE METHOD MTPP PRIMITIVES operation to the SG to announce that it wants to receive response method TFPs when appropriate for a particular socket. Before the SG sends a response method TFP (converted to a mtpp primitive) back to an IP node, the SG will interrogate the 'enable_response_method_primitives' flag for that socket and only perform the send if the flag allows it.

* ENABLE/DISABLE NORMALIZED SCCP: Version 1.0 of TALI specified that the 'sccp' TALI opcode must be used on point to multipoint connections in order to transmit SCCP MSUs between the SG and IP nodes. When using the 'sccp' opcode, the MTP3 header portion of the original SS7 MSU was stripped from the MSU and was NOT part of the data transmitted across the TALI connection. The sender of the 'sccp' TALI message was responsible for duplicating the DPC/OPC fields from the MTP3 header into appropriate fields in the SCCP portion of the message (into the Called/Calling Party Address Pt Code fields) before sending as a 'sccp' opcode. This option provides a way to send SCCP MSUs across TALI point to multipoint connections that includes the MTP3 header as part of the data transmitted, and does NOT involve any modification to the original SS7 SCCP MSU. When the ENABLE NORMALIZED SCCP primitive is received, SCCP MSUs should be sent across the TALI interface using the 'mtp3' opcode. This transmission should include the entire MTP3 header + the sccp portion of the original MSU. No modification of the original SS7 MSU should occur. When the DISABLE NORMALIZED SCCP primitive is received, SCCP MSUs should be sent across the TALI interface using the 'sccp' opcode as specified in version 1.0 of TALI.

* ENABLE/DISABLE NORMALIZED ISUP: Version 1.0 of TALI specified that the 'isot' TALI opcode must be used on point to multipoint connections in order to transmit ISUP MSUs between the SG and IP nodes. When using the 'isot' opcode, the original SS7 MSU, including the MTP3 header portion, was transmitted in a 'isot' TALI message. This option indicates that the far end would prefer to receive ISUP MSUs using the 'mtp3' TALI opcode as opposed to the 'isot' opcode. When the option is ENABLED, the 'mtp3' opcode is used to transmit ISUP MSUs, including the MTP3 header, across the TALI connection. When the option is DISABLED, the 'isot' opcode is used as in TALI Release 1.0.

The data structure used for 'sorp' messages is as presented in the next table. The data structure below should begin at byte 14 of the TALI message as shown in Table 12.

Octets	Field Name	Description	Field Type
2	SORP Operation	Identifies which 'sorp' operation/capability is provided in this message. This integer field uses the following encodings: 0x0001 = Set SORP Flags 0x0002 = Request Current SORP Flags Settings 0x0003 = Reply w/ Current SORP Flag Settings	Integer
2	SORP Flags	A 4 byte bit-field that uses each bit as an enabled/ disabled flag for a particular socket option. Bit x = 0 indicates the option is DISABLED. Bit x = 1 indicates the option is ENABLED. The assignments for each BIT are as follows: Bit 0 = Broadcast Phase MTPP Primitives Bit 1 = Response Method MTPP Primitives Bit 2 = Normalized SCCP Bit 3 = Normalized ISUP	Bit-Field

Table 28: Message Data Structure to be used for 'sorp' Primitive

4.5.2 Extended Service Message (xsrv)

The Extended Service, 'xsrv', opcode is added to the TALI 2.0 protocol to lay the groundwork for providing a means to transport other types of service traffic (beyond 'sccp', 'isot', 'mtp3', and 'saal') in future revisions of this protocol without having to define a new opcode as each new service type is identified and added. The PRIMITIVE field will uniquely identify each new service type as they are added. It is envisioned that some 'xsrv' messages can be received and processed in any of the TALI NEx-FEx state, while some other 'xsrv' messages can only be received and processed in the NEA-FEA state (such as Service data in version 1.0 of TALI).

There are no specific PRIMITIVES defined for this opcode in this release. It is expected that some new service messages will be added in the future. This opcode provides for grouping of the new service data types.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'xsrv'
8..9	LENGTH	Length
10..13	Primitive	To be determined
14.. 2000	Message Data	To be determined

4.5.3 Special Message (spcl)

The Special Message, 'spcl', opcode is added to the TALI 2.0 protocol to provide a way for vendors to build special services into their TALI implementations that are only activated when the implementation is connected to other equipment implementing the same special services. 'spcl' messages can be received and processed in any of the TALI NEx-FEx states. This opcode is intended to provide a general means to discover more information regarding who the TALI session is connected to, and to provide means to enable special features based on the vendor/implementation on the far end.

As part of the 2.0 specification, 4 primitives are initially defined for this opcode:

* 'smns' - Special Messages Not Supported.
* 'qury' - Query.
* 'rply' - Reply.
* 'usim' - UnSolicited Information Message.

Additional primitives can be added in future versions of the TALI protocol.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'spcl'
8..9	LENGTH	Length
10..13	Primitive	'smns' - special messages not supported 'qury' - query 'rply' - reply 'usim' - UIM (unsolicited information msg)
14..X	Data	Vendor dependent

4.5.3.1 Special Messages Not Supported (smns)

This message is sent as a response to a 'spcl' message with a 'qury' PRIMITIVE. A node may send out this message when it wants the Far End to know that it does not support 'spcl' messages and wishes not to receive them in the future.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'spcl'
8..9	LENGTH	Length
10..13	Primitive	'smns'

4.5.3.2 Query Message (qury)

This message can be sent to Query the far end of the connection (ie: try to find out more information about the VENDOR, TALI version, or other features). It is expected that each 2.0 implementation would respond to a 'qury' with a 'rply'.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'spcl'
8..9	LENGTH	Length
10..13	Primitive	'qury'

4.5.3.3 Reply Message (rply)

The 'rply' message provides a way for a TALI 2.0 implementation to identify itself in more detail. The information included in the reply includes:

* PEC - a 2 byte field that identifies the vendor for the TALI implemenation.

* Version Number - a 12 byte field that identifies the TALI version of the implementation.

* Other Vendor Specific Data - the format of any remaining data that a particular vendor wants to provide is specific to each vendor.

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'spcl'
8..9	LENGTH	Length
10..13	Primitive	'rply'
14..15	PEC	Private Enterprise Code * (Vendor ID Number, Integer Field)
16..27	Version Label	'vers xxx.yyy'
28..?	Other Vendor Specific Data	Free Format data area, specific to each vendor

*See Table 4 for details on the PEC field.

4.5.3.4 Unsolicited Information Message (USIM)

A 'usim' provides the same information as the 'rply' primitive. The 'usim' can be sent at any time by a 2.0 implementation (whereas the 'rply' should only be sent in reply to a 'qury').

Octets	Field Name	Description
0..3	SYNC	'TALI'
4..7	OPCODE	'spcl'
8..9	LENGTH	Length
10..13	Primitive	'usim'
14..15	PEC	Private Enterprise Code * (Vendor ID Number, Integer Field)
16..27	Version Label	'vers xxx.yyy'
28..?	Other Vendor Specific Data	Free Format data area, specific to each vendor

4.6 TALI Timers

Version 2.0 of the TALI specification does not introduce any new timers. The T1-T4 timers defined previously remain in effect.

While, it is expected that most implementations wishing to identify themselves as 2.0 (or later) would use a non-zero value for T4 - this is a not a hard requirement. The only requirement for identifying yourself as 2.0 is to send at least 1 'moni' as per the 2.0 format upon connection establishment.

4.7 TALI User Events

Version 2.0 of the TALI specification does not introduce any new user events. The user events defined in Section 3.4 (mgmt open, mgmt close, mgmt allow, mgmt proh, connection established, connection lost) remain in effect.

4.8 TALI States

Version 2.0 of the TALI specification does not introduce any new TALI states. The TALI states defined in Section 3.6 remain in effect.

4.9 TALI Version 2.0 State Machine

This section provides the state machine that must be followed by each TALI 2.0 implementation in order to be compliant with this specification. As mentioned throughout this document, a 2.0 implementation is based on several small additions to a 1.0 implementation and each 2.0 implementation must be willing to inter-operate in a backwards compatible mode (a 2.0 implementation connected to a 1.0 implementation must fall back to 1.0 features only).

4.9.1 State Machine Concepts

Before presenting the actual state machine, several concepts are discussed.

4.9.1.1 General Protocol Rules

A set of general protocol rules was presented in the 1.0 specification, in section 3.7.1.1; those rules are still applicable to 2.0 implementations. In addition to those earlier rules, the following rules are also applicable to 2.0 nodes:

* A 2.0 implementation should identify the TALI version it has implemented via the 'moni' message

* A 2.0 implementation should process any received 'moni' messages, attempting to determine the TALI version of the far end. A 2.0 implementation must use an internal flag, such as 'far_end_version', to track the TALI version that the far end of the connection has implemented. The 'far_end_version' flag should be initialized to version 1.0.

* A 2.0 implementation should reject/ignore internal requests (from software layers in it's own product, or requests from the management interface for the device) to send TALI messages that require 2.0 opcodes when the far end is a 1.0 implementation. A 2.0 implementation should only send TALI messages that require new 2.0 opcodes (mgmt, xsrv, spcl) when it knows the far end is capable of processing those opcodes (when 'far_end_version' is 2.0 or greater).

* Upon receiving a TALI message with a 2.0 opcode, a 2.0 implementation should interrogate its 'far_end_flag'; if the far end is not 2.0 or greater, the arrival of the message should be treated as a Protocol Violation. If the far end is 2.0 or greater, the message should be processed according to the nodes 2.0 capabilities, or ignored (if the node has chosen not to implement any 2.0 functionalities).

4.9.1.2 Graceful Shutdown of a Socket

The steps to perform a graceful shutdown of each socket were presented in the 1.0 specification, in section 3.7.1.2. Those steps are not changed for 2.0 implementations.

4.9.1.3 TALI Protocol Violations

Each TALI implementation must detect when violations of the TALI protocol have occurred and react accordingly. Protocol violations include:

* Invalid sync code in a received message
* Invalid opcode in a received message
* Invalid length field in a received message
* Not receiving an 'allo' or 'proh', in response to the origination of a 'test' , before the T2 timer expires
* Receiving Service Messages on a prohibited socket.
* TCP Socket errors - Connection Lost
* Receiving a TALI message with a 2.0 opcode ('mgmt', 'xsrv', ' spcl') from a far end that has not identified itself as a 2.0 implementation.

In the state machine that follows, State/Event combinations that should be treated as protocol violations are indicated via a 'PV' in the state/event cell. All of the 'PV' events are then processed as per the 'Protocol Violation' row in the table.

4.9.2 The State Machine

Internal Data required for State Machine:

* boolean sock_allowed. This flag indicate whether the NE is allowed to carry Service Messages.

* Far_end_version. This enumeration should track the TALI version of the far end of the socket.

Initial Conditions:
sock_allowed = FALSE
far_end_version = 1.0
state = OOS
no timers running

State / Event	OOS	Connecting	NEP-FEP	NEP-FEA	NEA-FEP	NEA-FEA
T1 Exp.			Send test Start T1 Start T2	Send test Start T1 Start T2	Send test Start T1 Start T2	Send test Start T1 Start T2
T2 Exp.			PV	PV	PV	PV
T3 Exp.			PV	PV		
T4 Exp.			Send moni Start T4	Send moni Start T4	Send moni Start T4	Send moni Start T4
Rcv test			Send proh	Send proh	Send allo	Send allo
Rcv allo			Stop T2 NEP-FEA	Stop T2	Stop T2 NEA-FEA	Stop T2
Rcv proh			Stop T2 Send proa	Stop T2 Send proa NEP-FEP	Stop T2 Send proa	Stop T2 Flush or reroute Send proa NEA-FEP
Rcv proa			Stop T3	Stop T3		
Rcv moni			Update 'far end version' based on moni Convert to mona Send mona	Update 'far end version' based on moni Convert to mona Send mona	Update 'far end version' based on moni Convert to mona Send mona	Update 'far end version' based on moni Convert to mona Send mona

Rcv mona			Implementation dependent	Implementation dependent	Implementation dependent	Implementation dependent
Rcv Service			PV	If T3 run Process Else PV	PV	Process
Rcv mgmt			If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process
Rcv xsrv			If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process
Rcv spcl			If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process	If FE< 2.0 PV Else Process
Connect. Estab.		Start T1 Start T2 Start T4 (if non-0) if sock_ allowed = TRUE send allo send test NEA-FEP else send proh send test NEP-FEP				
Connect. Lost			PV	PV	PV	PV
Protocol Violat.			Stop all timers Close the socket Connecting	Stop all timers Close the socket Connecting	Stop all timers Close the socket Connecting	Stop all timers Close the socket Connecting

Mgmt. Open Socket	Open socket Conne- cting					
Mgmt. Close Socket		Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS	Stop all timers Close the socket OOS
Mgmt. Prohibit Socket	sock_ allow- ed = FALSE	sock_allo- wed=FALSE	sock_all- owed= FALSE	sock_all- owed= FALSE	sock_all- owed= FALSE send proh start t3 NEP-FEP	sock_all- owed= FALSE send proh start t3 NEP-FEA
Mgmt. Allow Traffic	sock_ allow- ed = TRUE	sock_allo- wed=TRUE	sock_all- owed= TRUE send allo NEA-FEP	sock_all- owed= FALSE send allo NEA-FEA	sock_all- owed= TRUE	sock_all- owed= TRUE
User Part Msgs.	reject data	reject data	reject data	reject data	reject data	send data
Request to Tx mgmt			If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process
Request to Tx xsrv			If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process
Request to Tx spcl			If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process	If FE<2.0 Ignore Else Process

Table 29: TALI 2.0 State Machine

4.10 TALI 2.0 Specification Limitations

Several limitations with the TALI 2.0 specification are identified. These are considered possible areas for expansion of the protocol in the future:

* Support for different types of routing keys is limited. It is envisioned that new routing key types will need to be added and supported as new applications are identified.

* An opcode, or new primitive within an existing opcode, could be added as a means of returning unknown or unsupported data to the sender. In addition to discarding and storing internal debug data, an implementation may want to return the original TALI message to the sender when the receiver of the message deems the message to be unknown, unsupported, or incorrectly formatted.

5. Success/Failure Codes

The following list provides all the known success/failure codes that are being used for the rkrp feature. New defines will be added to the end of the list as they are identified.

Error #	Meaning
1	Transaction successfully completed.
2	Length of TALI msg is insufficient to contain all required information for rkrp operation
3	Unsupported 'rkrp' operation
4	Invalid SI. SI must be in range 0..15
5	Invalid SI/operation combination. Split and resize only supported for SI=4,5,13. Enter, delete and override supported for all SI.
6	Invalid DPC. Point code cannot be zero, and must be full point code.
7	Invalid SSN. SSN must be in range 0..255.
8	Invalid OPC. Point code cannot be zero, and must be full point code.
9	Invalid CICS. Must be in range appropriate for SI and PC type.
10	Invalid CICE. Must be in range appropriate for SI and PC type.
11	Invalid CIC range. CICS must be less than or equal to CICE. On a split operation, CICS must be strictly less than than CICE (cannot split an range with only one entry).
12	Invalid NCICS. Must be in range appropriate for SI and PC type.

13	Invalid NCICE. Must be in range appropriate for SI and PC type.
14	Invalid new CIC range. NCICS must be less than or equal to NCICE.
15	Invalid SPLIT value. Must be in range appropriate for SI and PC type. Must be greater than CICS and less than or equal to CICE.
16	No free entries in table.
17	CIC range overlaps but does not match existing entry.
18	Entry already has 16 associations.
19	Entry to be changed not found in table.
20	New entry would overlap another entry (allowed to overlap the entry being changed, but no others).
21	Entry to be deleted not found in table.
22	TUP routing keys are not supported for ANSI networks

6. Security Considerations

TALI is an interface for the transport of SS7 traffic and management messages across an IP network. As with traditional PSTN networks, the IP networks using TALI are expected to well engineered systems. The use of virtual private networks and firewalls is to be expected. In addition, the use of IPSEC will bring added security benefit to the network.

7. References

[1] Bell Communications Research, Specification of Signaling System Number 7, GT-246-CORE, Bellcore, Issue 1, December 1994.

[2] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[3] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

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

[5] Logical Link Control, IEEE 802.2 and ISO 8802.2

[6] Carrier Sense Multiple Access with Collision Detection (Ethernet), IEEE 802.3 and ISO 8802-3 CSMA/CD.

[7] Virtual LAN, IEEE 802.1 Q and ISO 8802-1Q CSMA/CD.

[8] Bell Communications Research, Generic Requirements for CCS Nodes Supporting ATM High-Speed Signaling Links (HSLs), GR-2878-CORE, Issue 1, Bellcore, November 1995.

[9] Bell Communications Research, Asynchronous Transfer Mode (ATM) and ATM Adaptation Layer (AAL) Protocols, GR-1113-CORE, Bellcore.

[10] American National Standards Institute, B-ISDN Signaling ATM Adaptation Layer - Service Specific Connection Oriented Protocol (SSCOP), T1.637.

[11] American National Standards Institute, B-ISDN Signaling ATM Adaptation Layer - Service Specific Coordination Function for Support of Signaling at the Network Node Interface (SSCF at the NNI), T1.645.

[12] American National Standards Institute, B-ISDN Signaling ATM Adaptation Layer - Layer Management for the SAAL at the NNI, T1.652.

8. Acknowledgments

The authors would like to thank Ken Morneault for his comments and contributions to the document.

9. Authors' Addresses

David Sprague
Tekelec
5200 Paramount Pkwy.
Morrisville, NC 27560
Phone: +1 919-460-5563
EMail: david.sprague@tekelec.com

Dan Brendes
Tekelec
5200 Paramount Pkwy.
Morrisville, NC 27560
Phone: +1 919-460-2162
EMail: dan.brendes@tekelec.com

Robby Benedyk
Tekelec
5200 Paramount Pkwy.
Morrisville, NC 27560
Phone: +1 919-460-5533
EMail: robby.benedyk@tekelec.com

Joe Keller
Tekelec
5200 Paramount Pkwy.
Morrisville, NC 27560
Phone: +1 919-460-5549
EMail: joe.keller@tekelec.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.