SINAP/SS7 ISDN User Part (ISUP) Guide

Part Number: R8053-14

April 2004

SINAP/SS7 ISDN User Part (ISUP) Guide

Stratus Technologies
R8053-14

Notice

Manual Name: *SINAP/SS7 ISDN User Part (ISUP) Guide*

Part Number: R8053
Revision Number: 14
Updated for SINAP/SS7 Release Number: 14.0
Publication Date: April 2004

Stratus Technologies, Inc.
111 Powdermill Road
Maynard, Massachusetts 01754-3409

Contents

Figures

Tables

Preface

The Purpose of this Manual

The *SINAP/SS7 ISDN User Part (ISUP) Guide* (R8053) documents the ISDN User Part (ISUP) services feature of the Stratus Intelligent Network Applications Platform (SINAP) SS7 product.The ISUP services feature provides the signaling functions required to support circuit-switched services for voice and non-voice connections to an integrated services digital network (ISDN).

Audience

This manual is intended for system administrators who are responsible for configuring and managing a SINAP/SS7 installation and for programmers who are responsible for developing ISUP applications that run on SINAP/SS7 in a Signaling System Number 7 (SS7) network.

Before using the *SINAP/SS7 ISDN User Part (ISUP) Guide* (R8053), you should be familiar with the following manuals.

- *SINAP/SS7 Technical Overview* (R8055)
- *SINAP/SS7 User's Guide* (R8051)
- *SINAP/SS7 Programmer's Guide* (R8052)

Revision Information

This manual has been revised to support the Stratus ft Linux ® operating system, the ISUP_UPU_FEATURE environment variable, and miscellaneous corrections to existing text.

Manual Organization

This manual is divided into the following chapters and appendices:

- Chapter 1 "ISUP Services Overview" describes how the existing SINAP and MultiStack products were modified to provide support for the ISUP services feature.
- Chapter 2 "Administering the ISUP Services Feature" describes the system administration tasks you must perform to configure and activate the ISUP services feature on your SINAP/SS7 installation.

- Chapter 3 “Designing and Developing ISUP Services Applications” provides information about how to design and develop a SINAP application that uses ISUP services.
- Chapter 4 “Implementing the Distributed Logical Point Code Feature” provides information on how to support distributed ISUP user applications on two SINAP nodes on different Continuum systems (or the same system).
- Chapter 5 “ISUP Services Reference” provides reference information about the functions and data structures that compose the ISUP application programming interface (API).
- Appendix A “ISUP Services Environment Variables” describes the ISUP services-specific environment variables for SINAP/SS7.
- Appendix B “ISUP Services Messages and Parameters” contains two tables that describe ISUP services messages and parameters. The first, Table B-1 “ISUP Services Message Types,” lists each ISUP services message, along with its acronym and binary and hexadecimal codes. The second, Table B-2 “ISUP Services Parameters,” contains a list of the parameters used in ISUP services messages.
- Appendix C “ISUP Services Error Messages” contains a list of the ISUP services-related error messages and associated error descriptions that are defined in the SINAP/SS7 `ca_error.h` include file.
- Appendix D “Belgian Version of ISUP Services” describes the Belgium version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix E “Chinese Version of ISUP Services” describes the Chinese version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix F “France1 Version of ISUP Services” describes the France1 version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix G “G500 Version of ISUP Services” describes the G500 version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix H “German Version of ISUP Services” describes the German version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix I “Italian Version of ISUP Services” describes the Italian version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix J “ITU97 Version of ISUP Services” describes the 1997 version of ITU-T ISUP services that is supported on a SINAP/SS7 node.
- Appendix K “Netherlands Version of ISUP Services” describes the Netherlands version of ISUP services that is supported on a SINAP/SS7 node
- Appendix L “NTT-IC Version of ISUP Services” describes the NTT-InterConnect (IC) version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix M “Swedish Version of ISUP Services” describes the Swedish version of ISUP services that is supported on a SINAP/SS7 node.
- Appendix N “Taiwanese Version of ISUP Services” describes the Taiwanese version of ISUP services that is supported on a SINAP/SS7 node.

- Appendix O “Q.767 Version of ISUP Services” describes the Q.767 version of ISUP services that is supported on a SINAP/SS7 node.

Notation Conventions

This manual uses the following notation conventions.

- *Italics* introduces or defines new terms. For example:

  The *ISUP services feature* offers support for the services provided by the ISDN User Part (ISUP) protocol of Signaling System Number 7 (SS7).

- **Boldface** emphasizes words in text. For example:

  If the ISUP services feature is active, existing MTP applications **cannot** register with SINAP with an application name of `IS` or a process name of `MG`.

- `Monospace` represents text that would appear on your display screen (such as commands, functions, filenames and directories). For example:

  The environment variable, `ISUP_FEATURE`, activates ISUP functionality on a SINAP/SS7 node.

Format for Commands and Functions

This manual uses the following format conventions for documenting commands and functions. Note that the command and function descriptions do not necessarily include each of the following sections.

SYNOPSIS

The syntax of the command or function. The following chart explains the notation used in the synopsis.

Notation	Meaning
<*argument_1* >	Required argument.
argument_1...	Required argument for which multiple values are allowed.
{*argument_1* | *argument_2* }	Set of arguments that are mutually exclusive; you must specify at least one of these elements.
[*argument_1*]	Optional argument.
[*argument_1*]...	Optional argument for which you can specify multiple values.

Notation	Meaning
`[argument_1 | argument_2]`	Set of optional arguments that are mutually exclusive; you can specify only one of these elements.
Note: Dots, brackets, and braces are not literal characters; you should **not** type them. Any list or set of arguments can contain more than two elements.	

INCLUDE FILES
A list of the files that must be included in any application that calls this function.

DESCRIPTION
A detailed description of the command or function. A *command description* includes descriptions of the command's arguments. A *function description* includes descriptions of the function's input or output parameters. An *input parameter* defines data that the application programmer must provide to the function (for example, the name of an application). An *output parameter* defines data that the function provides or returns (for example, a pointer to a data structure).

EXAMPLES
Examples of usage.

RETURN VALUES
Values returned by the command or function.

SEE ALSO
A list of related information.

Related Manuals

Refer to the following Stratus manuals for related documentation.

- *SINAP/SS7 User's Guide* (R8051) provides instructions for configuring and managing a SINAP/SS7 installation.
- *SINAP/SS7 Programmer's Guide* (R8052) provides information on designing and developing an application to run under SINAP/SS7 and also contains information on testing and debugging an application.
- *SINAP/SS7 Technical Overview* (R8055) contains an overview of the SINAP/SS7 product and lists the national and international standards to which the product complies.
- *SINAP/SS7 Installation Guide* (R8060) provides instructions for installing the SINAP/SS7 software on an HP-UX or FTX UNIX system.
- *SINAP Products Glossary* (R8010) contains definitions for terms and acronyms used in all Stratus intelligent network products, such as SCEnic, Service Node, and SINAP/SS7.

A Note on the Contents of Stratus Manuals

Stratus manuals document subroutines and commands of the user interface. Any other commands and subroutines contained in the operating system are intended solely for use by Stratus personnel and are subject to change without warning.

Accessing Documentation

SINAP product documentation is provided on CD-ROM. You can request a documentation CD-ROM in either of the following ways:

- Call the CAC (see ‘‘Customer Assistance Center (CAC)).
- If your system is connected to the Remote Service Network (RSN), add a call using the Site Call System (SCS). See the scsac(1) man page for more information.

When requesting a documentation CD-ROM, please specify the product and platform documentation you desire, as there are several documentation CD-ROMs available.

Commenting on This Guide

Stratus welcomes any corrections or suggestions for improving this guide. Contact the CAC to provide input about this guide.

Customer Assistance Center (CAC)

The Stratus Customer Assistance Center (CAC), is available 24 hours a day, 7 days a week. To contact the CAC, do one of the following:

- Within North America, call 800-828-8513.
- For local contact information in other regions of the world, see the CAC web site at http://www.stratus.com/support/cac and select the link for the appropriate region.

Chapter 1
ISUP Services Overview

This manual describes the *ISUP services feature* that supports the services provided by the ISDN User Part (ISUP) protocol of Signaling System Number 7 (SS7). ISUP provides the signaling functions required to support circuit-switched services for voice and non-voice connections to an integrated services digital network (ISDN).

The SINAP/SS7 system provides the base implementation of ISUP services that conform to: the 1993 edition of the CCITT Recommendations for ISUP; the 1992 edition of the ANSI standard for ISUP; and the 1992 edition of the Japan NTT standards for ISUP. Although call-control and circuit-management capabilities are not supported by this version of ISUP services, applications utilizing these features can be developed by the user.

Implementations of country-specific versions of ISUP services adhere to one of these base standards of ISUP services, with modifications to conform to that individual country's standards for ISUP. For example, the Chinese ISUP services version is based on the ITU-T (CCITT) Recommendation with modifications to accommodate the Chinese standards for ISUP.

Table 1-1 provides a listing of all currently supported country-specific versions of ISUP services.

Table 1-1. Valid Country-Specific ISUP Services Versions

For Information on...	Refer to...
Belgium	Appendix D - "Belgian Version of ISUP Services"
China	Appendix E - "Chinese Version of ISUP Services"
France	Appendix F - "France1 Version of ISUP Services"
Australia	Appendix G - "G500 Version of ISUP Services"
German	Appendix H - "German Version of ISUP Services"
Italian	Appendix I - "Italian Version of ISUP Services"
ITU97	Appendix J - "ITU97 Version of ISUP Services"
Netherlands	Appendix K - "Netherlands Version of ISUP Services"

Table 1-1. Valid Country-Specific ISUP Services Versions*(Continued)*

For Information on...	Refer to...
NTT-IC	Appendix L - “NTT-IC Version of ISUP Services”
Sweden	Appendix M - “Swedish Version of ISUP Services”
Taiwan	Appendix N - “Taiwanese Version of ISUP Services”
Q.767	Appendix O - “Q.767 Version of ISUP Services”

- An environment variable, `ISUP_FEATURE`, that activates the ISUP services functionality on a SINAP/SS7 node.
- A Dynamic Linked Library (DLL) to simplify the development of ISUP services functions on Stratus ft Linux and HP-UX systems. The `libCASL` library is used to aid the development of service applications that are to be deployed in the SS7 network. The `libissl` library contains the ISUP services functions and data structures required to develop ISUP services applications. Table 1-2 provides a more complete description of these libraries and their locations.

  To use the ISUP services feature, an application must register with SINAP at the ISUP boundary and follow specific registration criteria.
- Node management functionality to configure and manage ISUP services on your system.

Table 1-2. CASL and ISUP Libraries

Library Type	Library Name	Location
Stratus ft Linux Operating System		
DLL	`libCASL.so` `libissl.so`	`$SINAP_MASTER/Library/` During installation of the SINAP/SS7 software, a link to these files is automatically generated and placed in `/usr/lib`.
Archive	`libCASL.a` `libissl.a`	`$SINAP_MASTER/Library/` (`$SINAP_MASTER` is the location of the default master copy of the SINAP/SS7 software.)
HP-UX Operating System		
DLL	`libCASL.sl` `libissl.sl`	`$SINAP_MASTER/Library/` During installation of the SINAP/SS7 software, a link to these files is automatically generated and placed in `/usr/lib`.

Table 1-2. CASL and ISUP Libraries *(Continued)*

Library Type	Library Name	Location
Archive	`libCASL.a` `libissl.a`	`$SINAP_MASTER/Library/` (`$SINAP_MASTER` is the location of the default master copy of the SINAP/SS7 software.)

- The sample programs `isup_send.c` and `isup_recv.c` that show how to use the ISUP services API for an application so that it can send and receive ISUP messages.

The SINAP/SS7 ISUP services feature currently is available only on installations (stacks) running the following network variants: ANSI, CCITT, China, and NTT. If you plan to run existing Message Transfer Part (MTP) applications with this release, see “Considerations for Existing SINAP Applications” for important compatibility information.

Considerations for Existing SINAP Applications

If you plan to run existing MTP applications with this SINAP/SS7 release, consider the following issues.

- If the ISUP services feature is active, existing MTP applications **cannot** register with the SINAP node if they use an application name of `IS` or a process name of `MG`. These names are reserved for the *ISUP services manager*, which performs the management functions necessary to provide ISUP services to applications running on SINAP/SS7 system.
- If the ISUP services feature is active, existing MTP applications that register with the SINAP node with a Service Information Octet (SIO) of 5 **must** follow the application-registration rules described in the section “Initiating a Connection Request” in Chapter 3 of this manual. Otherwise, the applications will fail to register with the SINAP node.
- If the ISUP services feature is not active, MTP applications can register according to the MTP application registration rules described in “User Part (MTP) Application Registration” in the *SINAP/SS7 Programmer’s Guide* (R8052).

Chapter 2

Administering the ISUP Services Feature

This chapter is intended for system administrators responsible for installing and configuring the SINAP/SS7 ISUP services feature. It contains the following topics.

- “Activating the ISUP Services Feature”
- “Configuring Route Sets for Remote SSPs”
- “Configuring Kernel-tunable Parameters”
- “Setting ISUP Services Timers”
- “Handling RSC Messages”
- “Enabling the Generic Name Parameter in IAM Messages (ANSI)”
- “Handling Queries for Unequipped Circuits (ANSI)”

Activating the ISUP Services Feature

By default, the ISUP services feature is turned off. To activate it, you must uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh])` and set the variable to one of the ISUP versions listed in Table 2-1 before starting or restarting the SINAP/SS7 system.

NOTE

The ISUP version you specify in the environment variable must be compatible with the network variant you configured on the node during installation of the SINAP/SS7 software as described in Table 2-1. See the *SINAP/SS7 Installation Guide* (R8060) for details on configuring network variants.

Table 2-1. Valid ISUP Versions and Corresponding Network Variants

Network Variant	Valid ISUP Version
ANSI	ANSI
CCITT	ACIF_G500
	BELGIUM
	CCITT
	FRANCE1
	GERMANY
	ITALY
	ITU97
	MEXICO
	NETHERLANDS
	Q767
	SPAIN
	SWEDEN
	TAIWAN
	UK
CHINA	CHINA
NTT	NTT
	NTT_IC

NOTE

You must activate all environment variables to be enabled on a SINAP node **before** you start or restart the SINAP node.

Appendix A, "ISUP Services-Specific Environment Variables" provides a complete description of supported ISUP services-related environment variables. For more detailed information on defining SINAP environment variables, see "Defining SINAP Environment Variables" in the *SINAP/SS7 User's Guide* (R8051).

ISUP services applies only to the ANSI, CCITT, China, and NTT network variants. The TTC network variant **does not** support ISUP services functionality.

When the `ISUP_FEATURE` environment variable is properly defined, you can use all ISUP services features that are valid for the ISUP services version implemented on the SINAP/SS7 node.

- The ISUP Services Support Library (ISSL), an application programming interface (API) for developing applications that use ISUP services.
- CASL functions and structures that simplify the development of service applications for deployment on SS7 networks.
- Various environment variables you can define to enable or disable specific ISUP services features. Some variables apply only to certain versions of ISUP services. See Appendix A for a complete list of ISUP services environment variables.
- The `$SINAP_HOME/Include` directory that contains additional ISUP services files, including `issl.h and issl_sit.h`.
- Sample programs showing how to use the ISUP services functions in applications to enable them to send and receive ISUP messages.
- ISUP services timers that define time-out values for performing various tasks.
- The BITE subsystem and `sy` utility that log and display ISUP messages.

Handling Circuit Group Blocking Acknowledgment Messages (ANSI)

In the ANSI version of ISUP services, you can set the environment variable `ISUP_CGBA_PER_2CGB` to activate the feature (based on ANSI 1992 standards) that returns a Circuit Group Blocking Acknowledgment (CGBA) message to the originator of the Circuit Group Blocking (CGB) message whenever the SINAP node receives two CGB messages within a 5-second timer period. If the node receives the second CGB message after the timer expires, it discards the first CGB message received and waits to receive the second CGB message within the 5-second timer interval, then sends the CGBA message.

When the `ISUP_CGBA_PER_2CGB` environment variable is not set, the SINAP/SS7 system uses ANSI 1988 processing. In this case, the SINAP node waits to receive the second CGB message before the timer expires and then sends the CGBA message. If the node receives the second CGB message after the timer expires, the node discards the first CGB message. Then the node starts the timer and waits to receive the second CGB message before sending the CGBA message to the originator of the CGB message.

NOTE

In the CCITT version of ISUP services, the SINAP node sends a CGBA message for each CGB message received.

To activate this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.`[`csh` or `sh`]) **before** starting or restarting the SINAP node.

```
ISUP_CGBA_PER_2CGB
```

Disabling Automatic Congestion Control

The ISUP protocol software automatically implements the Automatic Congestion Control (ACC) parameter in a Release (REL) message when the level of available buffer resources falls below an internal threshold at the time an REL is sent.

In the ANSI version of ISUP services, you can set an environment variable to disable the ACC parameter in the REL message. To activate this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.`[`csh` or `sh`]) **before** starting or restarting the SINAP node.

```
ISUP_REL_NO_ADD_ACC
```

It is not necessary to provide a value for this variable. The SINAP/SS7 system only verifies the existence of the variable.

NOTE

If you are running the MultiStack product, you must set the environment variable separately for each node on which you want to activate this feature.

Configuring Route Sets for Remote SSPs

An ISUP application manages a particular set of Remote Service Switching Points (SSPs). As a system administrator, you must ensure that each SSP has an active route set defined for it; otherwise, the application will return an error message. Note that the remote SSP should be defined as the route set's Destination Point Code (DPC). For example, if an ISUP application is responsible for the SSPs 3002, 3003, 3004, and 3005, verify that the node configuration includes a route set for each of the following DPCs: 3002, 3003, 3004, and 3005.

Use the Man-Machine Language (MML) command `DISPLAY-RSET` to determine whether a route set exists for each remote SSP. For information about the command's format, see the command's manual (man) page.

If necessary, use the MML commands `CREATE-RSET` and `CONFIGURE-RSET`, respectively, to create and activate the necessary route sets. For information about the `CREATE-RSET` and `CONFIGURE-RSET` commands, see their man pages or see the *SINAP/SS7 User's Guide* (R8051).

Configuring Kernel-tunable Parameters

There are certain system parameters you may need to increase based on the number of circuits or Circuit Identification Codes (CICs) an ISUP services application manages. The `MAX_ISUP_CIRCUITS` variable contained in the include file `sinap.h` (located in the directory `$SINAP_HOME/Include/`) defines the maximum number of circuits or CICs each application user process can handle. The maximum value allowed for this variable is 20,000. Since each SINAP node can have up to 15 processes, a SINAP node can handle up to 300,000 circuits (20,000 x 15 processes=300,000).

The `isup_cfg_process()` function configures an ISUP services application and defines the maximum configuration parameters for circuits (and remote SSPs) for each application user process. Thereafter, the process(es) cannot configure more remote circuit groups than the maximum values specified by this function call. This function also enforces the maximum values defined by `MAX_ISUP_CIRCUITS` in the `sinap.h` include file.

The `isup_cfg_process()` function allocates a block of memory (used for message buffers, event structures, and control structures) from user heap space based on the number of circuits or CICs requested. The actual number of bytes for this memory block is based on the sum of the structure sizes. The number of bytes *allocated* is based on the ranges of circuit numbers listed in Table 2-2.

Table 2-2. Byte Allocation per Circuit

Number of Circuits Requested	Bytes Allocated Per Circuit
1 through 1000	24,364
1001 through 5000	11,500
5001 through 20,000	5,304

A user application request for 500 circuits, would result in the `isup_cfg_process()` function allocating 12.2 megabytes from user heap space. A request for 20,000 circuits would result in the allocation of 106 megabytes from user heap space.

The sample programs (`isup_send.c` and `isup_recv.c`) use approximately 10 megabytes of user heap before the `isup_cfg_process()` is invoked. Some ISUP applications may be larger and use more heap space to start the application. Note that an ISUP application uses additional heap space equal to the size of the CIC status structure (`isup_status_t`).

For each site that uses additional circuits or CICs, you should increase the appropriate kernel tunable parameters for your system as required. These kernel tunable parameters are described in the two sections following this one. After tuning kernel parameters, **you must reboot the system to begin using the new parameter settings**.

Tunable Kernel Parameters on HP-UX Systems

On HP-UX systems, the kernel, normally configured to use 67.1 megabytes (or 67,108,864 bytes) for the maximum data segment size (`maxdsiz`), can be tuned to increase the data segment size for each user application process. Stratus recommends you configure a maximum `maxdsiz` of approximately 167.7 megabytes (or 167,772,160 bytes). This allows you to configure up to 20,000 CICs.

You should increase the value of the tunable parameter `maxdsiz` as required depending on the number of circuits requested by the user application. The suggested value for `maxdsiz` is `0xA000000`.

You can view the current `maxdsiz` value and modify the parameter through the `sam` utility using the menu option "Kernel Configuration and Configurable Parameters." For detailed information on using the sam utility and reconfiguring the kernel parameters, see the HP-UX manual *HP-UX System Administration Tasks* (B2355-90079).

Tunable Kernel Parameters on Stratus ft Linux Systems

Stratus recommends you configure a maximum `maxdsiz` of approximately 167.7 megabytes (or 167,772,160 bytes). This allows you to configure up to 20,000 CICs. You should increase the value of the tunable parameter `maxdsiz` as required depending on the number of circuits requested by the user application. The suggested value for `maxdsiz` is `0xA000000`.

You can view the current `maxdsiz` value and modify the parameter through the `sysctl` utility. For detailed information on using the `sysctl` utility and reconfiguring the kernel parameters, refer to man pages of `sysctl`.

Setting ISUP Services Timers

You can examine and change the values of the ISUP services timer values stored in the SINAP/SS7 system tables. You access these system tables by means of the MML commands `DISPLAY-SYSTAB` and `CHANGE-SYSTAB`. For detailed information on these commands, see their man pages or see the discussions on "Displaying the MTP and SCCP System Tables" and "Changing the System Table Timer and Link-Congestion Threshold Settings" in the *SINAP/SS7 User's Guide* (R8051).

When you issue the `DISPLAY-SYSTAB` or `CHANGE-SYSTAB` command to access an ISUP timer, specify the following values for the command arguments.

- For *`TABID`*, specify the value `ISUPTM` if you are entering the MML command in free-form mode. If you are building the command using the Terminal Handler menus (options `Network Commands`, `Change Commands`, and `Change System Table`), specify the value `i` or `I` for the prompt `System table to be modified`.
- For *`TIMER`*, specify the appropriate ISUP services timer name (see Table 2-2).

The following examples show sample free-form commands for accessing ISUP services timers. The following command displays the values of all ISUP services timers.

```
DISPLAY-SYSTAB:TABID=ISUPTM:TIMER=ALL;
```

The following command changes the value of the ISTCCR timer.

```
CHANGE-SYSTAB:TABID=ISUPTM:TIMER=ISTCCR:NEWTIME=80S;
```

Table 2-2 lists the names of the ISUP services timers and their valid timeout ranges and default values. These timers are defined in the `network.h` include file (located in the `$SINAP_HOME/Include` directory). For more information on ISUP timers, see Table A.1 in ITU-T Recommendation Q.764 or Table 3 in ANSI T1.113.4. Note that the SINAP/SS7 ISUP timer names have a prefix of `IS`. For example, the `SINAP/SS7 IST1` timer listed in the "Timer Name" column corresponds to the `T1` timer listed in the "Symbol" column of ITU-T Recommendation Q.764 Table A.1 or Table 3 in ANSI Standard T1.113.4.

Table 2-3. ISUP Services Timers

Timer Name	Event Activating Timer	Timeout Value (Range of Values)	Default Value (in Seconds)
`IST1` *1	Release (REL) message sent	15 to 60 seconds	30
`IST2`	Controlling exchange receives a Suspend (SUS) user message	3 minutes	180
`IST4`	Receipt of an MTP-STATUS event with the cause "`remote user unavailable`"	5 to 15 minutes	480
`IST5` *2,3,4	Initial Release (REL) message sent	5 to 15 minutes	480
`IST6` *5	Controlling exchange receives a Suspend (SUS) network message	See ITU-T Q.118	10 (ITU) 21 (ANSI)
`IST7` *5	Latest Initial Address (IAM) message sent	20 to 30 seconds	25
`IST8` *5, *13	Receipt of an Initial Address (IAM) message requiring a continuity check	10 to 15 seconds	12
`IST9` *5	Controlling exchange receives an Address Complete (ACM) message	See ITU-T Q.118	10 (ITU) 180 (ANSI)
`IST12` *3	Blocking (BLO) message sent	15 to 60 seconds	35
`IST13`	Initial Blocking (BLO) message sent	5 to 15 minutes	480
`IST14` *6, *12	Unblocking (UBL) message sent	15 to 60 seconds	35

Table 2-3. ISUP Services Timers*(Continued)*

Timer Name	Event Activating Timer	Timeout Value (Range of Values)	Default Value (in Seconds)
IST15 *12	Initial Unblocking (UBL) message sent	5 to 15 minutes	480
IST18	Circuit Group Blocking (CGB) message sent	15 to 60 seconds	35
IST19 *5	Initial Group Blocking Message (CGB) sent	5 to 15 minutes	480
IST20	Circuit Group Unblocking (CGU) message sent	15 to 60 seconds	35
IST21 *5	Initial Group Unblocking (CGU) message sent	5 to 15 minutes	480
IST22	Circuit Group Reset (GRS) message sent	15 to 60 seconds	35
IST23 *5	Initial Circuit Group Reset (GRS) message sent	5 to 15 minutes	480
IST27 *11, *13	Receipt of a continuity check failure indication	4 minutes 3 minutes	240 (ITU) 180 (ANSI)
IST31	Release of an ISUP signaling connection based on connection-oriented SCCP. During this time period, the call reference is frozen (unused).	More than 6 minutes	360
IST33 *5	Information Request (INR) message sent	12 to 15 seconds	13
IST34 *7	Awaiting a Continuity Test (COT) message [ANSI only]	10 to 15 seconds	12
IST34 *7	Receipt of an indication of a segmented message for an Initial Address (IAM), Address Complete (ACM), Call Progress (CPG), Answer (ANM), or Connect (CON) message [CCITT only]	2 to 4 seconds	4
IST36	Awaiting a Continuity (COT) message [CCITT only]	10 to 15 seconds	12
ISTCCR *7	Continuity recheck timer [ANSI only]	20 seconds	20
ISTCRA *5	Circuit Reservation Acknowledge (CRA) message sent [ANSI only]	10 seconds	10
ISTCRM *9	Circuit Reservation (CRM) message sent [ANSI only]	3-4 seconds	4

Table 2-3. ISUP Services Timers*(Continued)*

Timer Name	Event Activating Timer	Timeout Value (Range of Values)	Default Value (in Seconds)
`ISTEXM` *10	Exit (EXM) message to be sent [ANSI only]	network dependent	20
`ISTGRES` *11	Receipt of a Circuit Group Reset (GRS) message	5 seconds	5
`ISTVAL` *8	Circuit validation test/response (CVT/CVR) [ANSI only]	10 seconds	10

NOTES

1. This timer corresponds to timer T16, mentioned in ITU-T Recommendation Q.764. Timer T16 is the same as timer T1, except that it is set when sending the Reset Circuit (RSC) message and not the Release (REL) message.
2. This timer corresponds to timer T17, mentioned in ITU-T Recommendation Q.764. Timer T17 is the same as timer T5, except it is set when sending the Reset Circuit (RSC) message and not the Release (REL) message.
3. Expiry action "alert maintenance personnel" is sent to the application by `isup_get_msg()` of primitive event `EVTLSISTAIND` with event type `CIR_OUTORD`.
4. Expiry action "send reset circuit message" is sent to the application by `isup_get_msg()` of primitive event `EVTSITSTAIND` with event type `CIRRESREQLOC`.
5. Expiry action "initiate release procedure" or "release all equipment and connection" is sent to the application by `isup_get_msg()` of primitive event `EVTSITRELIND` after the Release (REL) message has been sent. To complete the connection release operation, the application should always respond with `isup_put_msg()` of primitive event `EVTSITRELRSP`.
6. Expiry action "alert maintenance personnel" is sent to the application by `isup_get_msg()` of primitive event `EVTLSISTAIND` with the event type `CIRMGT_NORESP`.
7. Timer expiry is handled the same as timer T5; see Notes 3 and 4.
8. Expiry action "alert maintenance personnel" is sent to the application by `isup_get_msg()` of primitive event `EVTSISTAIND` with event type `COR_VAL_FAIL`.

9. TCRM expiry (which clears the connection to prepare for a reattempt), is sent to the application by the isup_get_msg() of primitive event EVTSITSTAIND with event REATTMEMP.
10. TEXM is not used in the current SINAP/SS7 ISUP software version.
11. TGRES is only used internally by the SINAP/SS7 ISUP software.
12. If the FRANCE1 version of ISUP services has been activated, then upon receipt of a Reset Circuit (RSC) message for a previously blocked circuit, the SINAP node clears ISUP timers IST14 and IST15 and changes the circuit state to “unblocked.”
13. If the FRANCE1 version of ISUP services has been activated and, upon receipt of an Initial Address (IAM) message with a continuity check indicator set to 2 (continuity check on previous circuit required), and receipt of a Continuity (COT) with failure message, the SINAP node performs the following tasks:
    - Stops timer IST8
    - Stops timer IST27
    - Waits for the Continuity Check Request (CCR) message
    - If a Continuity (COT) message with failure is received, the SINAP node stops timer IST8 but does not start timer IST27 (the node does not perform the continuity recheck procedure) and sends a Release (REL) message with Loc=4 and cause=31 to the remote point code.

Handling RSC Messages

The default action of the SINAP node after receiving a Reset Circuit (RSC) message is to respond immediately by sending a Blocking (BLO) message followed by a Release Complete (RLC) message. The node starts timers T12 and T13 when it sends the first BLO message, but does not start the timers when the node sends a BLO message in response to the RSC message received. The BLO message handling based on the T12 and T13 timeouts continues per the standard.

In the ANSI version of ISUP services you can enable an environment variable that allows the SINAP node to return a BLO message immediately after receiving the initial RSC message but handle the sending of further BLO messages based on timer T12 and T13 time-outs.

The SINAP node initiates the sequence by sending a BLO message and starting T12 and T13. The remote switch responds with an RSC message.

1. The SINAP node always sends a RLC message in response to the RSC message.
2. When T12 expires, SINAP/SS7 sends another BLO message and restarts T12.
3. When T13 expires, the node stops T12 and uses T13 as the new schedule for sending BLO messages.

To enable this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) **before** starting or restarting the SINAP node.

`ISUP_RSC_BLO_PER_EXP`

NOTE

If you are running the MultiStack product, you must set the environment variable separately for each node on which you want to activate this feature.

Enabling the Generic Name Parameter in IAM Messages (ANSI)

In the ANSI version of ISUP services, the Generic Name parameter is not used in Initial Address (IAM) messages. However, the SINAP/SS7 system provides an optional feature that enables you to use the Generic Name parameter in IAMs.

To enable this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) **before** starting or restarting the SINAP node.

`ISUP_GENERIC_NAME`

Enabling this variable will define the Generic Name parameter with a code of `11000111` or `0xc7`.

NOTE

If you are running the MultiStack product, you must set the environment variable separately for each node where you want to activate this feature.

If you do not define this environment variable, the SINAP node treats the parameter as *unrecognized* and discards it.

Handling Queries for Unequipped Circuits (ANSI)

The following sections contain configuration options that you can enable in the ANSI version of ISUP services by defining environment variables. These options **do not** comply with the ANSI standards. You should carefully consider use of these options before implementing them.

Each section provides an explanation of the optional feature and how that feature is implemented by the SINAP/SS7 system.

Reporting Transient States in CQR Messages for Unequipped Circuits

The ANSI version of ISUP services provides an optional feature that causes the SINAP node to report a *transient* state for unequipped circuits in Circuit Query Response (CQR) messages, but prevents the unequipped circuits from actually going into the transient, maintenance state during certain types of outage conditions (for example, when the ISUP services application or its process manager is not running). This action applies to all unequipped circuits, not just those managed by the application.

CAUTION

The actions of the SINAP node resulting from implementation of this feature do not comply with the ANSI standards. Stratus only provides this feature for special case-handling of some 5ESS switches.

To enable this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) **before** starting or restarting the SINAP node.

```
ISUP_CQR_TRANS_FOR_UCIC
```

NOTE

If you are running the MultiStack product, you must set the environment variable separately for each node where you want to activate this feature.

When this variable is not defined, the SINAP node reports that the state *unequipped* in a CQR message for all unequipped circuits that are queried. This applies to all unequipped circuits, not just those managed by the application.

Prohibiting UCIC Reply Messages from Being Sent to Remote Nodes

The ANSI version of ISUP services provides an optional feature that prevents the SINAP node from responding to messages destined for unequipped or unconfigured circuits with

Unequipped Circuit Identification Code (UCIC) messages. Normally, the SINAP node sends UCIC response messages to the remote node.

CAUTION

The actions of the SINAP node resulting from implementation of this feature do not comply the ANSI standards. Stratus only provides this feature for special case-handling of some 5ESS switches.

To enable this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) **before** starting or restarting the SINAP node.

```
ISUP_NO_UCIC_REPLIES
```

NOTE

If you are running the MultiStack product, you must set the environment variable separately for each node on which you want to activate this feature.

When this variable is not defined, the SINAP node responds to queries for unequipped circuits with UCIC messages. This action affects *all* unequipped circuits, not just those managed by the application.

Chapter 3
Designing and Developing ISUP Services Applications

This chapter provides background information for programmer's responsible for developing ISUP applications that run on SINAP/SS7 and interact with an SS7 network. It contains the following sections:

- "ISUP Services Application Processing Overview"
- "Design Considerations"
- "Summary of ISUP Services Application Processing"
- "Registering with the SINAP Node"
- "Configuring an ISUP Services Application"
- "Performing a Circuit Query Test"
- "Initiating a Connection Request"
- "Accepting a Connection Request"
- "Sending and Receiving ISUP Data"
- "Initiating a Connection Release"
- "Accepting a Connection-Release Request"
- "Sending a SINAP IPC Message"

ISUP Services Application Processing Overview

This section describes the processing required to support communication between an ISUP application running on a SINAP node (a *local application*) and an ISUP application running on a remote system (a *remote application*). The applications communicate with each other by exchanging ISUP primitives and messages. The sequence and type of primitives and messages determine the task being performed. For example, to establish a voice-circuit connection, a local ISUP application first sends an ISUP Initial Address Message (IAM) to the remote application. The remote application responds by sending back an Address Complete (ACM) message and an Answer (ANM) message. This message exchange or *call flow* establishes a connection for a Circuit Identification Code (CIC) between the local and remote systems (a *voice-circuit connection*).

To initiate a particular task, the local application calls the ISUP services function `isup_put_msg()` with an initialized `isup_msg_t` structure. The `isup_msg_t` structure is defined in the `issl.h` include file and is described in the section “ISUP Services Primitive Types and Structures” in Chapter 4. The local application defines the task to be performed by specifying a particular ISUP services primitive in the `prim_ev` field of the hdr structure, which is part of the `isup_msg_t` structure. Each primitive type defines a different type of task (for example, connection setup or connection release). In addition, the local application must initialize the event structure that corresponds to that primitive type. The *event structure* provides the information necessary to perform the specified task. The descriptions of the ISUP services functions `isup_get_msg()` and `isup_put_msg()` in the section “ISUP Services Functions” in Chapter 4 contain tables that list primitive types and their corresponding event structures.

The SINAP/SS7 implementation of ISUP services contains internal software that automatically generates the appropriate ISUP message for the task defined by a particular primitive. Likewise, the ISUP services internal software retrieves each incoming ISUP message and generates the appropriate ISUP primitive for that message type, which it then passes to the local application. For example, to initiate a connection-setup request, the local application initializes the `isup_msg_t.hdr` structure’s `prim_ev` field to the value `EVTSITCONREQ` (which is the connection-setup request primitive). The application also initializes the event structure `SiConEvnt` so it contains the necessary information to establish the connection. The internal ISUP software then generates an Initial Address Message (IAM) and sends it to the remote application.

NOTE

When sending an ISUP message, the local application must initialize the appropriate event structure fields to define the mandatory parameters required for that message type. For more information on mandatory parameters for ISUP mesages, see the appropriate standards for the version of ISUP services implemented at your site.

Design Considerations

You should consider the following issues as you design and develop applications that use the SINAP/SS7 ISUP services.

- An ISUP services application that requires SS7 services (that is, an application that will process data from an SS7 network) **must** follow the ISUP registration rules that are described in the section ‘‘Registering with the SINAP Node” later in this chapter. An ISUP application that does not require SS7 services (for example, a Service Node application) must register with the SINAP/SS7 node with its CA_REG variable’s `fss7` field set to the value `0`. None of the other ISUP registration rules then apply.

- Only one ISUP SS7 application can run on a SINAP/SS7 node at any one time; however, the ISUP SS7 application can consist of up to 15 individual user processes, each functioning as a separate application instance.

  The ISUP services manager registers as an ISUP application requiring SS7 services; therefore, you must define your ISUP SS7 logic as one or more user processes (application instances) that run as part of the ISUP manager application. Normally, a SINAP/SS7 application can consist of up to 16 application instances that handle data primitives. However, since the ISUP manager handles both control and data primitives, your ISUP SS7 logic is limited to 15 application instances that handle data primitives.

- Each ISUP services application user process must register separately with the SINAP node. In addition, each user process must define its own ISUP configuration (that is, the remote SSPs and circuits it will manage) by following the instructions contained in the section "Configuring an ISUP Services Application" later in this chapter. Note that the ISUP services manager is automatically registered with the SINAP application if the ISUP services feature is active.

  Throughout this description, the term *application* refers to either an application or an individual user process.

- An ISUP services application must contain calls to the functions in the ISUP Services Support Library (ISSL) as described in Chapter 4 "ISUP Services Reference." The section "Summary of ISUP Services Application Processing" later in this chapter describes the basic tasks an ISUP application must perform.
- An ISUP services application must contain the following include statement, which incorporates all of the include files necessary for the application to use the ISSL.

  ```
  #include <issl.h>
  ```

- When configuring a circuit, the near and far ends of the circuit must be in compatible states. See the section "Configuring an ISUP Services Application" later in this chapter for more information.
- To send a SINAP Interprocess Communications (IPC) message, an ISUP application must initialize the `isup_msg_t.hdr` structure's `prim_ev` field to the value `0` and initialize an `i_block_t` structure to define the IPC message. See the section "Sending a SINAP IPC Message" later in this chapter for more information.
- To enable L3MT to respond to whatever application process is issuing the network management request, store the `orig_id` field of the received IPC message in the `dest_id` field of the outgoing IPC message.
- When a SINAP/SS7 application registers with CASL, CASL checks the value of the maximum MSU input queue size (defined in the parameter `CA_REG.max_msu_input_que`) to determine whether the value falls within the input queue range of `7000` through `32000` (defined in the `sinap.h` include file in `MAX_MSU_INPUT_QUEUE`). If the value is greater than this limit, CASL rejects the registration and issues the error `CA_ERR_REG_MAX_INPUT_QUEUE (2109)`.

NOTE

CASL error messages are listed and described by number in the appendix "CASL Error Messages" in the *SINAP/SS7 Programmer's Guide* (R8052). ISUP services error messages are described in the appendix "ISUP Services Error Messages" at the end of this manual.

The SINAP driver keeps a running total of the MSU input count for all data processes that are already registered with CASL on the same system. The driver rejects the registration request and issues the error:

```
CA_ERR_REG_MAX_INPUT_QUEUE (2109)
```

If the requested queue size times the `(sizeof(m_block_t) + sizeof(msgb) + sizeof(datab)~= 500` bytes plus the running driver total of the MSU input count for all applications exceeds half the value of the `strthresh` parameter `strthresh/2`).

Note that the `strthresh` parameter is a tunable system parameter that defines the maximum amount of STREAMS memory that can be allocated. STREAMS memory is a limited resource, since it is tied to real memory, not virtual memory. You may need to adjust the STREAMS threshold if you are running multiple applications on the same system.

Make sure the total input queue count of all processes registered on the machine does not result in a number that is greater than half the value defined for the STREAMS memory threshold (`strthresh`). A `max_msu_input_que` value of `6000` or less is likely to avoid a registration error for a single application process.

- For the ANSI version of ISUP services, `isup_cfg_circuit_info()` function allows an ISUP services application to modify one or more default values of the Circuit Group Characteristic Indicator parameter contained in a Circuit Validation Response (CVR) message which is sent in response to a Circuit Validation Test (CVT) message. This function provides application control over the content of the circuit group characteristic indicator on a circuit-by-circuit basis.

NOTE

If the ISUP user application has defined a value for the Double Seizing Control Indicator field of the Circuit Group Characteristics Indicator parameter through the environment variable `ISUP_DBL_SEIZE_BITS`, using the `isup_cfg_circuit_info()` function to define the double seizing control indicator overrides the value specified by the environment variable.

Calling the `isup_cfg_circuit_info()` function is optional. If your application does not call this function, the `isup_add_circuit()` function automatically populates the circuit group characteristic indicator in the circuit configuration structure with the default values.

- ISUP supports user timers for ISUP-related events at the application layer, such as waiting for additional dialed digits from exchanges that initiate ISUP connections with incomplete addresses according to country-specific numbering/dialing schemes. The following functions are used to start and stop user timers:

  `isup_start_utimer()`

  `isup_stop_utimer()`

- In the ANSI version of ISUP services, you can set an environment variable that enables the SINAP node to return a Blocking (BLO) message immediately after receiving the Reset Circuit (RSC) message and handle the sending of BLO messages based on timer T12 and T13 timeouts.

  To activate this feature, uncomment the following environment variable in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.`[`csh` or `sh`]) **before** starting or restarting the SINAP node.

  `ISUP_RSC_BLO_PER_EXP`

  If this environment variable is not set, the default action is to respond immediately, without delay.

- Country-Specific ISUP Services Versions

  The SINAP/SS7 system is able to support a number of country-specific versions of ISUP services by defining the `ISUP_FEATURE` environment variable. Refer to the section “Activating the ISUP Services Feature” in the previous chapter for more information.

- Upon receiving an ISUP Initial Address Message (IAM) with a continuity check indicator set to `2` (continuity check on previous circuit required) and receipt of a Continuity (COT) with failure message, the SINAP node performs the following tasks:

  —Stops timer IST8

  —Starts timer IST27 and waits for the Continuity Check Request (CCR) message

  —If a Continuity (COT) message with failure is received, the SINAP node stops timer IST8 but does not start timer IST27 (the node does not perform the continuity recheck procedure). It also sends a Release (REL) message with `Loc=4` and `cause=31` to the remote point code

- The SINAP software discards an unrecognized message if the circuit state is `IDLE` and the message has no optional parameters.

If the circuit state of the unrecognized message is `BUSY` with no optional parameters, the SINAP software sends a Confusion (CFN) message.

Summary of ISUP Services Application Processing

The following list describes the tasks a SINAP/SS7 application (or user process) must perform to implement ISUP services feature. See Chapter 4 “ISUP Services Reference” for a description of the ISSL functions listed here.

1. The application registers for ISUP services by calling the CASL `ca_register()` function.
2. The application defines its configuration limitations by calling the ISSL `isup_cfg_process()` function.
3. The application defines the active remote SSPs it will manage by calling the ISSL `isup_add_rem_ssp()` function once for each configured remote SSP.
4. If the Distributed Logical Point Code (DLPC) feature is configured on the SINAP node, the application defines the standby remote SSPs it will manage by calling the ISSL `isup_add_standby_rem_ssp()` function once for each remote SSP.
5. The application defines the circuits it will manage by calling the ISSL `isup_add_circuit()` function once for each circuit.
6. The application performs a circuit query to audit the status of both the near and far ends and correct any discrepancies.
7. The application can begin passing and receiving ISUP messages. Call the `isup_put_msg()` function to send an outgoing message, and call the `isup_get_msg()` function to retrieve an incoming message.

   Use the functions `isup_alloc_msg()` and `isup_free_msg()`, respectively, to allocate and deallocate memory for an ISUP message.

   - The `isup_alloc_msg()` function allocates a structure for an outgoing ISUP message. Before calling `isup_put_msg()` to send an outgoing message, call `isup_alloc_msg()` to allocate a structure for the message. The structure is automatically freed when you call `isup_put_msg()` to send the message. Note that you need not call `isup_alloc_msg()` if you are reusing an incoming ISUP message structure.
   - The `isup_free_msg()` function deallocates an ISUP message structure. After calling `isup_get_msg()` to retrieve an incoming ISUP message, call `isup_free_msg()` to free the message structure. Note that you need not call `isup_free_msg()` if you plan to reuse the message structure to send an outgoing response message.
8. The ISUP services sample programs `isup_recv.c` and `isup_send.c` located in the directories `$SINAP_HOME/Samples/ccitt` and `$SINAP_HOME/Samples/ansi`, respectively, show how to use the ISUP services API

to send and receive ISUP messages. These sample programs also allow an ISUP services application to request the current state and status of an individual circuit or CIC using the `isup_get_circuit_status()` function. The states for the selected CIC are displayed on the console. You can modify the `isup_send.c` and `isup_recv.c` programs to display additional states and statuses if necessary. You can also request the circuit status using the command line parameter `-q` which activates the `isup_get_circuit_status()` function. The default is no circuit status is to be printed.

When the DLPC feature is configured on a SINAP node, you can use the `-L` argument in the `isup_send.c` program to specify the logical point code to be registered with CASL. When the `-L` argument is present, any messages originating from `isup_send.c` are sent with the originating point code (OPC) set to equal the specified logical point code (LPC). When the `-L` argument is not present, the LPC is set to 0 (the default) which specifies that `isup_send.c` originates messages using the node's own signaling point code (OSP).

NOTE

These sample programs do not contain the logic required to perform call-control processing. For information on how to design such an application, see the appropriate standards for the version of ISUP services implemented at your site.

Registering with the SINAP Node

As described in the section ‘‘Design Considerations,” your ISUP SS7 logic must run as one or more user processes (up to 15) of the ISUP services manager application. When registering with the SINAP/SS7 node, each user process (or application instance) must assign the values listed in Table 3-1 to the following `CA_REG fields`. (`CA_REG` is the global variable for the registration structure, `register_req_t`, in which the application's operating characteristics are stored. The `register_req_t` structure is defined in the `register.h` include file, and the `CA_REG` variable is defined in the `sinapintf.h` include file.) The application also must assign values to the other `CA_REG` fields, which are described in the `ca_register()` function's `register_req_t` structure description contained in Chapter 4.

In the column “Required Value for ISUP Registration,” the number in parentheses is the numerical equivalent of the text value. You can specify either value for the structure field.

Table 3-1. ISUP Services Application Registration Values

`CA_REG` field	Required Value for ISUP Registration
`appl`	The string “`IS`” enclosed in quotation marks.
`fsignal`	`IPC_NOTIFY_WITHOUT_SIGNAL (2)`

Table 3-1. ISUP Services Application Registration Values *(Continued)*

`CA_REG` field	**Required Value for ISUP Registration**
`fss7`	If your ISUP services application requires SS7 services, set this field to `1`; otherwise, set it to `0`.
`inbound_load_dist_type`	`ISUP_REMOTE_SSP (4)`
`proc`	A string of up to four characters, enclosed in quotation marks. You cannot use the string "`MG`", which is reserved for use by the ISUP manager. If your ISUP SS7 logic consists of multiple user processes, each user process must use the same value for this field.
`sio_ssn`	`L_ISDNUP_SI (5)`
`sio_ssn_ind`	`REG_SIO (1)`
`ss7_input_boundary`	`SS7_INPUT_BOUNDARY_ISUP (5)`
`ss7_primitive`	`SS7_DATA_PRIMITIVE (2)`

After registering with the SINAP node, an ISUP application must define its configuration, as described in the following section, "Configuring an ISUP Services Application."

Configuring an ISUP Services Application

The following list describes the steps that an ISUP services application must perform in order to define its configuration (that is, the remote SSPs and circuits that it will manage). Note that the `sinap.h` include file defines the maximum number of components that an application can manage.

1. The application defines its maximum configuration limits and identifies itself to the ISUP manager by calling the `isup_cfg_process()` function, as shown in the following example.

   ```
   isup_cfg_process(10, 12, 150);
   ```

   The preceding function call sets the application's configuration limits as follows: 10 remote SSPs, 12 circuit groups, and 150 circuits.

2. The application defines the active remote SSPs that it will manage by calling the `isup_add_rem_ssp()` function, as shown in the following example.

   ```
   isup_add_rem_ssp(3003);
   ```

   The preceding function call configures the point code `3003` as one of the remote SSPs that this application will manage.

3. If the DLPC feature is configured on an associated SINAP node, the user application must also define on that node or the other node the standby remote SSPs it will manage by calling the `isup_add_standby_rem_ssp()` function as shown in the following example.

   ```
   isup_add_standby_rem_ssp(3004)
   ```

4. The preceding function call configures the point code 3004 as one of the standby remote SSPs this application will manage.

5. The application defines the circuits it will manage by calling the `isup_add_circuit()` function, as shown in the following example.

   ```
   isup_add_circuit(25, 55, 3003);
   ```

   The preceding function call configures a circuit with a circuit ID of `25` and a circuit identification code of `55`, and indicates that the circuit connects to the remote SSP whose point code is `3003`.

6. Before an application can perform any protocol operations for a circuit, it must configure a circuit using the `isup_add_circuit()` function.

Performing a Circuit Query Test

After configuring a circuit, the ISUP Services protocol performs a circuit query test. The near and far ends of the circuit must be in compatible states. During a circuit query test, the ISUP services protocol takes certain corrective actions, if necessary, to resolve any discrepancies if the near and far ends of the circuit are in incompatible states.

The near-end initiates a circuit query test by sending a Circuit Group Query (CQM) message to the far-end. The far-end responds with a Circuit Query Response (CQR) message that contains the far-end circuit state data. When the near-end receives the CQR, it uses its own circuit state data and the circuit state data from the far-end to audit the circuit state and correct any discrepancies.

Table 3-2 describes the possible states that can occur at the near-end and far-end of the circuit, corrective actions the ISUP services protocol takes to align the two states when discrepancies exist, and the actions the application should take.

Table 3-2. Correcting Circuit Discrepancies

Near-end and Far-end Circuit States	Corrective Action
Near-end maintenance blocking state is *locally blocked* Far-end state should be *remotely blocked*	When the near-end state disagrees with the far-end state, the ISUP services protocol software internally sets the maintenance blocking state to show the near-end is not locally blocked. The application should initiate a circuit-blocking operation to impose its will on the far-end.
Near-end or far-end is *transient*	No action required.
Far-end is *unequipped;* disagrees with near-end state (near-end is equipped)	The ISUP services protocol software internally marks the circuit *locally blocked* (not remotely blocked). The application should detect a circuit state indicator value of *unequipped*. If the circuit is busy with a call processing connection, the application should idle the physical circuit and release any resources. It can free the connection with the ISUP services library by calling the `isup_put_msg()` API function for a connection release request.
Near-end is *idle* Far-end should also be idle (if not, corrective action is required)	The ISUP services protocol software internally marks the call processing state *busy* according to the far-end's call processing state. However, since no connection actually exists, the ISUP protocol responds negatively, as appropriate, to incoming messages related to a connection on the circuit. The application can choose to initiate a circuit reset operation to release the connection more quickly.
Near-end call processing state is *incoming busy*; far-end should be *outgoing busy* Near-end call processing state is outgoing busy; far-end should be incoming busy	The ISUP services protocol software internally changes the call processing state when the two states do not agree, but the connection should not be expected to proceed in an orderly fashion when such disagreement on its state exists. The application should call `isup_put_msg()` for a connection release request. Upon completion of the request, the application should idle the physical circuit and release any resources.

Table 3-2. Correcting Circuit Discrepancies *(Continued)*

Near-end and Far-end Circuit States	Corrective Action
Near-end maintenance blocking state is *locally blocked* Far-end state should be *remotely blocked*	When the near-end state disagrees with the far-end state, the ISUP services protocol software internally sets the maintenance blocking state to show the near-end is not locally blocked. The application should initiate a circuit-blocking operation to impose its will on the far-end.
Near-end maintenance blocking state is set to `unblocked (active)` Far-end state should also be unblocked	When the states disagree and the discrepancy includes the far-end being *remotely blocked*, the ISUP services protocol software internally sets the maintenance blocking state to show that the near-end is *locally blocked*. The application should initiate a circuit unblocking operation to impose its will on the far-end. If the discrepancy includes the far-end being *locally blocked*, the ISUP services protocol internally sets the maintenance blocking state to show that the near-end is *remotely blocked*. The application should note the change the far-end imposed, since no local protocol action can undo this change.
Near-end maintenance blocking state is *remotely blocked* Far-end maintenance blocking state should be *locally blocked*	When the near-end state disagrees with the far-end state, the ISUP services protocol software internally sets the maintenance blocking state to show the near-end is not remotely blocked. The application should note the change the far-end imposed since no local protocol can undo this change.

Initiating a Connection Request

Figure 3-1 shows a typical call flow for initiating a connection request in order to establish a connection. This call flow is summarized as follows:

- The local application calls `isup_put_msg()` to output an ISUP message whose primitive type is `EVTSITCONREQ`. This primitive tells the internal ISUP software to initiate a connection-setup request. The local application also initializes the `SiConEvnt` structure with the information needed to generate an Initial Address Message (IAM).
- Using the information in the `SiConEvnt` structure, the internal ISUP services software generates an IAM, which it sends to the remote application.

- The remote application responds by generating one or more of the messages shown in the following chart, which the internal ISUP services software translates into the appropriate primitive. The local application receives the primitive when it calls the `isup_get_msg()` function. (In Table 3-3, note that the column labeled "Event Type" indicates the value of the `isup_msg_t.hdr` structure's `ev_type` field, which defines the type of ISUP message received.)

Table 3-3. ISUP Messages, Primitives, and Event Types

ISUP Message Generated by the Remote Application	Primitive Received by the Local Application	Event Type
Information Request (INR) †	`EVTSITCNSTIND`	`INFORMATREQ`
Address Complete (ACM)	`EVTSITCNSTIND`	`ADDRCMPLT`
Call Progress (CPG)	`EVTSITCNSTIND`	`PROGRESS`
Connect (CON), if no ACM (not applicable to ANSI)	`EVTSITCONCFM`	
Answer (ANM), follows Address Complete (ACM)	`EVTSITCONCFM`	

† Upon receipt of an Information Request (INR) message, the local application should call `isup_put_msg()` to send an Information (INF) message to the remote application.

NOTE

The local application may receive status messages not listed above. For example, the application might receive a Confusion (CFN) message, which might indicate that the remote application could not handle the local application's Initial Address Message (IAM) due to invalid configuration data, such as an unrecognized CIC.

- When the connection is established, the local application will receive an ISUP message whose primitive type is `EVTSITCONFCM`. This primitive type confirms that connection setup is complete.

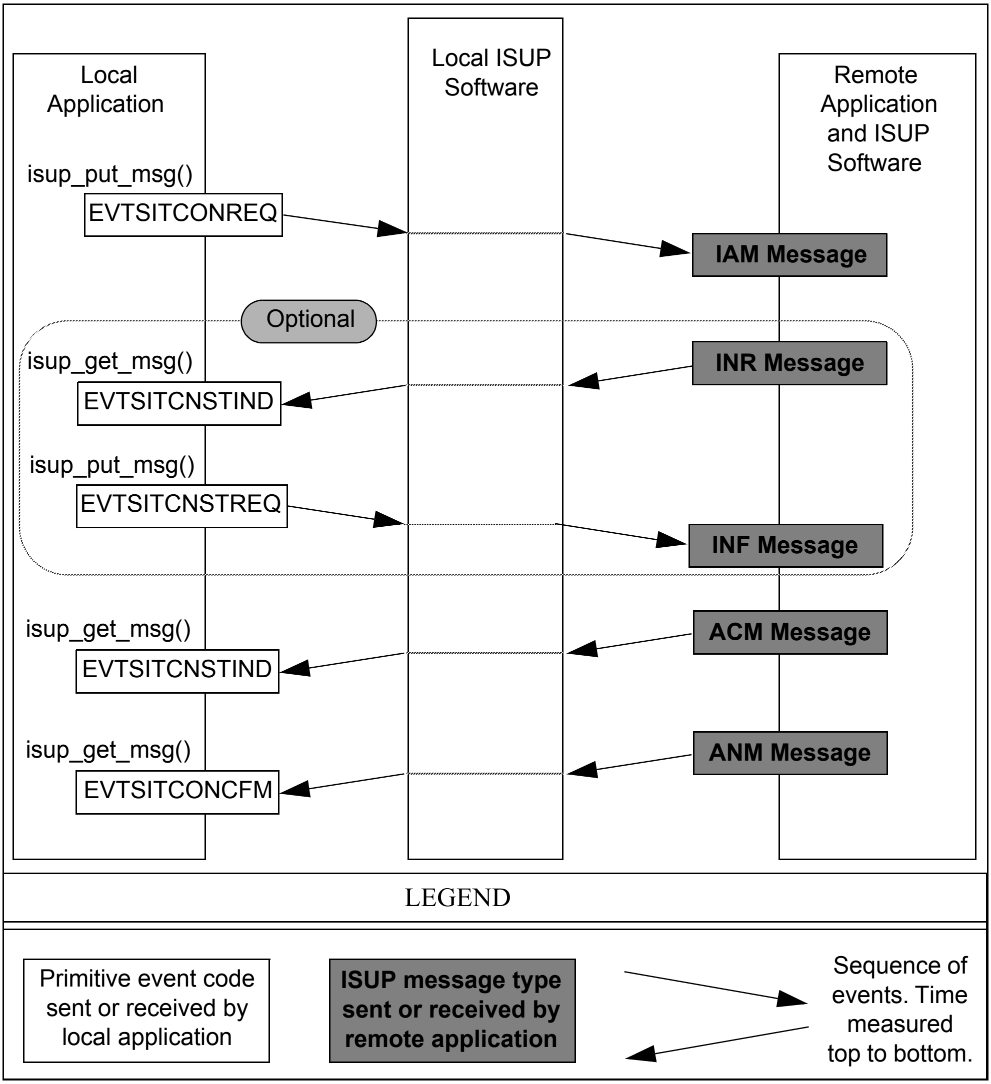

Figure 3-1. Typical Call Flow for Initiating a Connection Request

The preceding example call flow does not contain all logic normally required in a typical application. For more information about the necessary application logic, see ITU-T Recommendation Q.764, Annex H or ANSI Standard T1.113.4. For more information on ISUP services primitive event codes and ISUP message types, see the appendix “ISUP Services Messages and Parameters” at the end of this manual.

Accepting a Connection Request

Figure 3-2 shows a typical call flow for accepting a connection request. This call flow is summarized as follows:

- The remote application sends an Initial Address Message (IAM) to the local application.
- The local application calls `isup_get_msg()` to input the IAM, which the internal ISUP software has translated into an ISUP message whose primitive type is `EVTSITCONIND`. (This primitive type tells the local application that the remote application wants to establish a connection.) The local application evaluates the `SiConEvnt` structure to obtain information about the connection request.
- The local application responds to the IAM by generating one or more of the following messages, whose primitive type is one of those listed in the following chart. To define the type of ISUP message to send, the local application initializes the `isup_msg_t.hdr` structure's `ev_type` field to the value listed in Table 3-4's "Event Type" column. In addition, the local application initializes the structure that corresponds to the specified primitive. The internal ISUP software uses the information in the structure to generate the appropriate ISUP message, which it then sends to the remote application.

Table 3-4. Primitive Sent and Message Received

Primitive Sent by the Local Application	ISUP Message Received by the Remote Application	Event Type
`EVTSITCNSTREQ`	Information Request (INR) †	`INFORMATREQ`
`EVTSITCNSTREQ`	Address Complete (ACM)	`ADDRCMPLT`
`EVTSITCNSTREQ`	Call Progress (CPG)	`PROGRESS`
`EVTSITCONRSP`	Connect (CON) (not applicable to ANSI)	
`EVTSITCONRSP`	Answer (ANM)	

† After sending an Information Request (INR) message, the local application should call `isup_get_msg()` to retrieve the Information (INF) message that the remote application sends in response.

- The connection is established when the local application sends an ISUP message whose primitive type is `EVTSITCONRSP`. This primitive type is a positive response to the remote application's connection-setup attempt.

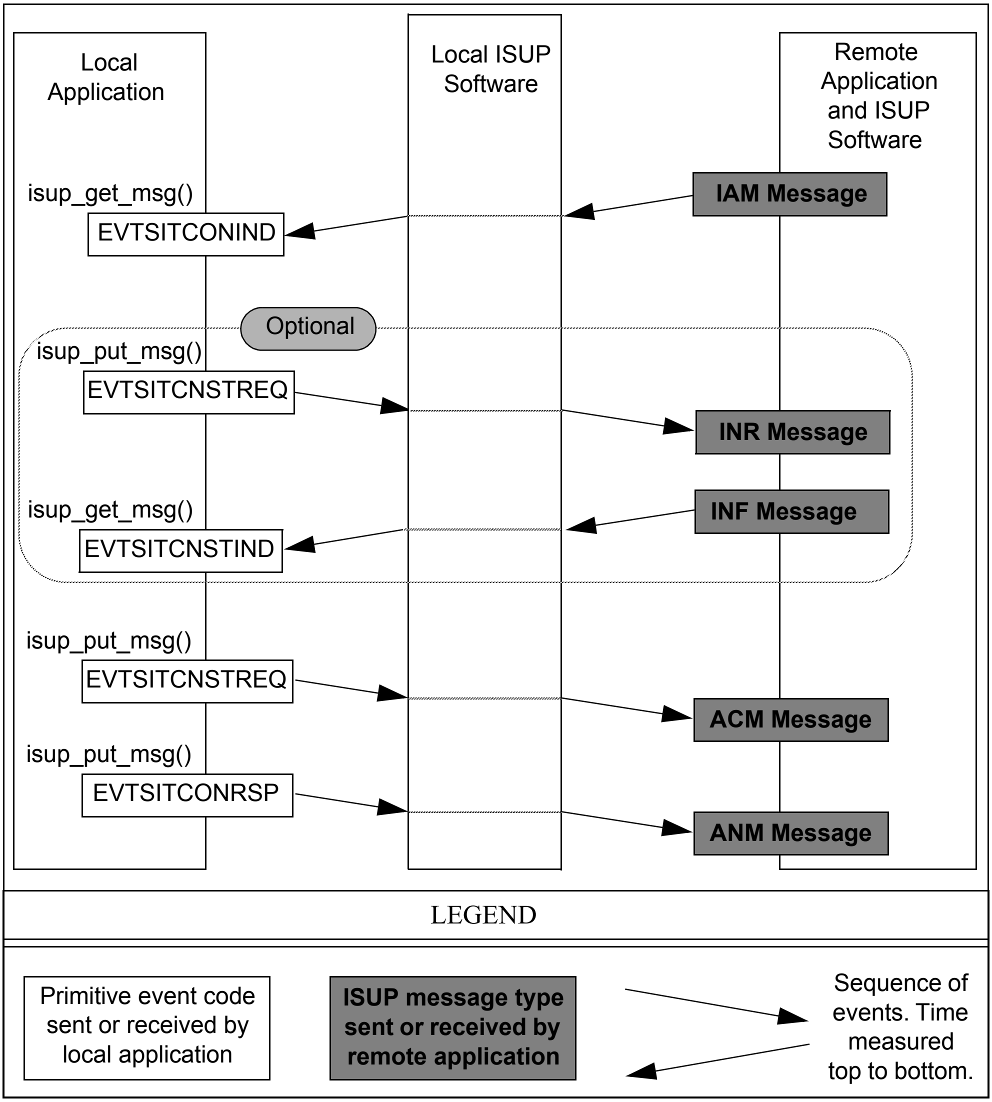

Figure 3-2. Typical Call Flow for Accepting a Connection Request

The preceding example call flow does not contain all logic normally required in a typical application. For information about the necessary application logic, see ITU-T Recommendation Q.764, Annex H or ANSI Standard T1.113.4. For more information on ISUP services primitive event codes and ISUP message types, see the appendix "ISUP Services Messages and Parameters" at the end of this manual.

Sending and Receiving ISUP Data

Figure 3-3 shows a typical call flow for sending and receiving ISUP data.

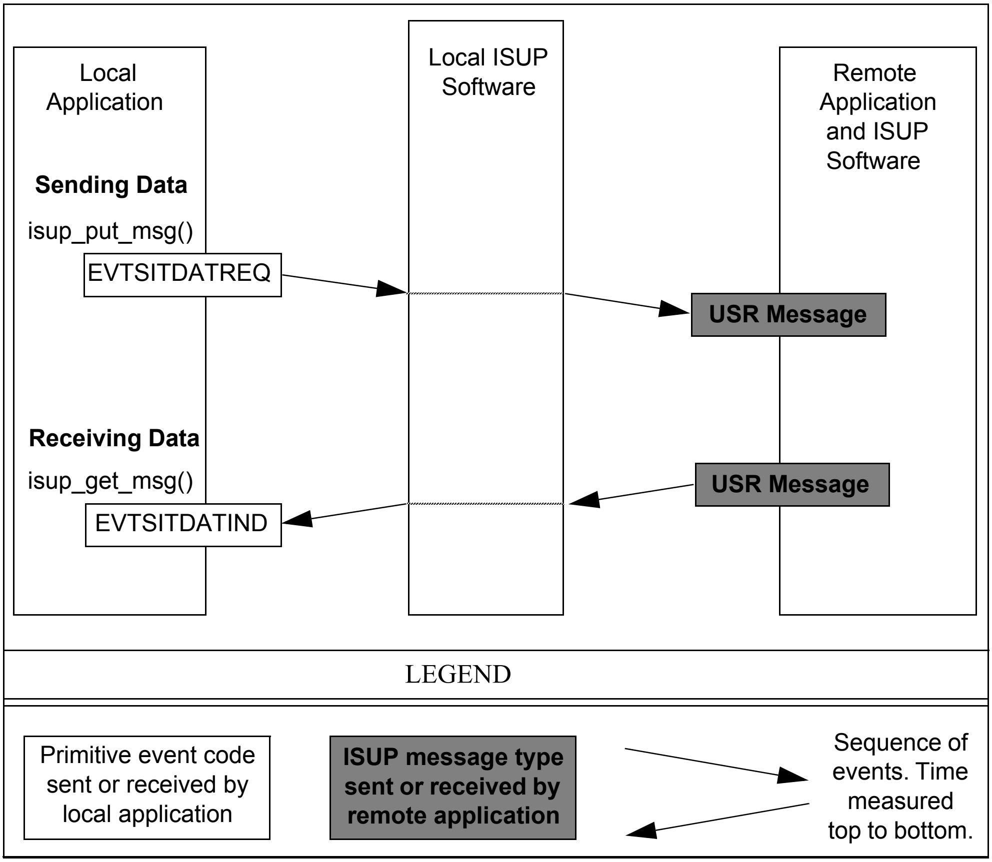

Figure 3-3. Typical Call Flow for Sending and Receiving ISUP Data

The preceding example call flow does not contain all logic normally required in a typical application. For information about the necessary application logic, see ITU-T Recommendation Q.764, Annex H or ANSI Standard T1.113.4. For more information on ISUP services primitive event codes and ISUP message types, see the appendix “ISUP Services Messages and Parameters” at the end of this manual.

Initiating a Connection Release

Figure 3-4 shows a typical call flow for terminating a connection with a remote application.

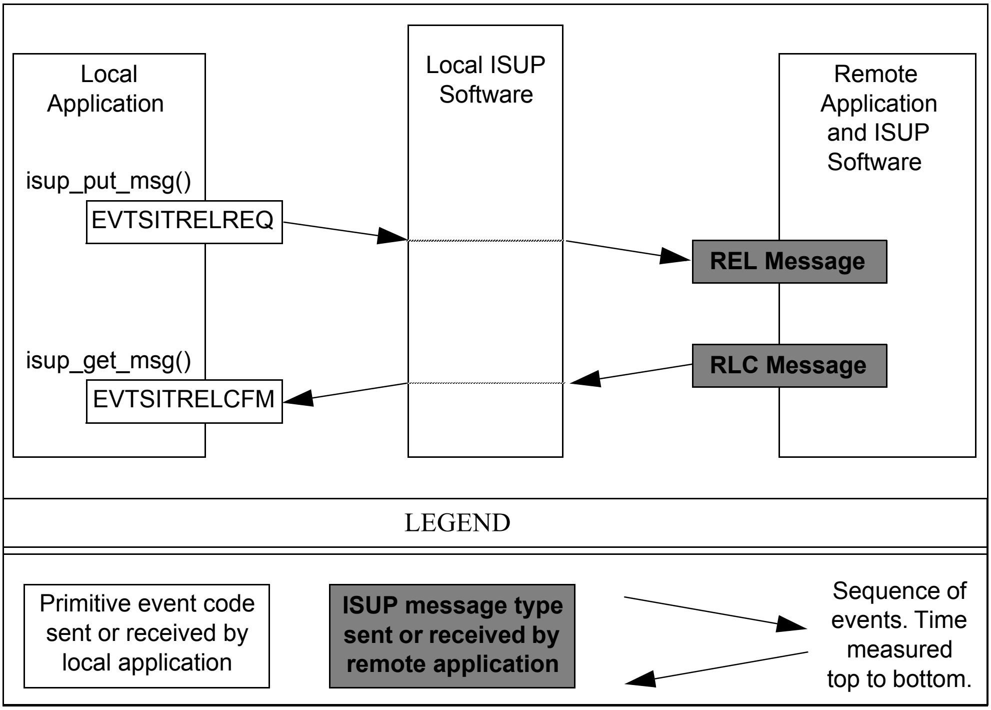

Figure 3-4. Typical Call Flow for Initiating a Connection Release

The preceding example call flow does not contain all logic normally required in a typical application. For information about the necessary application logic, see ITU-T Recommendation Q.764, Annex H or ANSI Standard T1.113.4. For more information on ISUP services primitive event codes and ISUP message types, see the appendix “ISUP Services Messages and Parameters” at the end of this manual.

Accepting a Connection-Release Request

Figure 3-5 shows a typical call flow for accepting a connection-release request. Accepting the request terminates the connection between the local and remote applications.

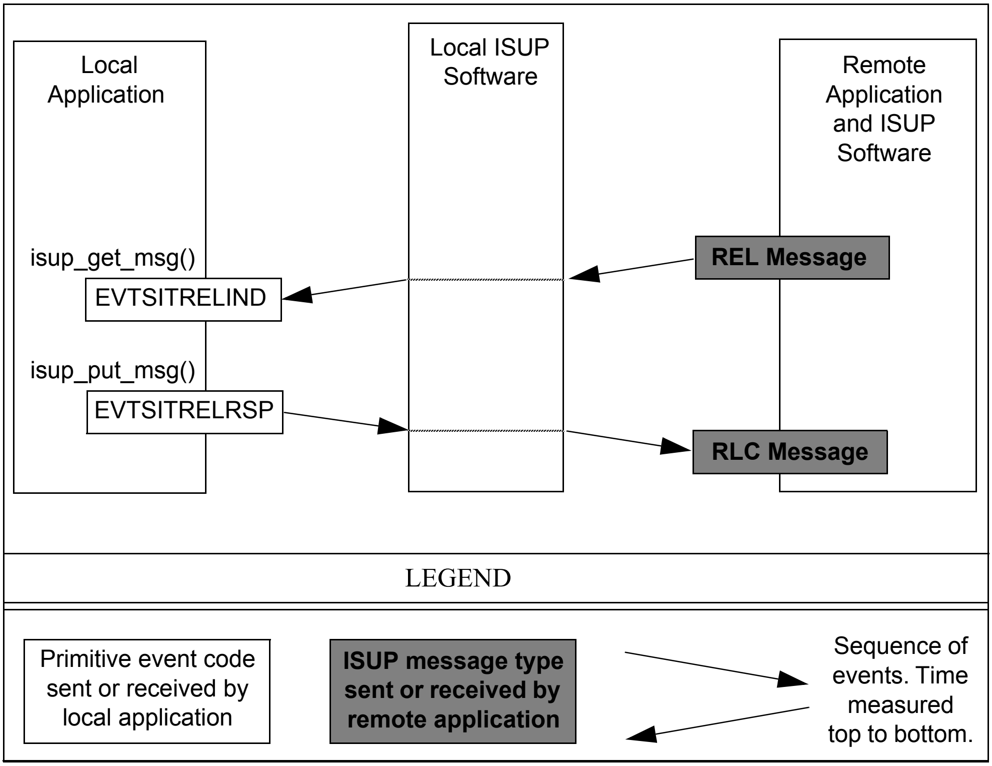

Figure 3-5. Typical Call Flow for Accepting a Connection-Release Request

The preceding example call flow does not contain all logic normally required in a typical application. For information about the necessary application logic, see ITU-T Recommendation Q.764, Annex H or ANSI Standard T1.113.4. For more information on ISUP services primitive event codes and ISUP message types, see the appendix “ISUP Services Messages and Parameters” at the end of this manual.

Sending a SINAP IPC Message

To send a SINAP Interprocess Communications (IPC) message, an ISUP services application must perform the following actions.

1. Initialize the `isup_msg_t.hdr` structure for the SINAP IPC message by performing **either** of the following steps.

   a. If you are allocating a new `isup_msg_t.hdr` structure, issue the following function call (where `0` is the primitive event for sending a SINAP IPC message). The function returns an `isup_msg_t.hdr` structure whose `prim_ev` field is set to the value `0`, which is the required value for sending a SINAP IPC message.

   ```
   isup_alloc_msg(0);
   ```

   b. If you are reusing an existing `isup_msg_t.hdr` structure, set the structure's `prim_ev` field to the value `0`, which indicates that you are sending a SINAP IPC message.

2. Initialize the `ipc` field of the `isup_msg_t.ev` pointer union to the address of an `i_block_t` structure that will contain the SINAP IPC message.
3. Define the SINAP IPC message header and IPC data by initializing the following fields in the `i_block_t` structure.
   – Specify the type of IPC message to send in the `ipc_trans_t` structure's `msg_type` field. The `ipc_trans_t` structure is part of the `i_block_t` structure.
   – Specify the IPC key of the process to which you want to send the message in the `i_block_t` structure's `dest_id` field. Use the CASL `ca_get_key()` function to obtain the IPC key.
   – Define the IPC message header in the `ipc_data_t` structure defined by the `i_block_t` structure's `msg` field.
   – Define the IPC message by following the example that appears after the `i_block_t` structure definition in the `iblock.h` include file.
4. Call the `fvc` function to send the SINAP IPC message to its destination.

NOTE

An ISUP application **must** use the ISSL functions `isup_get_msg()` and `isup_put_msg()` to send and retrieve IPC messages. The application **cannot** use the CASL functions `ca_get_msg()` and `ca_put_msg()` functions. If an application attempts to use these CASL functions, error(s) result.

Chapter 4

Implementing the Distributed Logical Point Code Feature

In the CCITT, ANSI, and China network variants the distributed logical point code (DLPC) feature supports distributed ISUP user applications on two SINAP nodes on different Continuum systems (or the same system). When the SINAP nodes are configured for the DLPC feature, up to 15 ISUP user applications per node can register for the same *logical point code* (LPC) by using a destination point code (DPC) other than the SINAP nodes' own signaling point code (OSP). From the SS7 network perspective, the LPCs appear to be behind a pair of signaling transfer points (STPs). The STPs are the DLPC-configured SINAP nodes executing on Continuum machines.

The ISUP user application registers with the SINAP node using a destination point code (DPC) other than the SINAP node's own signaling point code (OSP). The OSPs (configured via the CREATE-OSP command) on the SINAP nodes act as *logical point code routers* (LPCRs) that function like a pair of STPs. Note that although the DLPC-configured nodes are addressed by the network as an STP pair for purposes of routing ISUP messages to the LPCs, they do not route messages to other SS7 nodes and cannot forward messages to actual SS7 point codes behind them.

For incoming ISUP messages on each DLPC node, the originating point code (OPC) in the MTP routing label specifies the remote service switching point (SSP) that originated (or sent) the message. The destination point code (DPC) in the routing label identifies the logical point code that the ISUP application registered on the SINAP node. The SINAP driver along with the assistance of the ISUP manager (using the intermodule agent (IMA)) routes the message to the correct ISUP user application on either machine based on the following:

- SIO (service information octet)
- DPC (the LPC of the registered ISUP application)
- OPC (a remote SSP that the ISUP application configured)

In case a DLPC machine fails or is manually shutdown, the SS7 network routes all ISUP messages to the other DLPC machine. Note that the SINAP node network configurations on the two DLPC machines must match.

The ISUP user application can configure remote point codes (OPCs) in two modes:

- *Active* — The primary remote point code that processes incoming ISUP messages
- *Standby* — The secondary remote point code that is not active, but is ready to process ISUP messages in case the primary remote point code fails or a request occurs for the standby point code to take over processing

Major Components of the DLPC

The following major components that provide the DLPC functionality are:

- The Intermodule Agent (IMA) - A set of functions that route messages between Continuum machines when required using the TCP/IP network. When the IMA receives a message from the TCP/IP network, the routes the message to the correct ISUP user application (which can be on either machine).
- SINAP Driver - Accepts incoming messages for the SINAP node's own signaling point code (OSP) and for logical point codes (destination point codes (DPCs) specified in the message's MTP routing label) registered by the ISUP application. The driver routes SS7 messages to the ISUP user application based on the SIO, DPC and OPC specified in the messages.
- ISUP Manager (ISMG) - The process on each DLPC machine that manages the remote point codes configured by all ISUP applications. The ISUP manager notifies the SINAP driver of remote point codes configured by the ISUP user application(s). If the SINAP driver cannot find a registered ISUP user process for an incoming SS7 message, it sends the message to the ISUP manager (as done on a conventional SINAP node). When the ISUP manager receives this message, it sends the message via the local IMA to the peer IMA on the remote Continuum machine. When the peer IMA receives the message, it verifies the message against the node's local process table to determine if a process is running on that node that can handle the message. If the local ISUP user process is running, the IMA routes the message to the SINAP driver for distribution. If no local running ISUP user process exists, the ISUP manager responds with a message that indicates the incoming SS7 message cannot be handled on that machine.
- Existing ISUP User Applications - ISUP user applications configured on the SINAP node prior to configuring the DLPC feature. These applications have the option to register to receive messages for a destination point code other than the node's OSP. Note that if you choose not to use this option (that is, leave the registration parameter null), all messages destined for the node's OSP are routed to the application. In this case, the existing SINAP application remains unaffected by the DLPC functionality.

  Existing ISUP user applications also have the ability to register originating point codes (OPCs) in either *active* or *standby* mode. Point codes registered in active mode can send and receive messages from the SS7 network. However, point codes registered in standby mode cannot send or receive SS7 messages until the active ISUP user application fails (gives up the active status) or requests to change from active to standby mode. Then, the standby point becomes the active point code and automatically begins receiving SS7 messages.

- Common Application Services Layer (CASL) Library – The CASL library allows an ISUP user application to register a logical point code (LPC) on the SINAP node which enables the application to receive messages with a destination point code that is different from the SINAP node's OSP. The CASL also provides a method to reroute outbound SS7 messages to the remote Continuum machine when no route is available on the local Continuum machine.

Figure 4-1 illustrates DLPC functionality based on an example network in which point code 500 sends a message destined for point code 200. Note that the ISUP user application running on each Continuum machine handles a different remote point code. The message is routed through either of the SINAP nodes (point code 115 or 135) to the ISUP user application. If the ISUP user application is on the other machine, the ISUP manager receives the message and sends it to the IMA process for routing to the peer IMA process on the other Continuum machine. The receiving IMA sends the response message from point code 200 to point code 500.

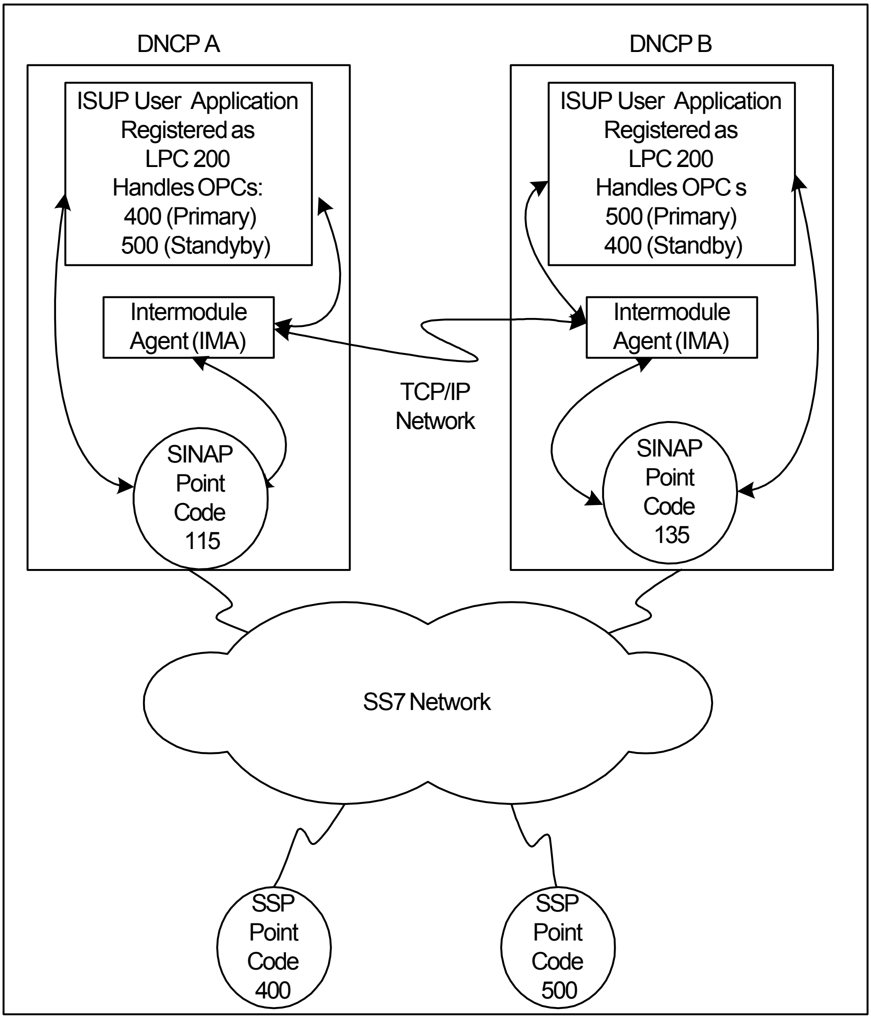

Figure 4-1. DLPC Functionality Overview

Figure 4-2 shows the processes that make up the DLPC machine pairs that are connected to the SS7 network.

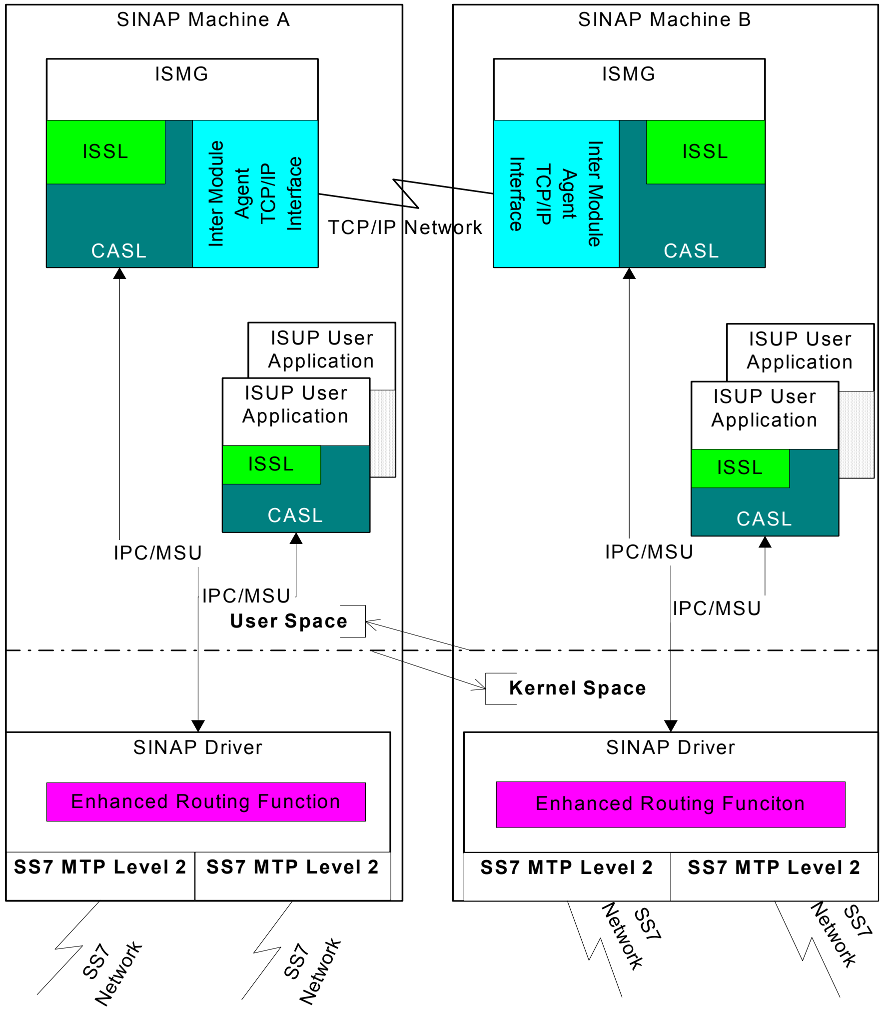

Figure 4-2. Major Components of the DLPC Feature

Figure 4-3 shows the SS7 ISUP message flow when both Continuum machines are operational. The network configuration for this figure would be similar to that shown in Figure 4-1. The ISUP user application running on each Continuum machine handles a different remote point code. SS7 messages flow to either machine and DLPC routes the message to the correct ISUP user application to allow for appropriate response to the message. Note that the string id.x (where x is a, b, or c) identifies messages a, b, and c to facilitate tracking the message flows.

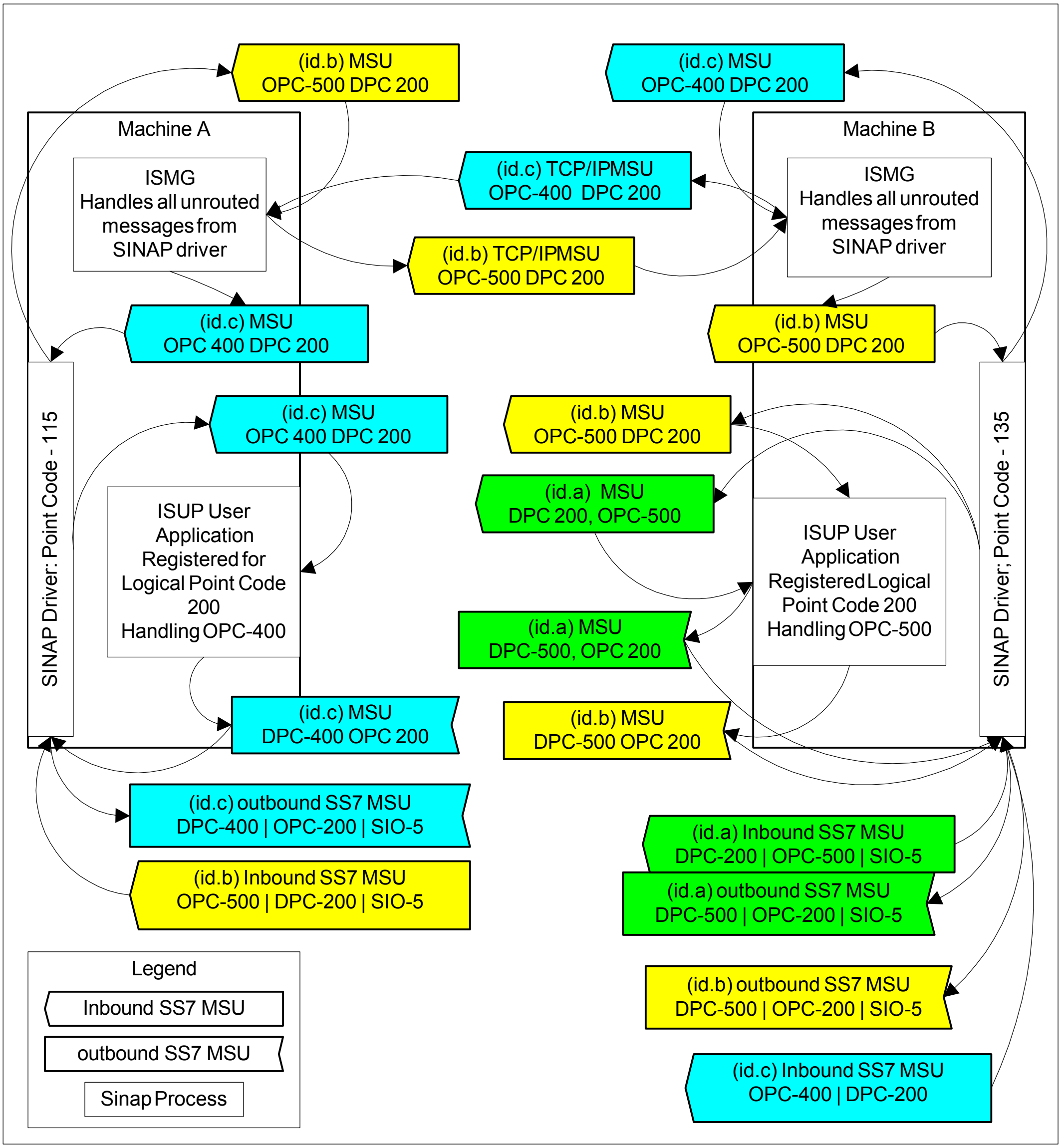

Figure 4-3. DLPC Message Flow

The SS7 Network View of DLPC-configured Nodes

In the SS7 network, the SINAP nodes configured with distributed logical point codes appear as a pair of STPs and perform STP-like functions. These functions include:

- Responding to route-set-test (RST) messages and route-set-congestion-test (RCT) messages
- Generating transfer-allowed (TFA) and transfer-prohibited (TFP) messages (or transfer-control-allowed (TCA) and transfer-controlled-prohibited (TCP) for ANSI cluster routing)
- In the ANSI network variant, the SINAP node responds to cluster-signaling-route-set-prohibited (RCP) test messages and cluster-signaling-route-set-restricted (RCR) test messages with a transfer-cluster-allowed (TCA) message when the logical point code (LPC) is available

The DLPC feature uses the broadcast method of sending TFA (TCA) messages to handle cases where the DLPC nodes are directly connected to SSPs, which do not generate RST query messages.

From the SS7 network point of view, the SINAP node controls more than one point code. The point codes for the SINAP user application(s) appear to be beyond the SINAP node as shown in Figure 4-4. The SINAP node responds to the RST and RCT messages based on own signaling point code (OSP). MTP Level 3 returns one of the following responses according to the current registration status of the user application:

- `available` (TFA message) - if the application is registered
- `no response` - if the application is not registered

The SINAP node treats the signaling-route-set-test message for restricted destinations (national option) (RSR) the same as an RST or RCT, although the DLPC does not establish restricted message flow.

From the SS7 network view shown in Figure 4-4, LPC X appears to be behind STP nodes Y and Z. The DLPC feature provides the connection of LPC X to nodes Y and Z. Thus, the messages sent from service switching point (SSP) node A to LPC X are routed through either node Y or Z depending on the network configuration and traffic. The figure shows the two network configurations supported by the DLPC feature:

- A DLPC (with node Y and node Z) connected through STPs to an end node SSP. Node availability is handled using a subset of the rules for an STP. The SSP does not use RST/RCT to determine node availability. The DLPC forward a TFA message when the LPC (user application) becomes available by configuring it for ISUP services.
- A DLPC (with node Y and node Z) connected directly to an SSP.

For either model the DLPC sends the TFA when the user application is configured for ISUP services via the isup_cfg_process() function. During the time between registering the user application and configuring it for ISUP services, it is possible to receive an RST and send a TFA. In both models (STP and SSP), the DLPC always provides the TFA message.

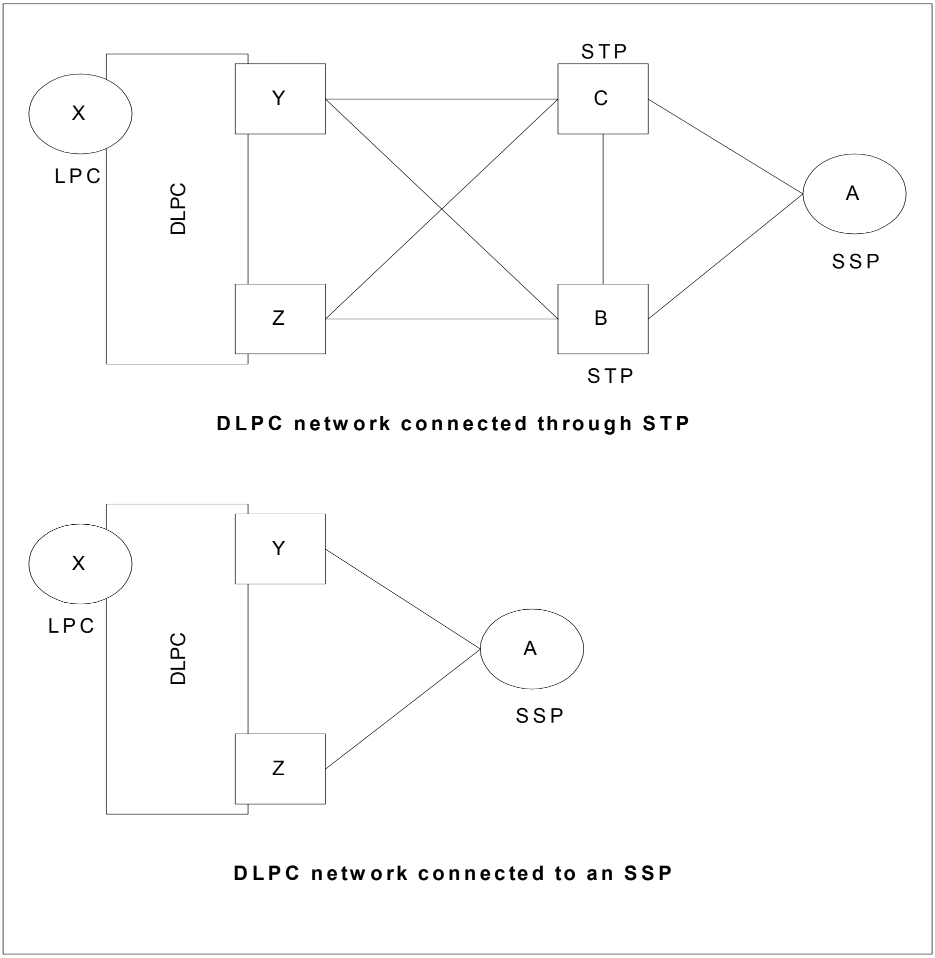

Figure 4-4. SS7 Network View of Two Types of DLPC Configurations

In the network connection where DLPC is connected through STP to the end node SSP, node availability follows the rules of an STP (signaling route set test (RST)). The DLPC responds to the STP messages according to the rules shown in Figure 4-5.

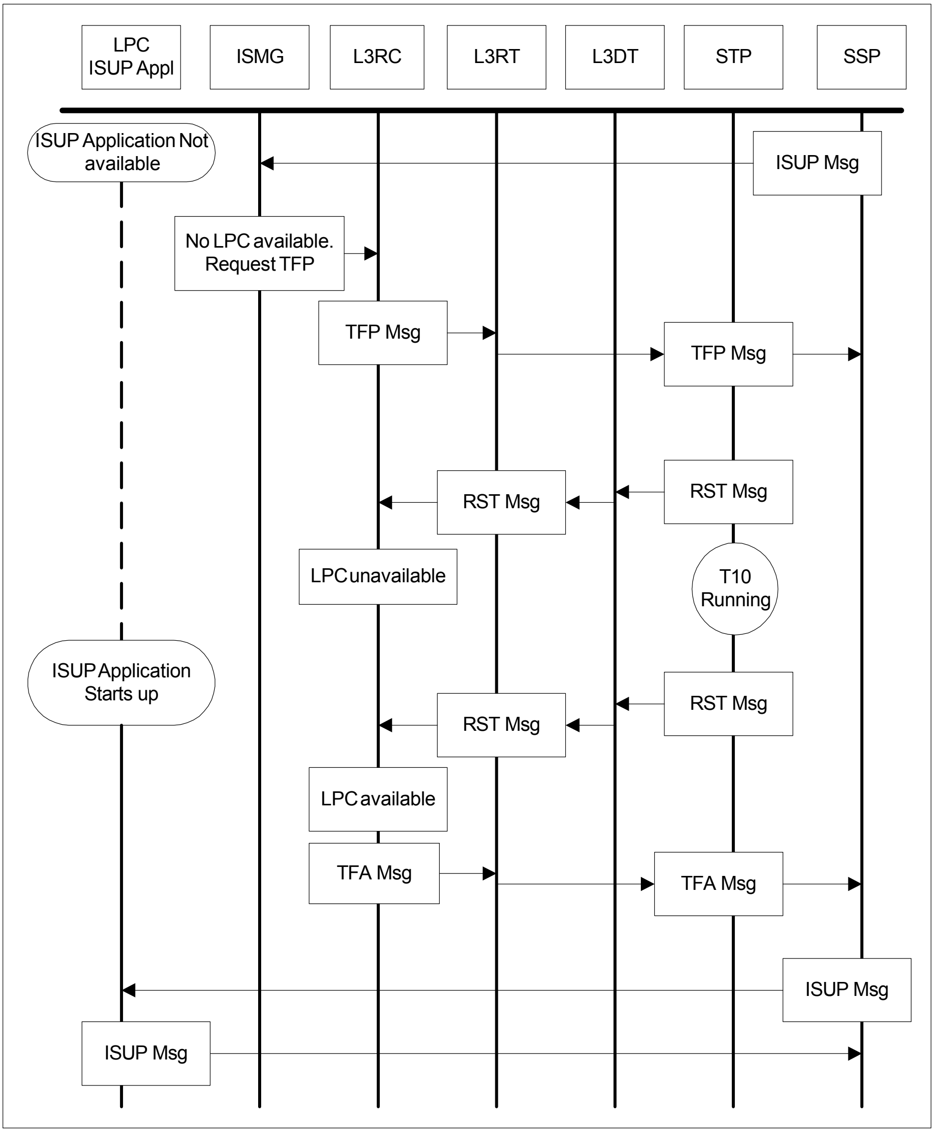

Figure 4-5. MTP Level 3 Route Management Data Flow (STP Model)

In the network configuration where the DLPC connects directly to an SSP, node availability is handled using a subset of the rules for an STP. The SSP does not use RST/RCT messages to determine node availability. The SINAP DLPC forwards TFA messages when the LPC (user application) becomes available by configuring the application for ISUP services as shown in

Figure 4-6. For either model, the DLPC sends the TFA message when the user application configures the ISUP application using the ISUP function call `isup_cfg_process()`. During the time between registering the user application and configuring it for ISUP services, it is possible to receive an RST and send a TFA. By using both models (STP and SSP), the DLPC always provides the TFA message and remains within the specifications for both models.

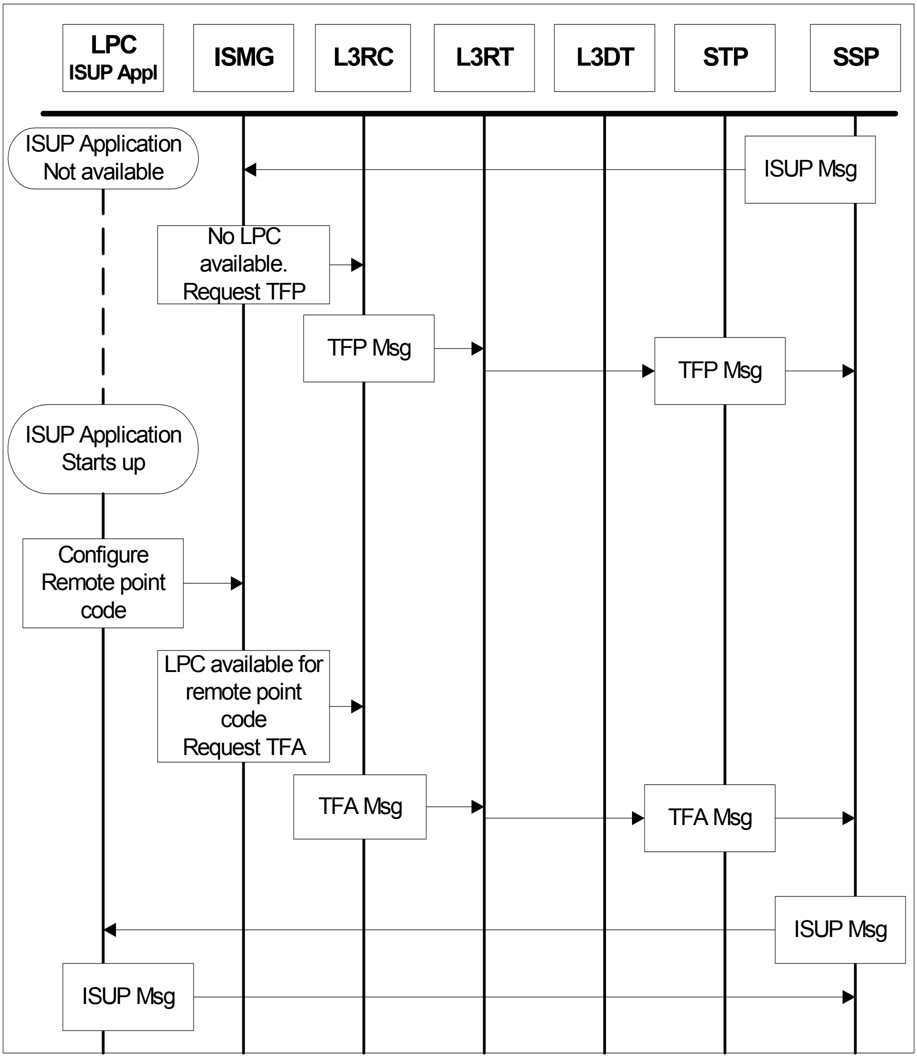

Figure 4-6. MTP Level 3 Route Management Data Flow (SSP and STP Model)

Configuration Requirements and Limitations

The following is a summary of the configuration requirements that apply to the DLPC feature. Note that the DLPC is supported in the CCITT, ANSI, and China network variants.

- In the current SINAP release, the DLPC feature applies only to the CCITT, ANSI, and China network variants and all ISUP versions that are valid for these network variants.
- You must define the `ISUP_FEATURE` environment variable for an ISUP version that is valid for the network variant configured on each SINAP node. If you do not define this variable, you cannot configure the DLPC feature through the `/etc/config_sinap` script. See the *SINAP/SS7 Installation Guide* (R8060) for detailed information on configuring the DLCP feature using the `/etc/config_sinap` script file.
- The associated Continuum machines can be co-located or located at different geographical sites.
- You must configure the DLPC feature using the `/etc/config_sinap` script. You can configure a standalone node as a logical point code router (LPCR) without an associated DLPC on another SINAP node or you can configure the node as a DLPC with an associated DLPC on another SINAP node.
- The user application on each SINAP node must be configured for ISUP services (via the `isup_cfg_process()` function.
- The two SINAP nodes configured for the DLPC feature must have matching network configurations. Configuring network elements differently on the two nodes, can result in unpredictable behavior and may cause potentially severe problems in your network. Note that the SINAP node does not verify configurations between machines.
- The DLPC-configured SINAP nodes are addressed from the SS7 network perspective as a pair of STP nodes and they perform STP-like functions for purposes of routing ISUP messages to LPCs. But, they cannot route messages to other SS7 nodes or forward messages to actual SS7 point codes behind them.
- Up to 15 ISUP applications can register for the same logical point code, but each originating point code (OPC) can be registered in the *active* mode for only one application process across both DLPC machines regardless of the LPC.
- Only one process per logical point code router (LPCR) can register an OPC in standby mode. Note that when an application process registers for an OPC, that process must handle all circuit identification codes (CICs) for that OPC. Split CIC ranges are not supported.
- The messages forwarded by the IMA must go through multiple hops. Therefore, you may experience increased response times on a DLPC-configured machine over those on a non-DLPC machine. However, the increase should be no more than 50%.
- In the ANSI network variant, the DLPC feature supports the configuration of `B` links and `D` links in link sets and combined link sets. You can view these link types using the `sy` commands `#STA,LST` (display static link set) and `#STA,CLS` (display static combined link set).

- The DLPC feature does not support use of `F` links or `C` links.
- The DLPC feature does not support MTP restart functionality.

Handling Failures

The DLPC feature allows for both planned outages (shutdown of a running application) and unplanned process failures where a process fails and is restarted (on same machine) or a process fails and the ISUP remote point codes are handled by a process on a different Continuum machine. Both of these types of outages are discussed in the following subsections.

Planned Outages (Manual Failover)

Planned outages or failures include software upgrades such as application and operating system upgrades and hardware upgrades. In these situations, the SINAP user application or machine is unavailable for some unknown time. You may need to move the SINAP user application to the other machine while the system is unavailable.

The complexity of the down time indicates the actions to be taken. If the down time duration is short enough (under 30 seconds) you may decide to do nothing more than restart the necessary processes. If the down time duration is planned for a long time (user-determined), then you should take actions to ensure 100% of the SS7 traffic is handled during the down time. For long duration down times you should move the SINAP user applications from the machine that is going down to the operational machine.

When you decide to shut down a machine, you must move the SINAP user application to the operational machine. The decision to initiate this action is user-determined.

It is the responsibility of the SINAP application to provide a user interface for specifying commands to gracefully shutdown. The details of the user interface design and implementation are also the responsibility of the ISUP user application.

Unplanned Failures (Automatic Failover)

All of the following failure scenarios are considered to be unplanned failures.

SINAP process failure. The SINAP node automatically restarts a failed critical SINAP process without knowledge of the SINAP user application and without loss of SINAP status or configuration data. This restart does not affect the availability of the SINAP user application. The current version includes the restart for both ISUP managers, and the same will apply for this enhancement. These processes restart in about one second.

SINAP user application failure or shutdown. The SINAP node recognizes both of these events and performs the restart operation.

Catastrophic failures that stop all SINAP process. These conditions cause the SINAP user application to become unavailable. In these cases, a restart operation is performed manually by the operator or it occurs automatically (restart is configured by the system using `inittab`). For purposes of this discussion, the SINAP node is assumed to be configured for an automatic restart. A complete SINAP restart takes approximately 1-2 minutes, plus the restart time of the SINAP user application.

Complete machine failure. The failure could cause a dump of the operating system to disk and possibly a restart if the system was configured to automatically restart. A complete system failure can last approximately 5 -30 minutes (assuming the system can be restarted).

TCP/IP network failures. This condition affects communications between Continuum machines (SS7 messages cannot be transported to the other machine). For a DLPC SINAP configuration, Stratus recommends that you install and run the Remote Network Interface (RNI), which provides fail-recovery backup for local area network (LAN) connections. This ensures that no single hardware fault can cause the TCP/IP network to fail. The recovery time is dependent on the TCP/IP network.

ISUP User Application Failure and Recovery

ISUP user applications use an active and standby method of failure recovery. An ISUP application registers to receive and process ISUP messages from a specified remote point code (each ISUP user application can register for more than one remote point code). Remote point codes are registered to an ISUP user application in one of the following two modes:

- Active — Inbound messages are routed to the ISUP user application, and the ISUP user application can send ISUP messages (as is normal on a conventional SINAP node)
- Standby — Inbound ISUP messages are not routed to the ISUP user application.

Two different ISUP user applications register for a remote point code using a different mode for each ISUP user application (one in active mode and the other in standby mode). This recovery method allows one ISUP user application to actively process ISUP messages for a specific remote point code while the other ISUP user application stands by to take over the active processing in case a failure occurs. Note that an ISUP user application could be registered with some remote point codes in active mode and other remote point codes in standby. If the ISUP user application (with the remote point code in active mode) fails, the ISUP manager is notified and it attempts to find the other ISUP user application (on either machine) that has the remote point code registered as standby changes its mode to active. Whenever the ISUP manager changes the mode of a remote point code, it notifies the ISUP user application of the change.

A failed process can recover by restarting, registering all the remote point codes as standby, then issuing a command (a message sent from the ISUP user application to the ISUP manager) to activate those remote point codes that were active before the failure. The command to activate the standby remote point code (for the ISUP user application issuing the command) finds the currently active remote point code (in an ISUP user application on either machine) and changes the remote point code's mode to standby and activates the standby remote point code for the ISUP user application. Note that each ISUP user application is responsible for the configuration of remote point codes to the application. The ISUP user application should have each remote point code registered twice—once in active mode and once in standby mode. The ISUP manager does not verify this configuration. See Table 4-1 for detailed information on user application failure modes.)

Table 4-1. Correcting Circuit Discrepancies

Active Remote Point Code This Machine	Standby Remote Point Code This Machine	Standby Remote Point Code Other Machine	Condition
Active	For own machine	Not applicable	Normal (standby on same machine).
Failed	For own machine (takes over as active)	Not applicable	Backup takes over on same machine (Application begins a restart to assume active role).
Failed	Failed	Not applicable	No ISUP messages are being processed for this remote point code until ISUP user application is restarted.
Active	Not applicable	For other machine	Normal (all standby on other machine).
Failed	Not applicable	For other machine (takes over as active)	Backup takes over on other machine (all applications shutdown on this machine).
Failed	Not applicable	Failed	No ISUP messages are being processed for this remote point code until the ISUP user application is restarted.
Active	For own machine	For other machine	This is a special case to allow the ISUP user application to be prepared to use either the standby on own machine or the other machine (see Note).

NOTE

The DLPC feature allows this special configuration to enable the ISUP user application to take over processing on either machine. This configuration allows for a failure or a commanded failover of the process for the active point codes (on own machine). When the active point code fails or is commanded to change mode, the process with the standby remote point codes on own machine become active. The standby point codes on the other machine are not affected. This allows for normal failure recovery on the same machine. If this machine has only the active remote point code processes, then a failure causes a switchover of the processing to the other machine (normal behavior).

If you must switch the user application to the other machine, then you must delete the standby remote point codes from the configuration on own machine before you perform the switchover. If you attempt to switchover to the other machine while own machine has a standby, the command is rejected, and no switchover occurs.

DLPC Failure Detection and Recovery

The DLPC machine pairs are used for ISUP user application processing. Each DLPC machine communicates with its mate and the SS7 network using external links. Each machine monitors the health of the communication paths. The ISUP manager is aware of communication failures, but does not repair failures. It only provides alternative paths (when possible) for the ISUP traffic. Note that if the ISUP manager dies, the SINAP node automatically restarts it without loss of status of the operational ISUP user applications.

Figure 4-7 shows the communications relationships between the Continuum systems configured for DLPC functionality

.

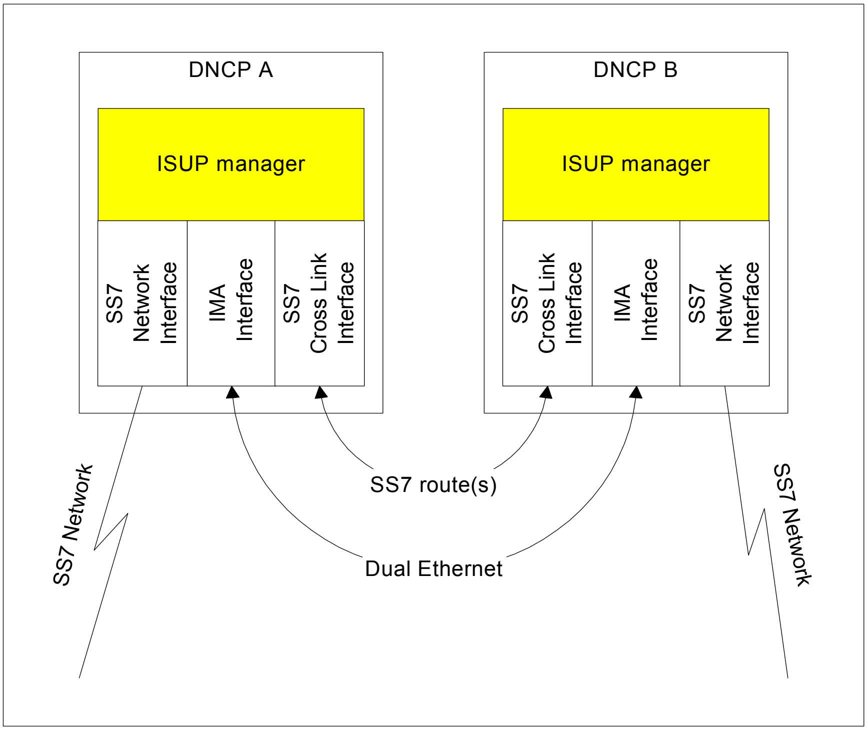

Figure 4-7. IMA Communications Between Continuum[1] Systems

Most single hardware faults or failures have no effect on the operation of the machine. A double fault (for example, both power supplies fail) causes the machine to die, and all communication from the machine is lost. Note that the communication Input/Output (I/O) boards (such as ARTIC, U4xx) are considered to be single points of failure. A loss of a communication I/O board shuts down communication attached to it. Thus, loss of multiple communication I/O boards (up to all boards) could cause the communications to and from the machine to fail. You should run the IMA cross-link on dual Ethernet boards (RNI) as shown in Figure 4-7 so a single fault does not cause an outage.

Major components of the software could fail or hang (continue to run, but not process messages). These failures are rare, and may not cause communications to fail instantly. However, the health-check messages should detect the failure. For example, if the SINAP driver causes the system to crash and perform a memory dump, during the memory dump (occurs over

[1] Continuum systems were previously called DNCP systems. This figure and other parts of the documentation still refer to DNCP systems.

several minutes) the absence of health-check message exchange indicates failure of the communications between the machines.

Individual SINAP process, including ISUP user applications are under the control of the SINAP node. If the node detects a failure (either by health check, or notification of termination), the node performs an automatic restart. For ISUP user applications, this means the application must register for health-check messages and restart options (or some other restart procedure).

You should monitor communication between the machines to determine the health of both machines in the DLPC configuration. The monitor includes failure notifications and health-check messages. Table 4-2 describes the communications modes for the IMA and SS7 cross-links and the SS7 network conditions (for outbound routes to remote point codes) associated with the modes. Note that in Table 4-2 the following terms apply:

- Operational — The communication path is working, normal traffic flowing, including health-check (`Are You Okay?`) messages.
- Failed — The communication path does not allow normal traffic to flow, the responses to health-check messages fail to occur in a timely manner, or the communication path is marked unavailable.

Table 4-2. Communication Modes and Conditions

Case Identifier	IMA Cross-link	SS7 Cross-link	SS7 network (all outbound routes to remote point codes)†	Condition (see Table 4-3)
1	Operational	Operational	Operational	Normal.
2	Failed	Operational	Operational	Cross-link failed.
3	Operational	Failed	Operational	SS7 cross-link failed.
4	Failed	Failed	Operational	Other machine failed.
5	Operational	Operational	Failed	Isolated from the SS7 network.
6	Operational	Failed	Failed	Isolated from the SS7 network.
7	Failed	Operational	Failed	Cross-link failed.
8	Failed	Failed	Failed	My machine is down.

† Failure of routes will generate the appropriate alarms. Recovery of the failed routes is handled by MTP level 3 (with the routing table updated as status changes).

Table 4-3 describes the communications failure conditions that can occur in the DLPC machines and the resulting action taken on each machine.

Table 4-3. Communications Conditions and Actions

Condition	Action
Normal	Normal traffic flow, Norman operation
Cross-link failed (the cross-link is a dual path network, thus a double fault)	Generate an alarm. Mark cross-link failed. Discard all IMA traffic destined for the other machine. Return TFP when appropriate. Wait for the cross-link to become active.
SS7 cross-link failed (this route should be configured as a dual path, thus a double fault)	Generate an alarm. Discard health-check messages. Wait for the route to become active.
Other machine failed	Generate an alarm. Activate all standby remote point codes (notify SINAP driver and ISUP user applications accordingly).
Isolated for the SS7 network	Generate an alarm. Accept ISUP messages from the other machine, and return the messages via IMA for distribution on the SS7 network by the other machine. Wait for routes to recover.
My machine is down	No action can be taken by ISUP manager.

System Health Checks on IMA Cross-links

The ISUP managers on the peer DLPC systems initiate health-check (`Are You OK?`) messages at regular intervals between the two machines. When the system is configured, an IMA connection is established between the two DLPC machines. The IMA library opens a connection between the machines whenever requested by the ISUP manager. A health-check message is sent to the other machine at 500ms intervals. The message contains a list of the logical point codes registered on the local machine (thus the health check message will contain LPC from this machine).

The ISUP manager should receive an acknowledgment for each health-check message sent. If the ISUP manager does not receive a health-check acknowledgment within 1500ms. (misses three acknowledgments), the IMA cross-link is considered unavailable. In this case, the ISUP manager generates an alarm indicating failure of the IMA.

The ISUP manager maintains the status of the IMA on each DLPC machine. The status flag is stored and maintained in shared memory. You can use sy to display the IMA cross-link status flags shown in Table 4-4.

Table 4-4. IMA Cross-link Status Flags

Status Flag Name	Description
`ISUP_XLINK_UNINITIALIZED`	No attempt has been made to start the IMA link. ISUP manager is in the process of starting or restarting.
`ISUP_XLINK_OPERATIONAL`	IMA cross-link is available and carrying messages between the peer ISUP managers.
`ISUP_XLINK_TIMED_OUT`	Missed the response for at least 3 consecutive health-check messages.
`ISUP_XLINK_DOWN`	No messages are being exchanged between peer ISUP managers.

System Health Checks on SS7 Cross-links

A health check is executed between the DLPC machines at regular intervals. When the system is configured, an SS7 route that carries the health-check messages established between the two DLPC machines. The health-check message and response are ISUP messages (SIO set to 5) with the message type set to an unused value (applies to both ANSI and ITU variants).

The health-check message is sent at 500ms intervals. In the MTP routing label the destination point code is the DLPC node (of the other machine), the originating point code is null (zero), the SIO is set to 5 (for ISUP message), and CIC value is set to null (zero). The message type is 160 (`0xA0`) for the health-check message and 161 (`0xA1`) for the health-check acknowledgment. Using the ISUP SIO and originating point code that is *not* registered (or a real point code), the message is routed to the ISUP manager by default (no ISUP user application registered for the originating point code). The message body contains a list of the logical point codes registered on the local machine (thus the health-check message contains the LPC from the local machine).

The ISUP manager should receive an acknowledgment for every health-check message it sends. If the IUSP manager does not receive a health-check response within 1500ms (misses receiving three acknowledgments in a row), the SS7 cross-link is considered unavailable.

The ISUP manager maintains status of the SS7 cross-link. The status flags are stored and maintained in shared memory. You can use `sy` to display the SS7 cross-link status flags shown in Table 4-5.

Table 4-5. SS7 Cross-link Status Flags

Status Flag Name	Description
`ISUP_XLINK_UNINITIALIZED`	No attempt has been made to start the SS7 cross-link. ISUP manager is in the process of starting or restarting.
`ISUP_XLINK_OPERATIONAL`	SS7 cross-link is available and carrying messages between the peer ISUP managers.
`ISUP_XLINK_TIMED_OUT`	Missed the response for at least 3 consecutive health-check messages.
`ISUP_XLINK_DOWN`	No health check messages are being exchanged between peer ISUP managers.

DLPC Alarms

DLPC related alarms are generated by the ISUP manager (`ismg`) and the SINAP driver.

ISUP Manager Alarms

The ISUP manager generates alarms for some DLPC-related events. Each alarm contains at least the fields shown in Table 4-6.

Table 4-6. DLPC Alarm Format

Field	Description
Category: `LPCR_EVENT`	Indicates a DLPC-generated event.

Subcategory:
`LPCR_FAILURE`
 	
`LPCR_MAJOR_EVENT`
	
	Specifies a critical event requiring user attention and action.
	
	Specifies a major or significant event requiring user attention, and maybe user action, depending on the event.

Type: `SW_EVENT`	Indicates the event is generated by software (not hardware).
State	Contains a value assigned by the event which has a specific meaning for the event.

Table 4-6. DLPC Alarm Format *(Continued)*

Field	Description
Code	Normally, contains the error code (`errno`) that generated the event, or in some cases, a value assigned to assist you in determining the cause of the event.
Message Text	Provides a brief description of the event. The message is formatted to include the string `ISMG` (indicates the message was generated by the ISUP manager), and a string that indicates the ISUP manager internal function call that generated the alarm.

Table 4-7 lists and describes the DLPC alarms generated by the ISUP Manager.

Table 4-7. DLPC Alarms Generated by the ISUP Manager

State Code	Code	Message Text	Reason Event is Generated
(N/A)	(N/A)	`ISMG:main() - Peer machine inaccessible`	The ISUP manager determined that both the IMA and SS7 cross-links failed.
(N/A)	(N/A)	`ISMG:main() - Activated all Standby point codes`	The ISUP manager determine that both cross-links failed and all standby point codes in this machine were activated.
(N/A)	(N/A)	`ISMG:main() - No Standby point code to activate`	The ISUP manager determined that both cross-links failed; the ISUP manager attempted to activate the standby point codes, but was unsuccessful.
Count of messages discarded (including this message	Message type code for message discarded	`ISMG:isup_put_ima() - Link down, could not send msg(s)`	After the first failure attempt to send an IMA message, this alarm is generated for each 13th attempt to send a message.

Table 4-7. DLPC Alarms Generated by the ISUP Manager *(Continued)*

State Code	Code	Message Text	Reason Event is Generated
(N/A)	Point code of peer DLPC node	`ISMG:ss7_xlink_health_check() - No route to peer`	No route was available during an attempt to generate and send the SS7 cross-link health-check message.
(N/A)	Error code that caused the alarm	`ISMG:ima_change_mode() - Could not change mode`	While attempting to change the mode of a remote point code as requested by the peer ISUP manager, the change request could not be completed.
(N/A)	Error code that caused the alarm	`ISMG:ima_active_standby() - Multiple point codes present`	While attempting to activate a standby remote point code as requested by the peer ISUP manager, an active and standby point code were found registered to two processes.
(N/A)	(N/A)	`ISMG:ima_active_standby() - No point code available for failover`	The peer ISUP manager requested a list of remote point codes to be activated, but none of the point codes in the list were in standby mode on this machine.
(N/A)	(N/A)	`ISMG:ima_lpc_not_available() - No route for TFP`	An attempt to respond with a TFP message to a point code failed. The ISUP manager could not find an LPC in either Continuum machine and no route exists to the point code. The message is discarded and no TFP is sent to the point code.

SINAP Driver Alarms

The SINAP driver generates an alarm with the category `LPCR_EVENT` and the subcategory `LPCR_MAJOR_EVENT` whenever it detects that the ISUP manager has failed. This alarm is in addition to the alarm it normally generates for process failure.

On a DLPC-configured node, failure of the ISUP manager is considered to be a major, but not critical, DLPC event since the ISUP manager is configured to be automatically restarted by the SINAP node and to recover its ISUP shared memory information. Alarms for lost messages due to the transitory absence of the ISUP manager are handled as they normally are on a conventional SINAP node. That is the alarms are sent up by the SINAP driver through the node management node interface (`nmni`) process which converts them to interprocess communications (IPC) messages that are sent to the node management trouble management (`nmtm`) process.

Sample Programs

Currently, the SINAP/SS7 software provides two sample ISUP programs, `isup_send` and `isup_recv`, that when used in tandem, simulate ISDN call setup, data transfer, and call release signaling in an ISDN network. The `isup_send` program generates the initiation side of all call setup, data transfer, and call termination. The `isup_recv` program simulates the call request receipt side of the network, including processing of the call indication, issuing an acceptance, receipt of data, processing the connection release indication, and issuing the release confirmation.

On a DLPC-configured node, the `isup_send` program contains an optional command line argument that allows you to specify a logical point code to be registered with the CASL.

The DLPC feature provides two extended versions (one for CCITT and one for ANSI) of the `isup_recv` program called `LPCR_recv` that allow you to simulate the behavior of the ISUP user application deployed on a DLPC-configured node. The DLPC feature also provides a utility (`LPCR_cmd`) that allows you to change the behavior of the `LPCR_recv` process.

The `LPCR_recv` and expanded `isup_send` programs are discussed in the two following subsections.

The `isup_send` Program

The `isup_send` program initiates the ISUP call setup sequence. On a DLPC-configured node, the `isup_send` program contains the optional switch `-L` that allows you to specify the logical point code to be registered with CASL. If you do *not* specify this switch, the LPC defaults to 0, and the LPC is not registered with the CASL. The presence of this switch results in the LPC being registered with the CASL. Any messages originating from `isup_send` are sent with the own point code equal to the specified LPC.

The format of the `isup_send` command is:

```
isup_recv -L pc [-dlsqvV -r pc -c qty -b first]
```

The following chart describes the arguments you can use with `isup_send`.

Argument	Description
`-L pc`	Specifies a logical point code (default is 0, or none).
`-v`	Terse usage statement .
`-V`	Verbose usage statement .
`-l`	Loop through setup, teardown.
`-s`	Silent mode (default is verbose).
`-q`	Display Circuit status (default is no display).
`-r pc`	Remote active pc.
`-c qty.`	Number of circuits (default is 3).
`-b first`	First circuit ID (default is 1).
`-d`	Debug mask set (default is not set).

The `isup_send` program also accepts three events generated by the ISUP Manager (`ismg`). The only action `isup_send` takes upon receiving an event is to write a message to the controlling terminal.

The LPCR_recv Program

The `LPCR_recv` program extends the capabilities of the `isup_recv` program to provide and example of and simulate the behavior of the ISUP user application deployed in a DLPC configuration. The `isup_recv` program simulates the call request receipt side of the network.

The `LPCR_recv` program extends the capabilities of the `isup_recv` program as follows:

- The `LPCR_recv` program allows registration of an LPC and multiple Origination Point Codes (OPCs) per LPC that request service from the ISUP application that is addressable by the LPC.
- The `LPCR_recv` interface allows you to define an OPC to be in the active or standby mode when it is initiated.
- The `LPCR_recv` program detects the mode change of an OPC from standby to active that is initiated by the ISUP manager. When the ISUP manager determines that an OPC in standby mode must be changed to active, the manager generates an IPC message to alert `LPCR_recv`. Upon receipt of this IPC message (`I_ISUP_ACTIVATE_PC`) the `LPCR_recv` program begins processing all indications for this OPC.

- The LPCR_recv program allows you to change all OPCs that are currently in standby mode to the active mode. It also allows you to change all OPCs that are currently in the active mode to standby mode. The initiator of the mode change in this case is the application, not the ISUP manager driven as described in the previous bullet.
- LPCR_recv allows you to set the circuit state of all circuits that are associated with an LPC/OPC that has transitioned from standby to active mode. However, it not necessary to change the circuit status of a circuit when an LPC/OPC transitions from active to standby mode. An application that has its LPC in standby mode receives no MSUs. When LPCR_recv calls isup_change_pc_mode() to transition an LPC back to active mode, the ISSL library automatically sets the state to IDLE for all circuits. LPCR_recv follows the call to isup_change_pc_mode() with a call to isup_restore_circuit_state() or isup_purge_circuit() for each circuit associated with the transitioned point code.
- LPCR_recv must be able to save the state of the circuits associated with a OPC in some external *container*. This container will be read by the LPCR_recv process taking over failed LPCR_recv circuits for the failed point code reads the container in order to set the state of all managed circuits. This container takes the form of a binary file that is accessible for both cooperating LPCR_recv processes.
- The file name of the binary file is constructed from the combination of the LPC and remote point code (RPC) values. There will be one file for each LPC/RPC pair. The process that maintains the point code as *active* is the only process that writes to each file. The LPCR_recv that has a point code in standby mode has *read* access to the file. The path defaults to /tmp but you can override this path on the LPCR_recv command line (using the -p option).
- The file must be located on an NFS-mounted file system that is viewable by both Continuum machines hosting the LPCR_recv tools. For LPCR_recv processes on the same Continuum system but different SINAP nodes, the file must simply be located on a file system that is viewable by both processes on the local host. When an NFS file system is required, Stratus recommend that the NFS file system be hosted on a third, non-Continuum system

The format of the LPCR_recv command is as follows

```
LPCR_recv -L n-c-m [-sqvVd -p path -r pc -b first -c qty [-r pc -b first-
c qty]]
```

The following chart lists and describes the arguments that can be used with LPCR_recv. Note that you must supply at least the -r or the -S argument.

Argument	Description
-L n-c-m	Required logical point code.

Argument	Description
`-r n-c-m`	Remote active point code (ANSI point code format); CCITT point codes are entered as a single number.
`-S n-c-m`	Remote standby point code.
`-b first`	First circuit ID (default is 1 or current highest circuit identification code (CIC) ID) .
`-c qty.`	Number of circuits (default is 3).
`-C qty.`	Number of remote point codes (if specifying a range from `-r` to `-r+qty`).
`-d`	Debug mask set, default not.
`-p path`	Directory path to circuit state files.
`-s`	Silent mode, default is verbose.
`-q`	Display circuit status, default is no display.
`-v`	Terse usage statement.
`-V`	Verbose usage statement.

Figure 4-8 shows the operation of two cooperating `LPCR_recv` instances on two Continuum[2] machines. The picture is simplified from the actual network configuration however it is accurate from the perspective of the `LPCR_send` and `LPCR_recv` applications.

[2] Continuum systems were previously called DNCP systems. This figure and other parts of the documentation still refer to DNCP systems.

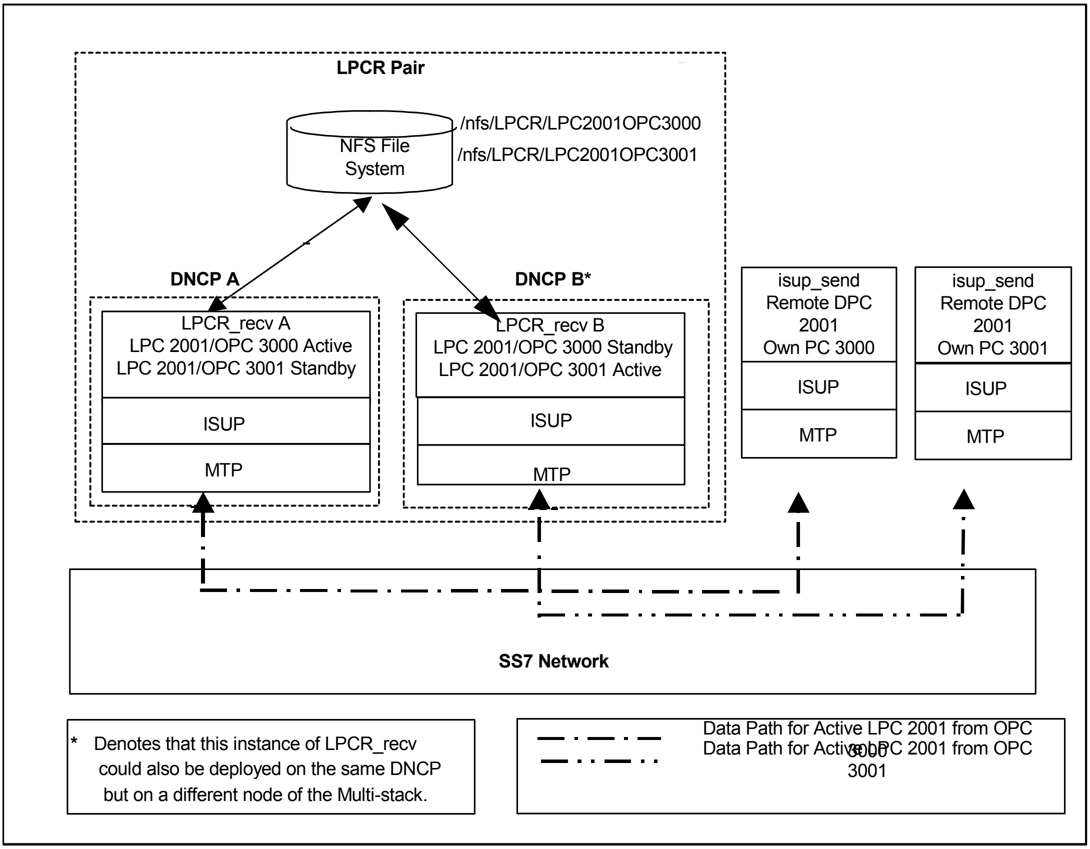

Figure 4-8. LPCR_recv Normal Behavior

NOTE

The fact that an MSU can arrive at one peer of a pair of cooperating LPCR systems then must be sent to the other LPCR system to be delivered to the application that owns the active LPC is invisible to the application. Therefore, this interaction is not shown in Figure 4-8 to simplify it from the `LPCR_recv` perspective.

LPCR_recv A processes all circuit requests for LPC 2001 initiated by the LPCR_send whose Origination PC is 3000 as this LPC/OPC combination is in the active mode. LPCR_recv also registers LPC 2001 however this LPCs associated Origination PC 3001 is registered in the standby mode and therefore should not receive any request unless a failure occurs on Continuum B.

On Continuum B, `LPCR_recv` registers LPC 2001 associated Origination PC 3001 as active and processes all requests destined for LPC 2001 from OPC 3001. Also on Continuum B, OPC 3000 is registered in standby mode for requests destined for LPC 2001. This `LPCR_recv` should not receive any request unless the associated LPCR_recv in active mode on Continuum A either terminates orderly or suffers catastrophic failure (unplanned failover).

Each `LPCR_recv` process is responsible for writing its circuit state of all circuits associated with an active LPC/OPC association to a binary file which is located in a file system that is readable by the peer LPCR_recv process. You should update this file periodically (since circuit states change) to ensure the state data in the file is current and accurate. In Figure 4-8, `LPCR_recv` A maintains its circuit state in file `/nfs/LPCR/LPC2001OPC3000` for its active point code. `LPCR_recv` B maintains its circuit state in `/nfs/LPCR/LPC2001OPC3001`.

Figure 4-9 depicts the function taken by `LPCR_recv` at the point of a peer `LPCR_recv` failure. In this case `LPCR_recv` B abnormally terminated. Upon detection that the LPC is no longer available the SS7 network will route all circuit activity to the peer pseudo-STP located on Continuum A. When this happens, `LPCR_recv` A receives an IPC message (`I_ISUP_ACTIVATE_PC`) from the ISUP manager indicating that OPC 3001 (currently in standby mode) was just made active for LPC 2001. When `LPCR_recv` A detects this situation, it first reads the status file (`/nfs/LPCR/LPC2001OPC3001` in the diagram) to obtain the last state all circuit requests managed for OPC 3001. `LPCR_recv` then makes use of the new ISSL API call `isup_restore_circuit_state()` to set the status of all circuits associated with the LPC that were in a stable state. For those circuits that were in a transient state, `isup_purge_circuit()` is called instead, to set them to idle. This completes the fail-over scenario. All events for LPC 2001 received from OPC 3001 will now be processed by `LPCR_recv` A.

NOTE

The simplest form of simulating this failure is to send a kill signal to the target process.

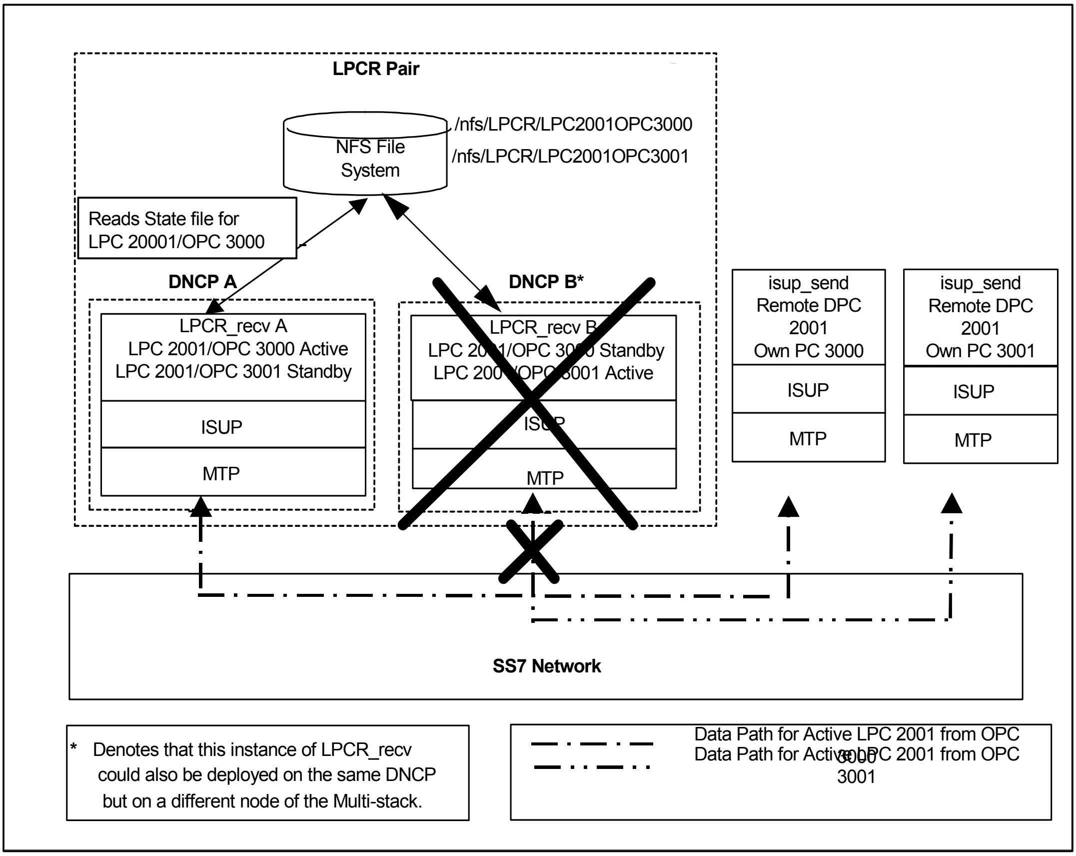

Figure 4-9. LPCR_recv Failure Behavior (Unplanned Failover)

Using `LPCR_recv` for Point Code Limit Testing

You can use the optional `-C qty` argument with `LPCR_recv` to perform point code limit testing. The qty specified is the number of active remote point codes for which `LPCR_recv` is to register. When this option is present, `LPCR_recv` registers active remote point codes for the starting point code specified by the `-r pc` option. `LPCR_recv` then increments the point code by one and registers the calculated point code as active. This action continues until qty point codes have been registered up to a maximum of 256 per ISUP process. Each point code processed registers for a single CIC.

The LPCR Utility Program (LPCR_cmd)

The LPCR_cmd is a utility program that only registers with CASL to make use of the SNAP interprocess communications (IPC) capabilities. This utility issues IPC commands to the target LPCR_recv process. You can use this utility to perform the following tasks:

- Change a specified OPC from standby to active mode
- Change a specified OPC from active to standby mode
- Request the LPCR_recv to read an OPC state file and set the state of all circuits for the specified OPC.
- Request the LPCR_recv to dump the status of a specified OPC to a specified file.

The format of the command is as follows:

```
LPCR_cmd -r n-c-m [-asRDvV -i instance]]
```

The following chart describes the arguments you can use with the LPCR_cmd.

Argument	Description
`-r n-c-m`	Required remote point code to be manipulated.
`-i instance`	LPCR_recv instance (default is 1st instance).
`-a`	Change specified point code to active mode.
`-s`	Change specified point code to standby mode.
`-R`	Reset the circuit state for the specified point code.
`-D`	Dump the circuits states (to file) for the specified point code.
`-v`	Terse usage statement.
`-V`	Verbose usage statement.

Using the LPCR_cmd to Perform an LPC Switchback

To perform a switchback operation, perform the following tasks.

- Restart the failed LPCR_recv process on the second Continuum so that the LPCs for which it is to resume responsibility are set to mode *Standby*. Do this by calling the isup_add_standby_rem_ssp() function for all LPCs to be registered.
- On the failover Continuum machine, issue a command via the LPCR_cmd to request that a state file for the specified LPC/OPC combination be generated to save the current circuit state. All states will be obtained by a call to the existing isup_get_circuit_status() function.

- On the recovered Continuum, issue a command via its instance of the interactive `LPCR_recv` command to change the state of the LPC to be reacquired to the *Active* mode. This is accomplished by calling the `isup_change_pc_mode()` function for the target LPC. When this occurs, the ISUP manager process sends a message to its peer requesting that the LPC/OPC combination be changed from active to standby mode.
- On the recovered Continuum machine, issue a command via its instance of the interactive `LPCR_recv` command to read the previously generated status file, and set the circuit states as specified in the file using the `isup_restore_circuit_state()` function or `isup_purge_circuit()` for those circuits that were in a transient state as specified in the file.

Using the `sy` Debugging Tool

For DLPC-configured nodes, the `sy` debugging tool provides commands that allow you to display logical point code information as described in the subsections that follow.

The `#ISUP` Command

The #ISUP command prints all ISUP shared memory data. This command has the following four forms:

- `#ISUP` — Generates output as if the PROC, PC and LPC subcommands were all issued
- `#ISUP,PROC` — Prints the contents of ISUP shared memory usage
- `#ISUP,PC` — Displays the point codes of all registered processes
- `#ISUP,LPC` — Displays all logical point codes registered on the cooperating DLPC node and all additional address information which may be optionally defined for a cooperating DLPC node

The `#ISUP,PROC` Command

The following example output of the `#ISUP,PROC` command shows the shared memory usage when logical point codes are registered.

```
ISUP Shared Memory Usage:
       active_pc=%d, reserved_pc=%d, active_proc=%d
Currently Registered ISUP processes:
      0: associated pid= %d, pc_count= %d, reserved_pc_count= %d, LPC= %d
       isup_key={node,mod,app,proc,#}
      1: associated pid= %d, pc_count= %d, reserved_pc_count= %d, LPC= %d
       isup_key={node,mod,app,proc,#}
```

The following table defines each of the keyword /value pairs shown in the example output.

Keyword	Meaning

active_pc	Number of active point codes for all ISUP application processes.
reserved_pc	Number of total reserved point codes for all ISUP application processes.
active_proc	Number of currently active processes for all ISUP application processes.
associated pid	UNIX process ID of the ISUP application process registered.
pc_count	Count of associated point codes with the process.
reserved_pc_count	Reserved point code count.
LPC	Logical Point Code if and only if the process defined an LPC that is not the same as the own signaling point code (OSP).
node	Up to a 4-byte node name assigned at CASL registration time.
mod	Up to a 4-byte module name assigned at CASL registration time.
App	Up to a4-byte application identifier assigned at CASL registration time.
proc	Up to a 4-byte process identifier assigned at CASL registration time.
#	Identifies the instance (1-16) of the registered application.

The `#ISUP,PC` Command

The `#ISUP,PC` command displays the point codes associated with the locally registered ISUP processes. If there are registered ISUP processes, the number of active point codes and number of active processes are printed. Then the point codes for each active process are printed in ANSI or CCITT format depending on the network variant in use. Following the point code is the node, module, application name, process name, and the instance as displayed by the `#ISUP,PROC`.

Example 1: `#ISUP,PC` command output when no LPCs are registered:

```
ISUP Shared Memory Usage:
     active_pc=%d, active_proc=%d
Point Codes associated with ISUP processes:
     pc=2001, pc_state=%s, peer_pc_state=%s, proc_index= %d ->
node,mod,app,proc,#
```

For each of the 2048 possible point codes.
Example 1: `#ISUP,PC` command output when LPCs are registered

```
ISUP Shared Memory Usage:
```

```
     active_pc=%d, active_proc=%d
Point Codes associated with ISUP processes:
    pc=2001, mode, pc_state=%s, proc_index= %d -> node,mod,app,proc,#
```

For each of the 2048 possible point codes.

The following table defines each of the keyword /value pairs found in the example output.

State Code	Reason Event is Generated
`active_pc`	Number of active point codes for all ISUP application processes.
`active_proc`	Number of currently active processes for all ISUP application processes.
`pc`	Point code (in either ANSI or CCITT format) for the associated process.
`mode`	The mode of the LPC can be either `ACTIVE` or `STANDBY`.
`pc_state`	The current state of the point code on the local machine. The state can be one of the following states: • `UNEQUIPPED` • `ALLOWED` • `PROHIBITED` • `CONGESTED`
`peer_pc_state`	The current state of the Point Code on the peer machine. The state can be one of the following: • `UNEQUIPPED` • `ALLOWED` • `PROHIBITED` • `CONGESTED`
`proc_index`	The numerical index of this process in the IPC table.
`node`	Up to a 4-byte node name assigned at CASL registration time.
`mod`	Up to a 4-byte module name assigned at CASL registration time.
`app`	Up to a 4-byte application identifier assigned at CASL registration time.
`proc`	Up to a 4-byte process identifier assigned at CASL registration time.
`#`	Identifies the instance (1–16) of the registered application.

The `#ISUP,LPC` Command

The `#ISUP,LPC` command displays all logical point codes registered on the cooperating DLPC and all additional address information which may be optionally defined for a cooperating DLPC. The following example shows sample output of the `#ISUP,LPC` command.

```
LPCR Peer Information:
IP Address: 101.201.301.401, Port = 2001
Retry Attempts = 10 Retry Interval = 2
Peer PC=2002

Status of Cross Links to Peer LPCR:
Inter-Module Agent Link: Operational
SS7 Cross Route:          Operational
Process Termination Request: None.

Logical Point Codes Registered at Peer:
0: LPC=3001
1: LPC=3002
```

Valid values for the status of cross links to peer LPCR are:

- `Uninitialized`
- `Attempting Start`
- `Operational`
- `Timed Out`
- `Down`

Valid values for the process termination request are:

- `None`
- `Processing`

When the DLPC feature is not enabled on the SINAP node, the output from the `#ISUP,LPC` command looks similar to the following example.

```
LPCR Peer Information:
IP Address: Not Provisioned, Port = 0
Retry Attempts = 0 Retry Interval = 0
Peer PC=Not Provisioned

Logical Point Codes Registered at Peer:
==> NONE <==
```

The `IP Address`, `Port`, `Retry Attempts`, and `Retry Interval` will only be non-zero if configured. The following chart defines each of the keyword /value pairs shown in the example output.

Keyword	Meaning
`IP Address`	The IP address of the cooperating Continuum in the DLPC/ Continuum pair.
`Port`	The port on the cooperating Continuum for the virtual C link connection.
`Retry Attempts`	The number of times a TCP/IP connection attempt to the peer DLPC should be retried before giving up.
`Retry Interval`	The number of seconds to wait between connection establishment retry attempts.
`Peer PC`	The Own Point Code of the cooperating Continuum in the DLPC pair.
`LPC`	A list of the Logical Point Codes that have been registered on the peer DLPC.

The `#IPC[[,table index]|[,LPC]]` Command

The command `#IPC` displays (without the optional decimal value table index) generates a terse list of all processes registered with CASL. A maximum of 256 processes can register. The format of the list is as follows:

`index:node,mod,app,proc,instance,`*`LPC= logical pc`*`, pid=UNIX id`

The following chart defines each of the keyword /value pairs found in the example output..

Keyword	Value Meaning
`node`	Up to a 4-byte node name assigned at CASL registration time.
`mod`	Up to a 4-byte module name assigned at CASL registration time.
`app`	Up to a 4-byte application identifier assigned at CASL registration time.
`proc`	Up to a 4-byte process identifier assigned at CASL registration time.
`instance`	The instance of a particular application. Each ISUP application can have up to 16 process instances to help share the load of the application.
`LPC`	Logical point code registered by this process. This will be optionally displayed only if the value of the Logical Point Code is not the same as the value of the OPC in IPC shared memory for the process.

Keyword	Value Meaning
pid	UNIX process id of the application instance.

The output generated for the #IPC command for a DLPC-configured node displays all registered LPCs using the following format:

```
Locally Registered Logical Point Codes:
index: xxx-xxx-xxx
[index2: xxx-xxx-xxx]
```

The following two examples show sample output generated by the `#IPC,LPC` command for a DLPC-configured node. Example 1 shows output for the ANSI network variant and Example 2 shows sample output for the CCITT network variant.

Example 1: Sample output for the ANSI variant.

```
Locally Registered Logical Point Codes:
1: 124-201-100
2: 124-201-101
```

Example 2: Sample output for the CCITT variant.

```
Locally Registered Logical Point Codes:
1: 2001
2: 2002
```

You can use the `sy #IPC` command to get more specific information about particular application and IPC table instances by adding the optional table index parameter to the command. The shared memory segment containing the IPC data has a `Logical PC` (point code) field added to it. When the DLPC feature is not enabled on a SINAP node, the `Logical PC` field contains the Own Signaling Point Code (OSP). When the `Logical PC` field does not match the OSP, the LPC is added to the process instance output. When the table index is specified, the output looks similar to the following example.

```
IPC process table entry entry_no:
key=nd,md,ap,pc,inst, key.ipc_index = entry_no, bfence=0x%x,
efence=0x%x
entry=%d, link=%d, pid=%ld, h_timer_id=%d
index=(idx.n,idx.m,idx.a,idx.p,idx.i) fsignal=%d, fcmd=%d, fss7=%d
fhealth=0, fparent=0, app_ver=[ANSI|CCITT], proc_ver=Relx.x
failure_option=[NO_ACTION |
                SCRIPT,script=%s\n",pipc->fail.script |
                SEND_MSG to %s,%s,%s,%s, msg=0x%lx]
```

```
[sio_ssn_ind=%s, sio_ssn=%d |
 sio_ssn_ind=%d, sio_ssn=%d ]
[ss7_input_boundary=%d, ss7_primitive=%d |
 ss7_input_boundary=%s, ss7_primitive=%s ]
LogicalPC = 124-201-100
```

The `#L3,RC,trace` Command

This `#L3,RC,trace` command traces delivery of internal messages within the MTP layers. On a DLPC-configured node, an inbound ISUP message can be received on the local DLPC node that cannot be processed but can be processed on the remote DLPC. This occurs when an application registers an LPC on one DLPC node but not on the associated DLPC node.

The MTP layer on one of the DLPC nodes cannot issue TFP and TFA messages to the STPs in the SS7 network without first checking to see if the associated DLPC has the LPC registered and determining whether the message can be forwarded to the associated DLPC node for processing.

The `#L3,RC,trace` command provides expanded capabilities which trace the internal communication between the MTP (`L3RC`) and the ISUP manager components that you can use for both testing and problem resolution purposes. The ISUP manager generates the following internal messages and delivers them to `L3RC`.

- Transfer Prohibited (TFP)
- Transfer Allowed (TFA)
- Transfer Restricted (TFR)

The following example shows output from the `sy` trace for each of these messages. The example is for the ANSI network variant.

```
M_MTP_TFP, dpc=126-301-11, lpc=124-201-100
M_MTP_TFA, dpc=126-301-11, lpc=124-201-100
M_MTP_TFR, dpc=126-301-11, lpc=124-201-100
```

The `#STA,LST` and `#STA,CLS` Commands

The `#STA,LST` command displays the contents of the static shared memory for configured link sets. The `#STA,CLS` command displays the contents of the static shared memory for configured combined link sets. The ANSI variant of this command displays the link type. Currently, conventional SINAP nodes, support link set types A, `E`, and `F`. Supported combined link set types are A and E. On DLPC-configured SINAP nodes, the two `sy` commands also display the link type as type `B` or `D` for the new link set or combined link set being supported between the peer Continuum machines in a DLPC configuration.

The `#STA,LST` Command

The format (for ANSI variant) of the `#STA,LST` command follows.

```
static LINKSET table:
idx(LSetId):lsetname st=%s,adpc=%d-%d-%d, %d dpcs,%d
lnks,act=%d,ld=%d,tp=%s, cls=N(0)

slc = 0 nc nc nc nc nc nc nc nc nc nc nc nc nc nc nc
```

The following example shows output from the `#STA,LST` command.

```
static LINKSET table:
 0( 0):LSET1 st=ACTV,adpc=254-54-1, 1 dpcs,1
lnks,act=1,ld=1,tp=A, cls=N(0)
 1( 1):LSET2 st=ACTV,adpc=254-55-1, 1 dpcs,1
lnks,act=1,ld=1,tp=B, cls=N(0)

slc = 0 1 nc nc nc nc nc nc nc nc nc nc nc nc nc nc
```

Note the *B* following the link type (`tp`) field of the output for link 1.

The `#STA,CLS` Command

In the ANSI network variant, the `#STA,CLS` command displays the contents of the static shared memory for configured combined link sets.

The following example shows output from `#STA,CLS` command.

```
COMBINED LINKSET:
0( 0):CLSET1 type = B, lset = 0 + 1
```

Note the *B* following the link type (`type`) field of the output for link 1. In each of the previous examples for the `#STA` command, if provisioned link sets with *D* links, a single capital D displays for the `tp` (or `type)` field be in place of the letter B.

Chapter 5
ISUP Services Reference

This chapter provides reference information about the ISUP application programming interface (API), which provides access to ISUP services on SINAP/SS7 nodes. It contains the following sections.

- "ISUP Services Functions"
- "ISUP Services Structures"
- "ISUP Services Primitives"
- "ISUP Services Include Files"

ISUP Services Functions

Table 5-1 describes the ISUP services API functions you include in an application so it can use the SINAP/SS7 ISUP services.

Table 5-1. ISUP Services Function Calls

Category	Function Call	Description
ISUP Circuit Handling	`isup_add_circuit()`	Adds an ISUP circuit to an application's configuration.
	`isup_del_circuit()`	Deletes an ISUP circuit from the application's configuration.
	`isup_get_circuit_status()`	Returns a pointer to the circuit identification code (CIC) status structure (`isup_status_t`) which contains the current state and status of an individual CIC.
	`isup_purge_circuit()`	Sets the requested circuit to an `IDLE` state and terminates any outstanding timers for the circuit.
	`isup_restore_circuit_state()`	Allows the ISUP user application to modify a specified circuit state to restore it to a previou state if the Distributed Logical Point Code (DLPC) feature is configured on the SINAP node.

Table 5-1. ISUP Services Function Calls *(Continued)*

Category	Function Call	Description
	`isup_start_utimer()`	Starts a user timer for an ISUP circuit the application process has configured
	`isup_stop_utimer()`	Stops a user timer previously started for a circuit.
ISUP Configuration	`isup_add_rem_ssp()`	Adds a remote service switching point (SSP) to the application's configuration.
	`isup_add_standby_rem_ssp()`	Adds a standby remote service switching point (SSP) to the application's configuration if the Distributed Logical Point Code (DLPC) feature is configured in the SINAP node.
	`isup_cfg_circuit_info()`	Modifies the circuit group characteristic indicator default values for a specified circuit.
	`isup_cfg_process()`	Configures a user process or application instance.
	`isup_change_pc_mode()`	Changes the point code mode of a registered point code from *active* to *standby* (or from *standby* to *active*) if the Distributed Logical Point Code (DLPC) feature is configured in the SINAP node.
	`isup_del_rem_ssp()`	Deletes a remote SSP from the application's configuration.
ISUP Message Handling	`isup_free_msg()`	Frees memory for an ISUP-layer message.
	`isup_get_msg()`	Inputs a message at the ISUP-layer interface.
	`isup_put_msg()`	Outputs a message at the ISUP-layer interface.
	`isup_alloc_msg()`	Allocates memory for an ISUP-layer message.

The `sinap.h` include file defines the maximum number of ISUP services components a single application user process can manage. For example, `MAX_ISUP_CIRCUITS` defines the maximum number of circuits, and `MAX_ISUP_REM_SSPS` defines the maximum number of remote service switching points (SSPs).

NOTE

You should **never** modify any settings in the `sinap.h` include file or any other of the SINAP/SS7 system's include files.

The following sections provide a more complete description of the ISUP services function calls listed in Table 5-1. The function calls are presented in alphabetical order.

isup_add_circuit()

SYNOPSIS

```
int  isup_add_circuit (CirId circuit,
                        Cic   cic_val,
                        int   ssp_pc);
```

DESCRIPTION

The `isup_add_circuit()` function adds an ISUP circuit to the calling application's configuration. A single ISUP services application user process can manage the number of circuits specified by the `MAX_CIRCUITS` to `isup_cfg_process`, which cannot exceed the `MAX_ISUP_CIRCUITS` definition in the `sinap.h` include file.

The ISUP services application must call this function once for each circuit that it wants to handle. The value you assign to the circuit's index number must be unique among all the circuits the application user process has configured for all of its remote service switching points (SSPs). The specified remote SSP must already be configured and have a valid route set configured for it.

PARAMETERS

* *`circuit`* (input)
  Specifies the circuit index number for the circuit being configured. This value must be unique among all of the application's circuits.

* *`cic_val`* (input)
  Specifies the Circuit Identification Code (CIC) value for the circuit being configured.

* *`ssp_pc`* (input)
  Specifies the point code of the remote SSP for the circuit being configured.

RETURN VALUES

If successful, the `isup_add_circuit()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_ALREADY_ADDED`	The specified circuit already exists.
`IS_ERR_INTERNAL_FAILURE`	An internal configuration error occurred.
`IS_ERR_LIMIT_EXCEEDED`	The maximum allowable number of ISUP circuits has already been configured.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_CONFIG_PARAMETER`	The circuit index number specified was 0. Change this value to a non-zero value.
`IS_ERR_UNKNOWN_REM_SSP`	The specified remote SSP is unknown.

isup_add_rem_ssp()

SYNOPSIS

```
int  isup_add_rem_ssp (int ssp_pc);
```

DESCRIPTION

The `isup_add_rem_ssp()` function adds an active remote SSP to the calling application's configuration. The application must call this function once for each remote SSP that it will handle. A single ISUP services application user process can manage the number of remote SSPs specified by the `max_rem_ssps` parameter of the `isup_cfg_process()`. Note that the `max_rem_ssps` parameter cannot exceed the `MAX_ISUP_REM_SSPS` variable in the `sinap.h` include file. The maximum number of remote SSPs an application user process can manage is 256. The point code of the remote SSP must be unique among all remote SSPs configured by the application user processes.

NOTE

The `isup_add_rem_ssp()` function adds an *active* point code to the ISUP application's configuration. If the Distributed Logical Point Code (DLPC) feature is configured on the SINAP node. The ISUP application should configure each remote SSP it will handle twice - once as an active remote SSP on the local Continuum system and once as a standby remote SSP (using the `isup_add_standby_rem_ssp()` function call on a different, associated Continuum system). See the `isup_add_standby_rem_ssp()` function in the chapter for details on configuring a standby remote SSP.

PARAMETERS

* *`ssp_pc`* (input)
  Specifies the point code of the remote SSP being added to the configuration.

RETURN VALUES

If successful, the `isup_add_rem_ssp()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_ALREADY_ADDED`	The specified remote SSP already exists.
`IS_ERR_ISMG_IPC_FAILURE`	An invalid IPC operation occurred.
`IS_ERR_LIMIT_EXCEEDED`	The maximum allowable number of remote SSPs has already been configured.
`IS_ERR_NO_ROUTESET`	The specified remote SSP does not have a route set configured for it.
`IS_ERR_PC_OWNED`	Another user has configured the specified remote SSP as its own.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_PROC_NOTCONFIG`	Not enough memory is available to add the requested point code(s).

isup_add_standby_rem_ssp()

SYNOPSIS

```
int isup_add_standby_rem_ssp(int ssp_pc);
```

DESCRIPTION

The `isup_add_rem_ssp()` function adds a remote SSP to the calling application's configuration in *standby* mode. A standby SSP does not handle SS7 messages until the ISUP application on the active, remote SSP fails or requests a change in status from active to standby mode. Then the standby remote SSP automatically begins receiving and sending SS7 messages.

The application can only call this function when the DLPC feature has been configured on the node. The application must call this function once for each standby remote SSP it will handle. The SINAP node treats the standby remote SSP the same as the active point code in that the node performs sanity checks to verify:

- The process is configured
- The same point code has not been added to the configuration
- The maximum number of point codes has not been exceeded

Note that point code modes are stored in the remote point code table. The SINAP node verifies the point code mode for incoming and out going messages, blocks all incoming ISUP messages on standby circuits, and prevents outgoing messages from being sent on standby circuits.

When the DLPC feature is configured, the ISUP application should configure each remote SSP it will handle as an active point code on one SINAP node and as a standby point code on the other associated node.

PARAMETERS

* `ssp_pc` (input)
    Specifies the point code of the standby remote SSP being added to the configuration.

RETURN VALUES

If successful, the `isup_add_standby_rem_ssp()` function call returns the value zero (0).

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_AlREADY_ADDED`	The specified remote SSP already exists.
`IS_ERR_ISMG_IPC_FAILURE`	An invalid IPC operation occurred.
`IS_ERR_INVALID_CIRCUIT`	The specified circuit did not match any of the CICs configured . Change the incorrect CIC value to an existing CIC and reissue the function call.
`IS_ERR_LIMIT EXCEEDED`	The maximum allowable number of remote SSPs has already been configured.
`IS_ERR_NO_ROUTESET`	The specified remote SSP does not have a route set configured for it.
`IS_ERR_PC_OWNED`	Another user has configured the specified remote SSP as its own.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_alloc_msg()

SYNOPSIS

```
isup_msg_t  *isup_alloc_msg (U8 prim_ev);
```

DESCRIPTION

The `isup_alloc_msg()` function allocates memory for an ISUP-layer message. The ISUP services message structure allocated is initialized for use to output a message with the specified primitive event code at the ISUP-layer interface. The application should call this function before calling `isup_put_msg()`.

Note that the ISUP services message structure (`isup_msg_t`) contains optional fields (`parm_ptr` and `parm_len`) for passing any proprietary parameters. The structure also contains the field `xtype_b` to specify the role of a type B (transit) exchange for a connection-setup event or a status event operation. These fields are initialized to zero for the default case where no proprietary parameters exist and the exchange type is an endpoint in the network, not a type B transit exchange. If you specify any outbound proprietary parameters, you must manage the memory associated with their use. See the `isup_put_msg()` function for more information.

PARAMETERS

* *`prim_ev`* (input)
  Specifies the primitive event code of the ISUP services message structure. If you are sending an Interprocess Communications (IPC) message, specify the value `0` for this parameter.

RETURN VALUES

If successful, the `isup_alloc_msg()` function call returns a pointer to the `isup_msg_t` structure that it allocated for the ISUP message.

If an error occurs, the function call returns `NULL` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_MEMORY_NOT_AVAIL`	No memory is available.
`IS_ERR_PRIMITIVE_EVENT`	The specified primitive is not valid.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_cfg_circuit_info()

SYNOPSIS

```
int isup_cfg_circuit_info(cirId circuit,
                          SiCirGrpCharInd * cirGcInd);
```

DESCRIPTION

The `isup_cfg_circuit_info()` function allows an ISUP services application to modify one or more default values of the Circuit Group Characteristic Indicator parameter contained in a Circuit Validation Response (CVR) message which is sent in response to a Circuit Validation Test (CVT) message. This function provides application control over the content of the circuit group characteristic indicator on a circuit-by-circuit basis. This function applies only to the ANSI network variant.

Calling the `isup_cfg_circuit_info()` function is optional. If your application does not call this function, the `isup_add_circuit()` function automatically populates the circuit group characteristic indicator in the circuit configuration structure with the default values.

> NOTE
>
> If the ISUP user application has defined a value for the Double seizing Control Indicator field of the Circuit Group Characteristics Indicator parameter through the environment variable `ISUP_DBL_SEIZE_BITS`, using the `isup_cfg_circuit_info()` function to define the double seizing control indicator overrides the value specified by the environment variable.

PARAMETERS

* *`circuit`* `(input)`
Specifies the circuit index number of the configured circuit.

* *`cirGCInd`* `(input)`
The name of a `SiCiriGrpCharInd` structure that contains the *circuit group characteristic indicator* parameter described in Section 3.10A of ANSI Standard T1.113.

This parameter specifies the values to be used for the following four fields of the outgoing circuit group indicator parameter.

`cirGCInd->cirGrpCarInd.val`	Circuit group characteristics indicator
`cirGCInd->dblSzCtrlInd.val`	Double seizing control indicator
`cirGCInd->alarmCarInd.val`	Alarm carrier indicator
`cirGCInd->contChkReqInd.val`	Continuity check requirements indicator

The input pointed to specifies whether or not the field is being changed.

`cirGCInd->cirGrpCarInd.pres`	Circuit group characteristics indicator
`cirGCInd->dblSzCtrlInd.pres`	Double seizing control indicator
`cirGCInd->alarmCarInd.pres`	Alarm carrier indicator
`cirGCInd->contChkReqInd.pres`	Continuity check requirements indicator

Each subfield specifies either `TRUE` or `FALSE` to specify that the field's value is being changed or that the field should be left at its existing value.

The default values for each field of this parameter are as follows:

```
cirGCInd->cirGrpCarInd.val =CG_DIGITAL (0x02)
cirGCInd->dblSzCtrlInd.val=DS_UNKNOWN (0x00)
cirGCInd->alarmCarInd.val=AC_HARDCARHAND (0x02)
cirGCInd->contChkReqInd.val=CO_NONE (0x01)
```

RETURN VALUES

If successful, the `isup_cfg_circuit_info()` function call returns the value `zero` (0).

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_INVALID_CIRCUIT`	The circuit specified did not match any of the CICs configured. Change the incorrect CIC value to an existing CIC and reissue the function call.

Error Code	Description
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_CONFIG_PARAMETER`	One or more of the values specified for the circuit group indicator parameter are invalid. Correct the value(s) before calling this function.
`IS_ERR_NOT_SUPPORTED`	This function is not supported under this ISUP version. Configure the SINAP node to the correct ISUP version, restart the SINAP node and restart the application.

isup_cfg_process()

SYNOPSIS

```
int  isup_cfg_process (int max_rem_ssps,
                        int max_cir_grps,
                        int max_circuits);
```

DESCRIPTION

The `isup_cfg_process()` function configures an ISUP services application and identifies the application to the ISUP services manager. Each user process that is part of an ISUP services application must call this function after registering with the SINAP node and before calling any other ISUP services function.

The `isup_cfg_process()` function defines the maximum configuration parameters (such as remote SSPs and circuits) for the application's user process. Thereafter, the process cannot configure more remote SSPs or circuit groups than the maximum values specified by this function call.

Note that the `isup_cfg_process()` function enforces the maximum values defined for each user process in the `sinap.h` include file. For example, `MAX_ISUP_REM_SSPS` indicates that a single user process can manage up to 256 remote SSPs as long as the total number of point codes does not exceed 2048. Therefore, an application with 15 user processes can manage up to 2048 remote SSPs. (Remember, an ISUP services application requiring SS7 services can only have 15 user processes.) Similarly, `MAX_ISUP_CIRCUITS` specifies that a single application user process can handle a maximum of 20,000 circuits or Circuit Identification Codes (CICs). Since each SINAP node can have up to 15 processes, a single SINAP node can handle a maximum of 300,000 circuits.

NOTE

If you experience problems with an ISUP application concerning user heap space, refer to the appropriate section in Chapter 2 for information on configuring tunable kernel parameters for your system.

PARAMETERS

* `max_rem_ssps` (input)
  Specifies the maximum number of remote SSPs (up to 256) that this ISUP services application user process can handle.

* `max_cir_grps` (input)
  Specifies the maximum number of circuit groups (up to 840) that this ISUP services application user process can handle. The `ISUP_MAX_CIR_GRPS` parameter in the `sinap.h` include file defines a maximum of 840 circuit groups. The `isup_cfg_process()` function enforces this maximum value.

* `max_circuits` (input)
  Specifies the maximum number of circuits (up to 20,000) that this ISUP services application user process can handle.

RETURN VALUES

If successful, the `isup_cfg_process()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
IS_ERR_ALREADY_CONFIG	The calling process is already configured for ISUP services.
IS_ERR_CONFIG_FORMAT	An invalid configuration value was present in an internal message used for this command.
IS_ERR_CONFIG_PARAMETER	A specified parameter value is too large.
IS_ERR_INTERNAL_FAILURE	An internal configuration error occurred.
IS_ERR_ISMG_IPC_FAILURE	An invalid IPC operation occurred.
IS_ERR_MEMORY_NOT_AVAIL	No memory is available with which to initialize the ISUP configuration parameters for the calling process.
IS_ERR_NOT_REGISTERED	The calling process is not registered for ISUP services. Call `ca_register()` before calling this function.
IS_ERR_ALREADY_CONFIG	Not enough memory available to add the requested point code(s).

isup_change_pc_mode()

SYNOPSIS

```
isup_change_pc_mode(intssp_pc, init mode)
```

DESCRIPTION

When the DLPC feature is configured on a SINAP node, the `isup_change_pc_mode()` function allows an ISUP user application to change the point code mode of a registered point code from active to standby or from standby to active. The requested mode is the *target* mode. If the DLPC feature is not configured, the application should not call this function.

When the application calls the `isup_change_pc_mode()` function, the SINAP node sends the request to the other ISUP manager through an interprocess communications (IPC) message (`ISUP_CHANGE_PC_MODE`). The ISUP manager returns the `ISUP_CHANGE_PC_MODE` IPC message to indicate success or failure. If the returned IPC message contains an error code, the function stores it in the global `errno` and returns the error code `RET_ERR` to the user application.

The `isup_change_pc_mode()` function tests the requested target point code mode. If the application change request specifies a point code mode that is the same as the mode currently defined for the point code, this function returns the error code `RET_ERR` to the user application

Each time the point code mode changes as the result of a failover or failback, the SINAP node updates the point code mode in the local point code table. The point code mode change request contained in the IPC message can come from either the application or the ISUP manager. The ISUP manager can send an IPC message to notify the application that a point code is switching states. When the IPC arrives in the function `isup_get_message()`, the SINAP node stores the target mode in the local point code table. If the request is to change the point code's mode from active to standby, the SINAP node resets the circuits associated with that point code to and `IDLE` state and stops all circuit timers.

PARAMETERS

* `ssp_pc` (input)
  Specifies the registered point code of the remote SSP that requires a point code mode change.

* `mode` (input)
  Specifies the target mode as either active or standby.

RETURN VALUES

If successful, the `isup_change_pc_mode()` function call returns the value zero (0).

If an error occurs, the function call returns the value `RET_ERR` (`-1`) and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_INVALID_CIRCUIT`	The circuit specified did not match any of the CICx configured. Change the incorrect CIC value to an existing CIC and reissue the function call.
`IS_ERR_PC_WRONG_MODE`	The application change request specified a point code mode change to a state that already exists for the point code. Reissue the function call using a different mode.

isup_del_circuit()

SYNOPSIS

```
int  isup_del_circuit (CirId circuit);
```

DESCRIPTION

The `isup_del_circuit()` function deletes a circuit from the configuration of the calling ISUP services application. Note that the specified circuit cannot be busy with an ISUP connection. If it is busy with a connection, the function call cannot execute because a circuit that is currently in use cannot be deleted. Also, the specified circuit must be one the application is configured to handle. Otherwise, an error results.

PARAMETERS

* *`circuit`* (input)
  Specifies the circuit index number of the circuit being deleted.

RETURN VALUES

If successful, the `isup_del_circuit()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_CIRCUIT_IS_BUSY`	The specified circuit is currently being used, and you cannot delete a circuit while it is being used.
`IS_ERR_INVALID_CIRCUIT`	The specified circuit is not valid.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_del_rem_ssp()

SYNOPSIS

```
int  isup_del_rem_ssp (int ssp_pc);
```

DESCRIPTION

The `isup_del_rem_ssp()` function deletes a remote SSP from the configuration of the calling ISUP services application. You cannot delete a remote SSP that has one or more circuits configured for it. You must first delete the circuit(s), then delete the remote SSP. Also, the specified remote SSP must be one that the user process is configured to handle. Otherwise, an error results.

PARAMETERS

* *ssp_pc* (input)
  Specifies the point code of the remote SSP being deleted.

RETURN VALUES

If successful, the `isup_del_rem_ssp()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
IS_ERR_CIRCUIT_DEFINED	A circuit is defined for the specified remote SSP, and you cannot delete the SSP without first deleting the circuit.
IS_ERR_ISMG_IPC_FAILURE	An invalid IPC operation occurred.
IS_ERR_PROC_NOTCONFIG	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
IS_ERR_UNKNOWN_REM_SSP	The specified remote SSP is not valid.

isup_get_circuit_status()

SYNOPSIS

```
#include <sys/types.h>
#include "caslinc.h"
#include "issl.h"
int isup_get_circuit_status(CirId circuit
                            isup_status_t* status);
```

DESCRIPTION

The `isup_get_circuit_status()` function allows an ISUP services application to request the current state and status of individual circuits that the ISUP services protocol maintains. The application specifies the Circuit Identification Code (CIC) of the desired circuit and specifies a pointer to an area in memory to hold the requested status information. The circuit identifier (`CirId` defined in the file `$SINAP_HOME/Include/issl_gen.h`) is used to look up internal state table of the ISUP services protocol. The state and status data is copied from the internal tables to the `isup_status_t` structure pointed to by the ISUP application user process requesting the CIC status.

> **NOTE**
>
> An ISUP services application uses additional heap space equal to the size of the `isup_status_t` structure. If you experience problems with your application related to user heap space, see the *SINAP/SS7 User's Guide* (R8051) for information on increasing user heap space and configuring the tunable kernel parameter for your system.

An ISUP application must configure the ISUP services protocol before issuing the `isup_get_circuit_status()` function by issuing a series of function calls such as `isup_cfg_process()`, `isup_add_rem_ssp()`, and `isup_add_circuit()`.

PARAMETERS

* *`circuit`* (input)
  Specifies the circuit index number for a particular circuit. This value must be unique among all of the application's circuits.

* *`status`* (input)
  Specifies the state and status of the requested circuit.

RETURN VALUES

If successful, the `isup_get_circuit_status()` function returns the value `RET_OK (0)`.

If an error occurs, the `isup_get_circuit_status()` function returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_INVALID_CIRCUIT`	The circuit specified did not match any of the CICs configured. Change the incorrect CIC value to an existing CIC, and reissue the function call.
`IS_ERR_PROC_NOTCONFIG`	The ISUP services protocol has not been configured for use. Issue the correct configuration functions calls to configure the CIC(s) and reissue the call.

isup_purge_circuit()

SYNOPSIS

```
int isup_purge_circuit(cir)
```

DESCRIPTION

The `isup_purge_circuit()` function sets the requested circuit to the IDLE state and terminates any outstanding timers for the circuit. Call this function when an event occurs that is outside the control of the SINAP node or the ISUP application and it causes the circuit to be in a condition that cannot be corrected. Such a condition might occur if, for example, an SSP is removed from the network while the ISUP application attempts to communicate with that SSP. This function applies to either DLPC or non-DLPC ISUP user applications.

CAUTION

Use this function with extreme care since no warning is presented and no confirmation is requested for resetting the circuit state. Purging a circuit can cause outstanding calls on the circuit to be dropped and may produce other unpredictable results and potentially serious problems in the network.

PARAMETERS

* *`cir`* (input)
  Specifies the Circuit Identification Code (CIC) of the circuit to be reset to the `IDLE` state.

RETURN VALUES

If successful, the `isup_purge_circuit()` function returns the value `RET_OK (0)`.

If an error occurs, the `isup_purge_circuit()` function returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_INVALID_CIRCUIT`	The specified circuit did not match any of the configured CICs. Change the incorrect CIC value to an existing CIC, and reissue the function call.

isup_restore_circuit_state()

SYNOPSIS

```
int isup_restore_circuit_state(circuit, procstat, blkFlags)
```

DESCRIPTION

The `isup_restore_circuit_state()` function call allows the ISUP user application to modify a specified circuit state to restore a previous state. If the circuit is configured, this function stops all circuit timers, initializes the circuit blocking state to `IDLE`, then sets the specified states. Call this function to re-create a state that was previously in effect for a process that failed or changed to a standby state.

NOTE

The ISUP user application is responsible for tracking the call processing states. You should determine whether or not any calls are in progress on the circuit, and if there are calls in progress, whether the calls are incoming or outgoing and whether the calls are blocked or unblocked.

PARAMETERS

* *`circuit`* (input)
  Specifies the Circuit Identification Code (CIC) of the circuit. This value must be unique among all of the application circuits.

* *`procStat`* (input)
  Defines one of the following call processing states to be set for the specified circuit.

Processing State Name	Value	Description
`INCBUSY`	0x04	Incoming Busy
`OUTBUSY`	0x08	Outgoing Busy
`CALL_IDLE`	0x0C	Idle

* `blkFlags` (input)
Defines one of the following blobking flags to be set for the specified circuit.

BLOCK Name	Value	Description
FL_NOBLK	0x00	Not blocked
FL_LOCMBLK	0x01	Local maintenance block
FL_RMTMBLK	0x02	Remote maintenance block

Defines one of the following blobking flags to be set for the specified circuit.

Call Processing State Name	Block Name
INCBUSY	(FL_LOCMBLK, FL_RMTMBLK) or FL_LOCMBLK FL_RMTMBLK) or FL_NOBLK
OUTBUSY	(FL_LOCMBLK, FL_RMTMBLK) or FL_LOCMBLK FL_RMTMBLK) or FL_NOBLK
CALL_IDLE	(FL_LOCMBLK, FL_RMTMBLK) or FL_NOBLK, FL_UNEQUIP

Certain blocking combinations are allowed. For either local or remote blocking, you cannot set the state to both maintenance and hardware blocked, but combinations of local and remote blocking are allowed. If `unequipped` is set, no other blocking flags can be specified and the call processing state must be set as `CALL_IDLE`. Calling the function `isup_restore_circuit_state()` with `CALL_IDLE` and `FL_NOBLK` is equivalent to calling `isup_purge_circuit()` for that circuit, except that `LPC_ROUTING` must be in effect to use the `isup_restore_circuit_state()` function. Currently, there is no functionality for restoring a `SUSPENDED` circuit state.

NOTE

In the ANSI network variant, setting of local or remote hardware blocking flags is not allowed.

RETURN VALUES

If successful, the `isup_restore_circuit_state()` function returns the value `RET_OK (0)`.

If an error occurs, the function returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	**Description**
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_INVALID_CIRCUIT`	The specified circuit did not match any of the configured CICs. Change the incorrect CIC value to an existing CIC, and reissue the function call.
`IS_ERR_INTERNAL_FAILURE`	A BUSY call processing state structure error occurred. Try reissuing the function call.
`IS_ERR_INVALID_CIR_STATE`	The circuit state is invalid.
`IS_ERR_LPCR_NOTFIG`	The `LPCR_ROUTING` option is not configured on the node.

isup_free_msg()

SYNOPSIS

```
int  isup_free_msg (isup_msg_t *msg);
```

DESCRIPTION

The `isup_free_msg()` function frees up the memory for an ISUP-layer message. Call this function to free the memory of a message structure used to input an ISUP message. If you plan to reuse the message structure to output a response message, you need not call this function. **Do not call this function** to free the memory for a message structure that is used to output a message. The internal ISUP services software will free the structure. Also, **do not modify fields marked `INTERNAL USAGE ONLY`** in the ISUP services message structure. These fields assist the ISUP Services Support Library (ISSL) in cleaning up memory space for any inbound proprietary parameters.

PARAMETERS

* *`msg`* (input)
  Specifies a pointer to the message structure whose memory you want to free.

RETURN VALUES

If successful, the `isup_free_msg()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_MEMORY_NOT_AVAIL`	No memory is available to free.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_get_msg()

SYNOPSIS

```
isup_msg_t  *isup_get_msg (BOOL wait);
```

DESCRIPTION

The `isup_get_msg()` function retrieves an incoming ISUP message from the ISUP-layer interface. If no message is presently available, the function's wait parameter specifies whether to wait for a message or return immediately. When a message is available, the type of the message input operation is notified to the user by

```
msg->hdr.prim_ev = primitive_event_code
```

in the message header. If zero, the notified input operation is the reception of the Interprocess Communications (IPC) message whose `i_block_t` structure is indicated by the pointer

```
msg->ev.ipc = IPC_message_block
```

in the message. Otherwise, the notified input operation is the invocation of the internal ISUP function for the primitive associated with the primitive event code. The circuit to which the input operation applies is supplied by the internal ISUP function, and this is indicated to the user by

```
msg->hdr.circuit = circuit_index_number
```

in the message header. If there are any proprietary parameters for the input operation, they are indicated to the user by

```
msg->hdr.parm_ptr = proprietary parameters pointer
msg->hdr.parm_len = proprietary parameters length
```

in the message header. If there are no proprietary parameters, these items are set to zero. Note that these items are initialized to zero for the default case where there are no proprietary parameters. When proprietary parameters are received, the ISSL library manages the memory required for the parameters. In this case, do not attempt to free this memory. Also, do not modify the `parm_ptr` or `parm_len` fields.

NOTE

ISUP services support for proprietary parameters only applies to the CCITT (ITU) network variant of the SINAP/SS7 system, where the appropriate protocol procedures are defined together with use of the Parameter Compatibility Information parameter in the ISUP message.

The internal ISUP function also supplies the appropriate event structure (if any) for the input operation. When present, this is indicated to the user by one of the following pointers in the message:

`msg->ev.con` = Connection Setup event

`msg->ev.cnst` = Connection Status event

`msg->ev.info` = User Information event

`msg->ev.fac` = Call Facility event

`msg->ev.susp` = Call Suspend event

`msg->ev.resm` = Call Resume event

`msg->ev.rel` = Connection Release event

`msg->ev.sta` = Circuit Status event

`msg->ev.tid` U32 timer ID (instead of an event structure pointer for user timer expiration)

The following chart lists each input operation primitive, along with the associated primitive event code and the corresponding event structure.

Input Operation Primitive	Primitive Event Code	Event Structure
Call Facility Confirmation	EVTSITFACCFM	SiFacEvnt
Call Facility Indication	EVTSITFACIND	SiFacEvnt
Call Resume Indication	EVTSITRESMIND	SiResmEvnt
Call Suspend Indication	EVTSITSUSPIND	SiSuspEvnt
Circuit Status Indication	EVTSITSTAIND	SiStaEvnt
Connection Release Confirmation	EVTSITRELCFM	SiRelEvnt
Connection Release Indication	EVTSITRELIND	SiRelEvnt
Connection Setup Confirmation	EVTSITCONCFM	SiConEvnt

Input Operation Primitive	Primitive Event Code	Event Structure
Connection Setup Indication	`EVTSITCONIND`	`SiConEvnt`
Connection Status Indication	`EVTSITCNSTIND`	`SiCnStEvnt`
User Information Indication	`EVTSITDATIND`	`SiInfoEvnt`
Unsolicited Status Indication	`EVTLSISTAIND`	†None

† The `msg->ev.tid` contains a U32 timer ID (instead of an event structure pointer) for a user timer expiration.

If the input operation primitive is a Connection Status, Circuit Status, or Unsolicited Status Indication, the particular operation is indicated to the user by

`msg->hdr.ev_type =` *`event_type`* (for primitive)

in the message header. This is supplied by the internal ISUP services function for the particular Status Indication primitive.

The following chart lists the Connection Status event types that can be returned, and the corresponding connection input operation it indicates.

Event Type	Connection Input Operation
`ADDRCMPLT`	Address complete
`CALLINGCLR`	Calling Party Clear [China only]
`EXIT`	Exit [ANSI only]
`FRWDTRSFR`	Forward transfer [not for Japan]
`IDENTREQ`	Identification request [CCITT only]
`IDENTRSP`	Identification response [CCITT only]
`INFORMATION`	Information [not for Japan]
`INFORMATREQ`	Information request [not for Japan]
`METERPULSE`	Metering Pulse [China only]
`MODCMPLT`	Call modify complete [CCITT only]
`MODIFY`	Call modify [CCITT only]
`MODREJ`	Call modify reject [CCITT only]
`NETRESMGT`	Network resource management [CCITT only]

Event Type	Connection Input Operation
`OPERATORSIG`	Operator Signal [China only]
`PROGRESS`	Call progress
`SUBSADDR`	Subsequent address [CCITT only]

The Circuit Status event types pertain to circuit management operations. The following chart lists each Circuit Status event type that can be returned, and the corresponding management input operation it indicates.

Event Type	Management Input Operation
`CIRBLKREQ`	Circuit block request
`CIRBLKRSP`	Circuit block response
`CIRUNBLKREQ`	Circuit unblock request
`CIRUNBLKRSP`	Circuit unblock response
`CIRRESREQ`	Circuit reset request
`CIRRESREQLOC`	Circuit reset request (local)
`CIRGRPBLKREQ`	Circuit group block request [not for Japan]
`CIRGRPBLKRSP`	Circuit group block response [not for Japan]
`CIRGRPUNBLKREQ`	Circuit group unblock request [not for Japan]
`CIRGRPUNBLKRSP`	Circuit group unblock response [not for Japan]
`CIRGRPQRYRSP`	Circuit group query response
`CIRGRPRESREQ`	Circuit group reset request.
`CIRGRPRESACK`	Circuit group reset acknowledge
`CIRUNEQPD`	Circuit unequipped [ANSI only]
`CONTCHK`	Continuity check [not for Japan]
`CONTREP`	Continuity report [not for Japan]
`STPCONTIN`	Stop continuity [not for Japan]
`CIRRESERVE`	Circuit reservation [ANSI only]
`CIRRESERVEACK`	Circuit reservation acknowledge [ANSI only]
`LOOPBCKACK`	Loop-back acknowledge [not for Japan]

Event Type	Management Input Operation
CONFUSION	Confusion [not for Japan]
REATTMEMP	Re-attempt indication
ERRORIND	Error indication
CHGINFO	Charge information [Japan only]

The Unsolicited Status event types pertain to alarms that are generated to report events that are either unusual or not related to the exchange of ISUP protocol messages. The following chart lists each Unsolicited Status event type that can be returned, and the corresponding alarm status that it indicates.

Event Type	Alarm Status Indication
CIR_IN_SERV	Circuit back in service
CIR_OUT_ORD	Circuit out of order
CIR_UNEQUIP	Circuit unequipped [ANSI only]
CIR_VAL_FAIL	Circuit validation failed [ANSI only]
CIR_VAL_SUCC	Circuit validation success [ANSI only]
CIRMGT_NORESP	Circuit management no response
CONG_LVL_1	ISUP congestion level 1 [not for Japan]
CONG_LVL_2	ISUP congestion level 2 [not for Japan]
RLC_RCVD	Release Complete (RLC) message received
SICIR_CFG_OK	Circuit configured OK
SIMTP_CONG	MTP congestion indication
SIMTP_PAUSE	MTP pause indication
SIMTP_RESUME	MTP resume indication
SIMTP_RMTUSRAV	MTP remote user available [CCITT only]
SIMTP_RMTUSRUNAV	MTP remote user unavailable [CCITT only]
SIMTP_STPCONG	MTP stop congestion indication
UTIMER_EXPIRY	User timer expiration

PARAMETERS

* `wait` (input)
Specifies whether the function is to wait for a message: the value `TRUE` indicates the function is to wait for a message; the value `FALSE` indicates the function is to return if there are no messages. If you use the value `FALSE`, the function returns with `errno` set to `CA_ERR_NO_MSUS` when there are no messages.

RETURN VALUES

If successful, the `isup_get_msg()` function call returns a pointer to the `isup_msg_t` structure containing the ISUP message.

If an error occurs, the function call returns `NULL` and `errno` is set to one of the following error codes.

Error Code	Description
`CA_ERR_NO_MSUS`	No ISUP messages are available.
`EINTR`	SINAP node detected a signal.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_purge_circuit()

SYNOPSIS

```
int  isup_purge_circuit (cir);
```

DESCRIPTION

The `isup_purge_circuit()` function unconditionally purges (deletes) a specified circuit. This function sets the requested circuit to the `IDLE` state and terminates any outstanding timers for the circuit.

CAUTION

Use this function with extreme care since no warning is presented and no confirmation is requested for resetting the circuit state. Purging a circuit can cause outstanding calls on the circuit to be dropped and may produce other unpredictable results and potentially serious problems in the network.

PARAMETERS

* *`cir`* (input)
  Specifies the Circuit Identification Code (CIC) of the circuit to be purged.

RETURN VALUES

If successful, the `isup_purge_circuit()` function call returns the value `RET_OK (0)`. If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_INVALID_CIRCUIT`	The specified circuit did not match any of the configured CICs. Change the incorrect CIC value to an existing CIC, and reissue the function call.

Error Code	Description
IS_ERR_PROC_NOTCONFIG	The calling process is not configured for ISUP services. Call isup_cfg_process() before calling this function.

isup_put_msg()

SYNOPSIS

```
int  isup_put_msg (isup_msg_t *msg);
```

DESCRIPTION

The `isup_put_msg()` function outputs a message at the ISUP-layer interface. The type of message output operation is specified by

`msg->hdr.prim_ev` = *`primitive_event_code`*

in the message header. If zero, the output operation performed is to send the Interprocess Communications (IPC) message whose `i_block_t` structure is specified by the pointer:

`msg->ev.ipc` = *`IPC_message_block`*

in the message. Otherwise, the output operation performed is to invoke the internal ISUP services function for the primitive associated with the primitive event code. The circuit to which the output operation applies is specified by

`msg->hdr.circuit` = *`circuit_index_number`*

in the message header. If there are any proprietary parameters for the output operation, they are indicated to the user by

```
msg->hdr.parm_ptr = proprietary parameters pointer
msg->hdr.parm_len = proprietary parameters length
```

in the message header. Otherwise, these items are set to zero. Note that these items are initialized to zero for the default case where no proprietary parameters exist. When proprietary parameters are used, note that you must manage the memory required for the parameters.

If the exchange is a type B transit exchange, the role of a type B (transit) exchange for a connection-setup or status event is specified by a setting of `TRUE` for:

```
msg->hdr.xtype_b = exchange type B (transit)
```

in the message header. Otherwise this field is set to zero. Note that this field is initialized to zero for the default case where the exchange type is an endpoint in the network (not a type B transit exchange).

NOTE

ISUP services support of proprietary parameters applies only to the CCITT (ITU) network variant of the SINAP/SS7 software where appropriate protocol procedures are defined together with use of the Parameter Compatibility Information parameter in the ISUP message.

The circuit and any optional parameters are passed to the internal ISUP services function along with the appropriate event structure for the input operation, which is specified by one of the following pointers in the message:

`msg->ev.con` = Connection Setup event

`msg->ev.cnst` = Connection Status event

`msg->ev.info` = User Information event

`msg->ev.fac` = Call Facility event

`msg->ev.susp` = Call Suspend event

`msg->ev.resm` = Call Resume event

`msg->ev.rel` = Connection Release event

`msg->ev.sta` = Circuit Status event

The following chart lists each output operation primitive, along with the associated primitive event code and the corresponding event structure.

Output Operation Primitive	Primitive Event Code	Event Structure
Call Facility Request	EVTSITFACREQ	SiFacEvnt
Call Facility Response	EVTSITFACRSP	SiFacEvnt
Call Resume Request	EVTSITRESMREQ	SiRelEvnt
Call Suspend Request	EVTSITSUSPREQ	SiSuspEvnt
Circuit Status Request	EVTSITSTAREQ	SiStaEvnt

Output Operation Primitive	Primitive Event Code	Event Structure
Connection Release Request	`EVTSITRELREQ`	`SiRelEvnt`
Connection Release Response	`EVTSITRELRSP`	`SiRelEvnt`
Connection Setup Request	`EVTSITCONREQ`	`SiConEvnt`
Connection Setup Response	`EVTSITCONRSP`	`SiConEvnt`
Connection Status Request	`EVTSITCNSTREQ`	`SiCnStEvnt`
User Information Request	`EVTSITDATREQ`	`SiInfoEvnt`

If the output operation primitive is a Connection Status or Circuit Status Request, the particular output operation is specified by:

`msg->hdr.ev_type =` *`event_type`* (for primitive)

in the message header. This is passed to the internal ISUP function for the `Connection Status` or `Circuit Status Request` primitive.

The following chart lists each Connection Status event type that can be specified, and the corresponding connection output operation it invokes.

Event Type	Connection Output Operation
`ADDRCMPLT`	Address complete
`EXIT`	Exit [ANSI only]
`FRWDTRSFR`	Forward transfer [not for Japan]
`IDENTREQ`	Identification request [CCITT only]
`IDENTRSP`	Identification response [CCITT only]
`INFORMATION`	Information [not for Japan]
`INFORMATREQ`	Information request [not for Japan]
`MODCMPLT`	Call modify complete [CCITT only]
`MODIFY`	Call modify request [CCITT only]
`MODREJ`	Call modify reject [CCITT only]
`NETRESMGT`	Network resource management [CCITT only]
`PROGRESS`	Call progress
`SUBSADDR`	Subsequent address [CCITT only]

The Circuit Status event types pertain to circuit management operations. The following chart lists each Circuit Status event type that can be specified, and the corresponding management output operation it invokes.

Event Type	Management Output Operation
CALLINGCLR	Calling Party Clear [China only]
CHGINFO	Charge information [Japan only]
CIRBLKREQ	Circuit block request
CIRBLKRSP	Circuit block response
CIRGRPBLKREQ	Circuit group block request [not for Japan]
CIRGRPBLKRSP	Circuit group block response [not for Japan]
CIRGRPQRYREQ	Circuit group query request
CIRGRPQRYRSP	Circuit group query response
CIRGRPRESREQ	Circuit group reset request
CIRGRPUNBLKREQ	Circuit group unblock request [not for Japan]
CIRGRPUNBLKRSP	Circuit group unblock response
CIRRESERVE	Circuit reservation [ANSI only]
CIRRESERVEACK	Circuit reservation acknowledge [ANSI only]
CIRRESREQ	Circuit reset request
CIRUNBLKREQ	Circuit unblock request
CIRUNBLKRSP	Circuit unblock response
CONTCHK	Continuity check [not for Japan]
CONTREP	Continuity report [not for Japan]
LOOPBCKACK	Loop-back acknowledge [not for Japan]
METERPULSE	Metering Pulse [China only]
OPERATORSIG	Operator Signal [China only]

PARAMETERS

* *msg* (input)
  ISUP services message structure pointer for output.

RETURN VALUES

If successful, the `isup_put_msg()` function call returns the value `RET_OK (0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_INVALID_CIRCUIT`	The specified circuit is not valid.
`IS_ERR_PRIMITIVE_EVENT`	The specified primitive is not valid.
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.

isup_start_utimer()

SYNOPSIS

```
uintptr_t isup_start_utimer (CirId circuit,
                             int  duration);
```

DESCRIPTION

The `isup_start_utimer()` function call starts a user timer for an ISUP circuit. Each call to this function starts a new instance of a user timer. So, it is possible to have multiple user timers running concurrently for a circuit. The circuit must be one that the application process has configured. The duration of the user timer must be a positive value that does not exceed the maximum timer duration value of `43200` (12 hours defined in one-second ticks), defined by the `MAX_UTIMER_DURATION` parameter within the `issl.h` file. The data type `uintptr_t` is defined as`long` (8 bytes) in 64-bit mode, and `int` (4 bytes) in 32-bit mode.

If a user timer is not stopped via the `isup_stop_utimer()` function call, the SINAP/SS7 system sends a timer expiration message to the ISUP application when the user timer expires.

PARAMETERS

* *`circuit`* (input)
  Specifies the circuit index number of the configured circuit. This value must be unique among all of the application's circuits.

* *`duration`* (input)
  Specifies the duration of the user timer in one-second ticks.

RETURN VALUES

If successful, the `isup_start_utimer()` function call returns the timer ID that the SINAP/SS7 software assigned to the user timer when the timer was started.

If an error occurs, the function call returns the value zero `(0)` and `errno` is set to one of the following error codes.

Error Code	Description
`IS_ERR_PROC_NOTCONFIG`	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
`IS_ERR_INVALID_CIRCUIT`	The circuit specified did not match any of the CICs configured. Change the incorrect CIC value to an existing CIC and reissue the function call.
`IS_ERR_UTIMER_DURATION`	The user timer has an invalid timer duration defined. Change the incorrect value specified for the timer duration to a positive value that does not exceed the value defined by the `MAX_UTIMER_DURATION` parameter in the `issl.h` file. Be sure to specify the duration in one-second ticks.
`IS_ERR_MEMORY_NOT_AVAIL`	No memory is available to initialize the ISUP user timer for the calling process.

isup_stop_utimer()

SYNOPSIS

```
int isup_stop_utimer (uintptr_t timer_id);
```

DESCRIPTION

The `isup_stop_utimer()` function stops a user timer that was previously started for a circuit. Stopping a user timer guarantees that no user timer expiration message is sent to the ISUP services application process. To avoid possible confusion regarding timer expirations, you should stop any active timers for a circuit before deleting the circuit from the configuration. The data type `uintptr_t` is defined as`long` (8 bytes) in 64-bit mode, and `int` (4 bytes) in 32-bit mode.

PARAMETERS

* *timer_id* (input)
  Specifies the timer ID assigned to the user timer when the timer was started.

RETURN VALUES

If successful, the `isup_stop_utimer()` function call returns the value `RET_OK(0)`.

If an error occurs, the function call returns the value `RET_ERR (-1)` and `errno` is set to one of the following error codes.

Error Code	Description
IS_ERR_PROC_NOTCONFIG	The calling process is not configured for ISUP services. Call `isup_cfg_process()` before calling this function.
IS_ERR_INVALID_UTIMER	The timer ID specified for the user timer is invalid. The timer ID may refer to a timer that was not previously started. In this case, start the user timer (using the `isup_start_utimer()` function). This error also occurs if the application process has already received a timer expiration message for the specified user timer.

ISUP Services Structures

This section describes the structures used by ISUP services applications to pass information. With the exception of `isup_msg_t`, which is defined in the include file `$SINAP_HOME/Include/issl.h`, all ISUP structures are defined in the include file `$SINAP_HOME/Include/issl_sit.h` (where `$SINAP_HOME` is the root SINAP/SS7 directory for your installation). The ISUP services structures are listed below, along with a brief description. Refer to the appropriate section for more information.

- `isup_msg_t` - the main ISUP structure, which is comprised of a number of nested structures used to pass information to a particular ISUP event.
- `isup_status_t` - contains both the value of the Circuit Identification Code (CIC) and the Circuit Index Number (`CirId`) for a particular circuit requested by the ISUP services application user process.
- `SiConEvnt` - provides ISUP connection-setup event information.
- `SiCnStEvnt` - provides ISUP connection-status event information.
- `SiInfoEvnt` - provides information for an ISUP user-information event.
- `SiFacEvnt` - provides ISUP call-facility event information.
- `SiSuspEvnt` - provides ISUP call-suspend event information.
- `SiResmEvnt` - provides ISUP call-resume event information.
- `SiRelEvnt` - provides ISUP connection-release event information.
- `SiStaEvnt` - provides ISUP circuit-status event information.

When sending an ISUP message to another application, your application must initialize the appropriate structure for the primitive type specified in the ISUP message. For example, when sending an ISUP message whose primitive type is `EVTSITSTAREQ` and whose event type is `CIRBLKREQ` (a circuit block request), your application must initialize applicable fields in the `SiStaEvnt` structure.

For information on which structure fields to initialize, see ITU-T Recommendation Q.763, ANSI Standard T1.113, or NTT SS7 ISDN User Part (ISUP) Specification NTT-Q762-a, NTT-Q763-a, and NTT-Q764-a.

NOTE

Each element in the event structures within this chapter is flagged in the left margin with **C** if it applies to the CCITT (ITU) ISUP version, with **A** if it applies to the ANSI ISUP version, and with **J** if it applies to the NTT ISUP version.

When an ISUP services application processes an incoming ISUP message, the structure fields will have been initialized according to the ISUP message from the remote application.

ISUP Services Message Structure (`isup_msg_t`)

The `isup_msg_t` structure is used to pass an ISUP message. The structure is defined in the `issl.h` include file, and its format is shown here.

```
typedef struct
{
    struct isup_msg_hdr          /* ISUP message header */
    {
   Chn    chain; /* [INTERNAL USAGE ONLY]  */
   CirId circuit; /* circuit index number   */
   U16   status;  /* [INTERNAL USAGE ONLY]  */
   U8    prim_ev;  /* primitive event code   */
    U8    ev_type; /* event type (only for
               "Status" primitives)    */
   U8    *parm_ptr;/* proprietary parms pointer */
    U16  parm_len;/* proprietary parms length  */
   U8    xtype_b;/* exchange type B (transit) */
   }             hdr;
    union                      /* event structure pointer  */
    {
        i_block_t *ipc;  /* IPC message block, etc.  */
        SiConEvnt  *con;  /* Connection Setup event   */
        SiCnStEvnt *cnst;  /* Connection Status event  */
        SiInfoEvnt *info; /* User Information event   */
        SiFacEvnt *fac;  /* Call Facility event      */
        SiSuspEvnt *susp; /* Call Suspend event       */
        SiResmEvnt *resm;  /* Call Resume event        */
        SiRelEvnt  *rel;  /* Connection Release event */
        SiStaEvnt  *sta; /* Circuit Status event     */
        uintptr_t tid; /* Timer ID of user timer   */
    }              ev;
}
isup_msg_t;
```

NOTE

The ISUP services message structure (`isup_msg_t`) contains two fields (`chain` and `circuit`) that are reserved for internal usage. Also, a range of IPC message type values is reserved for internal ISUP usage (see `iblock.h`) to distinguish them from the application-specific use of IPC messages.

A user timer expiration message contains the timer ID instead of an event structure pointer.

The following list describes the fields within the `isup_msg_t` structure.

* *`chain`*
  Reserved for internal use to allow the ISSL library to clean up memory space for any inbound proprietary. Do **not** modify it.

* *`circuit`*
  The circuit index number of the circuit to which the message applies.

* *status*
  Reserved for internal use to allow the ISSL library to clean up memory space for any inbound proprietary parameters. Do **not** modify it.

* *prim_ev*
  The primitive event code associated with the desired ISUP activity (for example, connection setup, call suspend, call resume, and so on).

* **parm_ptr*
  An optional pointer to an internal message buffer that contains any proprietary parameters to be passed in the message. This field is initialized to zero for the default case where no proprietary parameters exist.

* *parm_len*
  The length of the unformatted data contained in any proprietary parameters. This field is initialized to zero for the default case where no proprietary parameters exist.

* *xtype_b*
  Defines the role of a type B (transit) exchange for a connection-setup event or a status event output operation. This field is initialized to zero for the default case where the exchange type is an end point in the network, not a type B transit exchange.

* *ev_type*
  The event type for the specified primitive event code. This field is used for connection-status events, circuit-status events, and unsolicited status.

* **ipc*
  A pointer to an `i_block_t` structure containing a SINAP IPC message header, followed by any IPC data.

* **con*
  A pointer to a `SiConEvnt` structure that contains information for a connection-setup event.

* **cnst*
  A pointer to a `SiCnStEvnt` structure that contains information for a connection-status event.

* **info*
  A pointer to a `SiInfoEvnt` structure that contains information for an ISUP user-information event.

* **fac*
  A pointer to a `SiFacEvnt` structure that contains information for a call-facility event.

* **susp*
  A pointer to a `SiSuspEvnt` structure that contains information for a call-suspend event.

* **resm*
  A pointer to a `SiResmEvnt` structure that contains information for a call-resume event.

* **rel*
  A pointer to a `SiRelEvnt` structure that contains information for an ISUP connection-release event.

* **sta*
  A pointer to an `SiStaEvnt` structure that contains information for a circuit-status event.

* *tid*
  The timer ID of a user timer. The data type `uintptr_t` is defined as `long` (8 bytes) in 64-bit mode and `int` (4 bytes) in 32-bit mode.

CIC Status Structure (`isup_status_t`)

The `isup_status_t` structure is defined in the `issl.h` include file, and its format is shown here. The letters **C** and/or **A** shown to the left of each element in the structure indicate whether the element applies to the CCITT (ITU) or the ANSI ISUP version.

```
       typedef struct isup_status_s
       {
[- A C] Bool   circfg_contReq;     /* continuity check required [TRUE/FALSE]*/
[- A C] Bool   sicon_exchCalRef;   /* exchange of call references required */
[- A C] Bool   sicon_charge;       /* responsible for charging */
[- A C] Bool   sicon_end2end;      /* end to end connection */
[- A C] Bool   sicon_useSCCP;      /* use SCCP */
[- A C] Bool   sicon_u2uExplReq;   /* User to User requested */
[- A C] Bool   cirCtl;             /* circuit control */
[- A C] Bool   locHardFlg;         /* local hardware flag */
[- A C] Bool   rmtHardFlg;         /* remote hardware flag */
[- A C] Bool   noRspFlgToUp;       /* no response flag to Upper */
[- A C] Bool   noRspFlgToLw;       /* no response flag to Lower */
[- A C] Bool   congest;            /* current circuit congestion ssp status */
[- A C] Bool   flwCntrl;           /* flow control SSP marked prohibited (TRUE) */
[- A C] Bool   resFlag;            /* In the circuit reset mode */
[- A C] Swtch  swtch;              /* switch SINAP Variant flag */
[- A C] Addrs  sicon_dstAdr;       /* Destination Address */
[- A C] Addrs  sicon_srcAdr;       /* Source Address */
[- A C] Ticks  sicon_calDura;      /* call duration */
[- A C] Addrs  circfg_clli;        /* common language location identifier */
        /* circuit group is defined by ISUP application the first time a circuit
         group message is sent */
[- A C] Addrs  circfg_outTrkGrpN; /* outgoing trunk group number this circuit is
                                   part of */
[- A C] U8   circfg_bearProf;   /* bearer profile */
[- A C]      /* Transmission Medium Requirements [see Q.763 or ANSI T1.113.3] */
[- A C]      /* variable names defined in issl_lsi.h */
[- A C]      /* TMR_SPEECH     0x00 */
[- A C]      /* TMR_64KBITS    0x02 */
[- A C]      /* TMR_31KHZ      0x03 */
[- A C]      /* TMR_ALTSPEECH  0x04 */
[- A C]      /* TMR_ALT64KBITS 0x05 */
[- A C]      /* TMR_384KBITS   0x08 */
[- A C]      /* TMR_1536KBITS  0x09 */
[- A C]      /* TMR_1920KBITS  0x0A */
[- A C] U8   cirgrp_state;      /* circuit group state, this circuit is part of */
[- A C]  /* circuit group state values */
[- A C]      /* variable names defined in issl.h */
[- A C]      /* CGRSTIDLE        0  circuit group idle state */
[- A C]      /* CGRSTBLKACK      1  waiting for circuit group block ack */
[- A C]      /* CGRSTUNBLKACK    2  waiting for circuit group unblock ack */
[- A C]      /* CGRSTBLKRSP      3  waiting for circuit group block response */
[- A C]      /* CGRSTUNBLKRSP    4  waiting for circuit group unblock respse */
[- A C]      /* CGRSTRESACK      5  waiting for circuit group reset ack */
[- A C]      /* CGRSTBLKACKRSP   6  waiting for circuit group block response or
                               circuit group unblock ack  */
[- A C]      /* CGRSTUNBLKACKRSP 7  waiting for circuit group unblock response
                                   or circuit group unblock ack  */
[- A C] U8  icon_resDir;          /* direction of reset request */
[- A C]    /* direction of reset                     */
[- A C]    /* variable names defined in issl_gen.h    */
[- A C]    /* FROM_LWR        1  direction from lower */
[- A C]    /* FROM_UPR        2  direction from upper */
[- A C] U8 sicon_evntType;    /* Last Event Type  posted for this circuit */
[- A C] U8 transStat;         /* transient states */
[- A C]    /* transient states */
[- A C]    /* variable names defined in issl_lsi.h */
[- A C]    /* IDLE             0 idle       */
[- A C]    /* UNEQUIP          1 unequipped */
```

```
[- A C]    /* LOCBLKED         2 locally blocked */
[- A C]    /* BLKACK           3 waiting for blocking ack */
[- A C]    /* UNBLKACK         4 waiting for unblocking ack */
[- A C]    /* LOCBLKRSP        5 locally blocked, waiting for blocking response */
[- A C]    /* LOCUNBLKRSP      6 locally blocked, waiting for unblocking response */
[- A C]    /* RMTBLKACK        7 remotely blocked, waiting for blocking ack */
[- A C]    /* RMTUNBLKACK      8 remotely blocked, waiting for unblocking ack */
[- A C]    /* LOCRMTBCKED      9 locally and remotely blocked */
[- A C]    /* BLKACKRSP       10 waiting for blocking ack and blocking response */
[- A C]    /* UNBLKACKUNRSP   11 waiting for unblocking ack and unblocking response */
[- A C]    /* BLKACKUNRSP     12 waiting for blocking ack and unblocking response */
[- A C]    /* UNBLKACKRSP     13 waiting for unblocking ack and blocking response */
[- A C]    /* BLKRSP          14 waiting for blocking response */
[- A C]    /* UNBLKRSP        15 waiting for unblocking response */
[- A C]    /* RMTBLKED        16 remotely blocked */
[- A C] U8 calProcStat;        /* call processing states */
[- A C]    /* call processing states */
[- A C]    /* variable names defined in issl_lsi.h */
[- A C]    /* TRANS           0x00  transient */
[- A C]    /* COTCHK          0x01  continuity check  */
[- A C]    /* INCBUSY         0x04  incoming busy */
[- A C]    /* OUTBUSY         0x08  outgoing busy */
[- A C]    /* CALL_IDLE       0x0c  idle */
[- A C] U8 circfg_typeCntrl;   /* type of control */
[- A C]    /* Current direction allowed */
[- A C]    /* variable names defined in issl_lsi.h */
[- A C]    /* INCOMING           0    incoming */
[- A C]    /* OUTGOING           1    outgoing */
[- A C]    /* BOTHWAY            2    both ways circuit */
[- A C] U8 cirConInState:           /* Incoming Connection State */
[- A C] U8 cirConOutState:          /* Outgoing Connection State */
           /* connection call states */
[- A C]    /* variable names defined in issl.h */
[- A C]    /* CON_ST_IDLE        0x00 /* idle */
[- A C]    /* CON_ST_WTFORCONTIN 0x01 /* waiting for continuity */
[- A C]    /* CON_ST_WTFORACM    0x02 /* waiting for ACM */
[- A C]    /* CON_ST_WTFORANSWR  0x03 /* waiting for answer */
[- A C]    /* CON_ST_ANSWRD      0x04 /* answered */
[- A C]    /* CON_ST_SUSP        0x05 /* suspended */
[- A C]    /* CON_ST_WTFORRELCMP 0x06 /* waiting for release complete */
[- A C]    /* CON_ST_WTFORRELRSP 0x07 /* waiting for release resp. from upper layer */
[- A C]    /* CON_ST_WTFORRLCRRS 0x08 /* waiting for reset complete */
           /* !!! states above apply for both CCITT and ANSI to either */
           /*     incoming, outgoing, or bothway */
           /*     (see circfg_typeCntrl above)   */
[- A -]    /* CON_ST_WTFORCOTIAM  0x09   /* waiting for IAM and COT */
[- A -]    /* CON_ST_WTFORIAM     0x0a   /* waiting for IAM */
           /* !!!! states above apply to ANSI incoming calls only */
[- A -]    /* CON_ST_WTFORCOTCRA  0x09   /* waiting for continuity rep and CRA */
[- A -]    /* CON_ST_WTFORCOTREP  0x0a   /* waiting for COT have CRA */
[- A -]    /* CON_ST_WTFORCRA     0x0b   /* waiting for CRA */
[- A -]    /* CON_ST_WTFORDIGS    0x0c   /* waiting for digits */
           /*!!!! States above apply to ANSI outgoing calls only */
       } isup_status_t;
```

The following list describes the fields within the `isup_status_t` structure.

* *circfg_contReq*
  A boolean indicator that specifies whether or not (TRUE or FALSE) a continuity check is required for this circuit.

* *sicon_exchCalRef*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the exchange of call references is required for this circuit.

* *sicon_charge*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the call on this circuit is chargeable and should be timed.

* *sicon_end2end*
  A boolean indicator that specifies whether or not (TRUE or FALSE) end-to-end connection is established.

* *sicon_useSCCP*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the SCCP method is to be used for end-to-end transfer of information.

* *cirCtl*
  A boolean indicator that specifies whether or not (TRUE or FALSE) a release should be sent to the upper layer when a connection request is seen on this circuit while waiting on an answer for a call request.

* *locHardFlg*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the local hardware flag is set to indicate the ownership of the circuit.

  `TRUE` = The circuit is local to this box.

  `FALSE` = The circuit is not part of this box.

* *rmtHardFlg*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the remote hardware flag is set to indicate this circuit is remotely connected to this box.

* *noRspFlgToUp*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the no-response flag to the upper layer is set to indicate if a response should be sent to the upper layer (user application).

* *noRspFlgToLw*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the no-response flag to the lower layer is set to indicate if a response should be sent from upper layer to lower layer.

* *congest*
  A boolean indicator that specifies whether or not (TRUE or FALSE) the current status of the remote SSP is congested.

* *flwCntrl*
  A boolean indicator that specifies whether or not (TRUE or FALSE) remote SSP is marked prohibited. The default setting is `TRUE` (assumes the remote SSP is available).

* *resFlag*
  A boolean indicator that specifies whether or not (TRUE or FALSE) this circuit is in the circuit reset mode.

* *swtch*
  A flag that indicates the SINAP/SS7 network variant.

* *sicon_dstAdr*
  The destination address parameter contains the destination address for this connection.

* *sicon_srcAdr*
  The source address parameter contains the source address for this connection.

* *sicon_calDura*
  The call duration parameter specifies the amount of time this call has been connected.

* *circfg_clli*
  The name of the `circfg_clli` structure that contains the Common Language Location Identification (CLLI) code used to validate this circuit's switching office by town, state, and building.

* *circfg_outTrkGrpN*
  The name of the `circfg_outTrkGrpN` structure that contains the outgoing trunk group number to which this circuit belongs.

* *circfg_bearProf*
  The bearer profile for this circuit configuration that contains the transmission medium requirements for transfer mode and transfer rates. The variable names are defined in the header file `issl_lsi.h`.

* *cirgrp_state*
  The code (0-7) that represents the current state of the circuit group to which this circuit belongs. The variable names are defined in the `issl.h` header file.

* *sicon_resDir*
  The direction (from lower layer or from upper layer) of the reset request for this circuit connection. The variable names are defined in the `issl_gen.h` header file.

* *transStat*
  The code (0-16) that represents the current transient state of this circuit. The variable names for transient states are defined in the `issl_lsi.h` header file.

* *calProcStat*
The hexadecimal code that represents the current call processing state for this circuit.

* *circfg_typeCntrl*
The type of control that defines the current direction of calls (incoming, outgoing, or both ways) allowed on this circuit.

* *cirConInState*
The current state of the connection for incoming calls on this circuit. The variable names for connection states are defined in the `issl.h` file.

* *cirConOutState*
The current state of the connection for outgoing calls on this circuit. The variable names for connection states are defined in the `issl.h` file

Connection Setup Event Structure (`SiConEvnt`)

The `SiConEvnt` structure contains information for an ISUP connection-setup event. The following table presents the ISUP primitives and messages used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” indicates the type of action invoked by the primitive.

The column labeled “ISUP Message Table” indicates the ISUP messages associated with the primitive and the tables (in parentheses) in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and NTT Specification NTT-Q763-a in which the message is defined. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	ISUP Message Table ITU-T	ANSI	NTT	Direction
EVTSITCONREQ	Connection Setup Request	IAM (32) SGM (49)	IAM (14)	IAM (10)	Output
EVTSITCONIND	Connection Setup Indication	IAM (32) SGM (49)	IAM (14)		Input
EVTSITCONRSP	Connection Setup Response	ANM (22) CON (27) SGM (49)	ANM (6)	ANM (6)	Output
EVTSITCONCFM	Connection Setup Confirmation	ANM (22) CON (27) SGM (49)	ANM (6)		Input

NOTE

The handling of the CCITT Segmentation (SGM) message is transparent to the user interface.

The `SiConEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **`J, C,`** and/or **`A`** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT, or ANSI ISUP versions.

```
        typedef struct siConEvnt        /* Connection Setup Event */
      {

[J A C]  SiNatConInd natConInd;       /* Nature of conn indicators */
[J A C]  SiFwdCallInd fwdCallInd;     /* forward call indicators */
[J A C]  SiCgPtyCat cgPtyCat;         /* calling party category */
[J - C]  SiTxMedReq txMedReq;         /* transmission medium rqmt */
[- A -]  SiUsrServInfo usrServInfoA   /* user service info */
[J A C]  SiCdPtyNum cdPtyNum;         /* called party number */
[- A C]  SiTranNetSel tranNetSel;     /* transit network selection */
[- A -]  SiCallRefA callRefA          /* call reference */
[J - C]  SiCallRef callRef;           /* call reference */
[J A C]  SiCgPtyNum cgPtyNum;         /* calling party number */
[J - C]  SiOpFwdCalInd opFwdCalInd;   /* optional fwd call indicators */
[J A C]  SiRedirNum redirgNum;        /* redirecting number */
[J A C]  SiRedirInfo redirInfo;       /* redirection information */
[J - C]  SiCugIntCode cugIntCode;     /* closed group interlock code */
[- A -]  SiConnReqA connReqA;         /* connection request */
[- - C]  SiConnReq connReq;           /* connection request */
[J A C]  SiOrigCdNum origCdNum;       /* original called number */
[J - C]  SiUsr2UsrInfo usr2UsrInfo;   /* user to user information */
[J A C]  SiAccTrnspt accTrnspt;       /* access transport */
[- A -]  SiChargeNum chargeNum;       /* charge number */
[- A -]  SiOrigLineInf origLineInf;   /* originating line information */
[J - C]  SiUsrServInfo usrServInfo;   /* user service info */
[J - C]  SiUsr2UsrInd usr2UsrInd;     /* user to user indicators */
[- - C]  SiPropDly propDly;           /* propagation delay counter */
[- - C]  SiUsrServInfo usrServInfo1;  /* user service info prime */
[- - C]  SiNetSpecFacil netFac;       /* network specific facility */
[- - C]  SiSigPointCode orgPteCde;    /* originating ISC point code */
[- A C]  SiGenDigits genDigits;       /* generic digits */
[J - C]  SiGenNum genNmb;             /* generic number */
[- - C]  SiUsrTSrvInfo usrTSrvInfo;   /* user tele service info */
[- - C]  SiRemotOper remotOper;       /* remote operations */
[- - C]  SiParmCompInfo parmCom;      /* parameter compatibility info */
[- - C]  SiNotifInd notifInd;         /* notification indicator */
[- A -]  SiServiceCode servCode;      /* service code */
[- A -]  SiServiceAct serviceAct1;    /* service activation */
[- - C]  SiServiceAct serviceAct;     /* service activation */
[- - C]  SiMlppPrec mlppPrec;         /* MLPP precedence */
[- - C]  SiTxMedReq txMedUsPr;        /* transmission medium used */
[J A C]  SiBckCalInd bckCallInd;      /* backward call indicators */
[- - C]  SiCgPtyNum cgPtyNum1;        /* location number */
[- A C]  SiOptBckCalInd optBckCalInd; /* optional bwd call indicators */
[- - C]  SiConnectedNum connNum;      /* connected number */
[- - C]  SiAccDelInfo accDelInfo;     /* access delivery information */
[- - C]  SiPropDly cllHstry;          /* call history information */
[- - C]  SiRedirNum redirNum;         /* redirection number */
[- - C]  SiRedirRestr redirRstr;      /* redirection restriction */
[- A -]  SiBusinessGrp businessGrp;   /* business group */
[- A -]  SiInfoInd infoInd;           /* information indicators */
[- A -]  SiCarrierId carrierId;       /* carrier identification */
[- A -]  SiCarrierSelInf carSelInf;   /* carrier selection info */
[- A -]  SiEgress egress;             /* egress service */
[- A -]  SiGenAddr genAddr;           /* generic address */
```

```
[- A -]  SiInfoReqInd infoReqInd;     /* info request indicators */
[- A -]  SiJurisInf jurisInf;         /* jurisdiction information */
[- A -]  SiNetTransport netTransport; /* network transport */
[- A -]  SiSpecProcReq specProcReq;   /* special processing request */
[- A -]  SiTransReq transReq;         /* transaction request */
[J - -]  SiNotifInd reasonNoNumInd;   /* reason no number indicated */
[J - -]  SiChargeNum contractorNum;   /* contractor number */
[J - -]  SiChgZoneInfo chgZoneInfo;   /* charging zone information */
[J - -]  SiNetFuncType netFuncType;   /* network function type */
[- A -]  SiGenName genName;           /* generic name */
       } SiConEvnt;
```

The following list describes the fields in the `SiConEvnt` structure, each of which corresponds to an ISUP parameter. (Note that ITU_T Recommendations are for the CCITT network variant.)

* *`natConInd`*
  The name of an `SiNatConInd` structure that contains the *nature of connection indicators* parameter described in Section 3.23 of Japan NTT Specification NTT-Q763-a, Section 3.24 of ANSI Standard T1.113, and Section 3.35 of ITU-T Recommendation Q.763. This parameter provides information about the type of connection to be used for the call.

* *`fwdCallInd`*
  The name of an `SiFwdCallInd` structure that contains the *forward call indicators* parameter described in Section 3.20 of Japan NTT Specification NTT-Q763-a, Section 3.20 of ANSI Standard T1.113, and Section 3.23 of ITU-T Recommendation Q.763. This parameter provides information about how to route the call.

* *`cgPtyCat`*
  The name of an `SiCgPtyCat` structure that contains the *calling party's category* parameter described in Section 3.9 of Japan NTT Specification NTT-Q763-a, Section 3.8 of ANSI Standard T1.113, and Section 3.11 of ITU-T Recommendation Q.763. This parameter provides information indicating the category of the calling party and, in case of semiautomatic calls, the service language to be spoken by the incoming, delay and assistance operators (for example, English, German, or French).

* *`txMedReq`*
  The name of an `SiTxMedReq` structure that contains the *transmission medium requirement* parameter described in Section 3.35 of Japan NTT Specification NTT-Q763-a and Section 3.54 of ITU-T Recommendation Q.763. This parameter provides information indicating the type of transmission medium required for the connection (for example, speech, 64 Kbit/s unrestricted, 3.1 kHz audio, or 64 Kbit/s preferred).

* *`usrServInfoA`*
  The name of an `SiusrServInfoA` structure that contains the *user service information* parameter described in Section 3.33 of ANSI Standard T1.113.

* *cdPtyNum*
  The name of an `SiCdPtyNum` structure that contains the *called party number* parameter described in Section 3.7 of Japan NTT Specification NTT-Q763-a, Section 3.6 of ANSI Standard T1.113, and Section 3.9 of ITU-T Recommendation Q.763. This parameter provides information that identifies the called party.

* *tranNetSel*
  The name of an `SiTranNetSel` structure that contains the *transit network selection* parameter described in Section 3.31C of ANSI Standard T1.113 and Section 3.53 of ITU-T Recommendation Q.763. This parameter provides information indicating the transit network(s) requested to be used in the call.

* *callRefA*
  The name of the `SiCallRefA` structure that contains the *call reference* parameter described in Section 3.5 of ANSI Standard T1.113.

* *callRef*
  The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.6 of Japan NTT Specification NTT-Q763-a and Section 3.8 of ITU-T Recommendation Q.763. This parameter provides circuit-independent information for identifying a particular call.

* *cgPtyNum*
  The name of an `SiCgPtyNum` structure that contains the *calling party number* parameter described in Section 3.8 of Japan NTT Specification NTT-Q763-a, Section 3.7 of ANSI Standard T1.113, and Section 3.10 of ITU-T Recommendation Q.763. This parameter provides information that identifies the calling party.

* *opFwdCalInd*
  The name of an `SiOpFwdCalInd` structure that contains the *optional forward call indicators* parameter described in Section 3.25 of Japan NTT Specification NTT-Q763-a and Section 3.38 of ITU-T Recommendation Q.763.

* *redirNum*
  The name of an `SiRedirNum` structure that contains the *redirecting number* parameter described in Section 3.28 of Japan NTT Specification NTT-Q763-a, Section 3.27A of ANSI Standard T1.113, and Section 3.44 of ITU-T Recommendation Q.763.

* *redirInfo*
  The name of an `SiRedirInfo` structure that contains the *redirection information* parameter described in Section 3.29 of Japan NTT Specification NTT-Q763-a, Section 3.29 of ANSI Standard T1.113, and Section 3.45 of ITU-T Recommendation Q.763.

* *cugIntCode*
  The name of an `SiCugIntCode` structure that contains the *closed user group interlock code* parameter described in Section 3.13 of Japan NTT Specification NTT-Q763-a and Section 3.15 of ITU-T Recommendation Q.763.

* `connReqA`
  The name of an `SiConnReqA` structure that contains the *connection request* parameter described in Section 3.15 of ANSI Standard T1.113.

* `connReq`
  The name of an `SiConnReq` structure that contains the *connection request* parameter described in Section 3.17 of ITU-T Recommendation Q.763.

* `origCdNum`
  The name of an `SiOrigCdNum` structure that contains the *original called number* parameter described in Section 3.26 of Japan NTT Specification NTT-Q763-a, Section 3.26 of ANSI Standard T1.113, and Section 3.39 of ITU-T Recommendation Q.763.

* `usr2UsrInfo`
  The name of an `SiUsr2UsrInfo` structure that contains the *user-to-user information* parameter described in Section 3.38 of Japan NTT Specification NTT-Q763-a and Section 3.61 of ITU-T Recommendation Q.763.

* `accTrnspt`
  The name of an `SiAccTrnspt` structure that contains the *access transport* parameter described in Section 3.2 of Japan NTT Specification NTT-Q763-a, Section 3.1A of ANSI Standard T1.113, and Section 3.3 of ITU-T Recommendation Q.763.

* `chargeNum`
  The name of an `SiChargeNum` structure that contains the *charge number* parameter described in Section3.10 of ANSI Standard T1.113.

* `origLineInf`
  The name of the `SiOrigLineInf` structure that contains the *originating line information* parameter described in Section 3.26A of ANSI Standard T1.113.

* `usrServInfo`
  The name of an `SiUsrServInfo` structure that contains the *user service information* parameter described in Section 3.36 of Japan NTT Specification NTT-Q763-a and Section 3.57 of ITU-T Recommendation Q.763.

* `usr2UsrInd`
  The name of an `SiUsr2UsrInd` structure that contains the *user-to-user indicators* parameter described in Section 3.37 of Japan NTT Specification NTT-Q763-a and Section 3.60 of ITU-T Recommendation Q.763.

* `propDly`
  The name of an `SiPropDly` structure that contains the *propagation delay counter* parameter described in Section 3.42 of ITU-T Recommendation Q.763.

* *usrServInfo1*
  The name of an `SiUsrServInfo` structure that contains the *user service information prime* parameter described in Section 3.58 of ITU-T Recommendation Q.763.

* *netFac*
  The name of an `SiNetSpecFacil` structure that contains the *network specific facilities* parameter described in Section 3.36 of ITU-T Recommendation Q.763.

* *orgPteCde*
  The name of an `SiSigPointCode` structure that contains the *origination ISC point code* parameter described in Section 3.40 of ITU-T Recommendation Q.763.

* *genDigits*
  The name of an `SiGenDigits` structure that contains the *generic digits* parameter described in Section 3.20B of ANSI Standard T1.113 and Section 3.24 of ITU-T Recommendation Q.763.

* *genNmb*
  The name of an `SiGenNum` structure that contains the *generic number* parameter described in Section 3.56 of Japan NTT Specification NTT-Q763-a and Section 3.26 of ITU-T Recommendation Q.763.

* *usrTSrvInfo*
  The name of an `SiUsrTSrvInfo` structure that contains the *user teleservice information* parameter described in Section 3.59 of ITU-T Recommendation Q.763.

* *remotOper*
  The name of an `SiRemotOper` structure that contains the *remote operations* parameter described in Section 3.48 of ITU-T Recommendation Q.763.

* *parmCom*
  The name of an `SiParmCompInfo` structure that contains the *parameter compatibility information* parameter described in Section 3.41 of ITU-T Recommendation Q.763.

* *notifInd*
  The name of an `SiNotifInd` structure that contains the *generic notification* parameter described in Section 3.25 of ITU-T Recommendation Q.763.

* *servCode*
  The name of an `SiServiceCode` structure that contains the *service code* parameter described in Section 3.29D of ANSI Standard T1.113.

* *serviceAct1*
  The name of the `SiServiceAct` structure that contains the *service activation* parameter described in Section 3.29C of ANSI Standard T1.113.

* *serviceAct*
  The name of an `SiServiceAct` structure that contains the *service activation* parameter described in Section 3.49 of ITU-T Recommendation Q.763.

* *mlppPrec*
  The name of an `SiMlppPrec` structure that contains the *MLPP precedence* parameter described in Section 3.34 of ITU-T Recommendation Q.763.

* *txMedUsPr*
  The name of an `SiTxMedReq` structure that contains the *transmission medium used* parameter described in Section 3.56 of ITU-T Recommendation Q.763.

* *bckCallInd*
  The name of an `SiBckCalInd` structure that contains the *backward call indicators* parameter described in Section 3.4 of Japan NTT Specification NTT-Q763-a, Section 3.3 of ANSI Standard T1.113, and Section 3.5 of ITU-T Recommendation Q.763.

* *cgPtyNum1*
  The name of an `SiCgPtyNum` structure that contains the *location number* parameter described in Section 3.30 of ITU-T Recommendation Q.763.

* *optBckCalInd*
  The name of an `SiOptBckCalInd` structure that contains the *optional backward call indicators* parameter described in Section 3.24 of Japan NTT Specification NTT-Q763-a, Section 3.24C of ANSI Standard T1.113, and Section 3.37 of ITU-T Recommendation Q.763.

* *connNum*
  The name of an `SiConnectedNum` structure that contains the *connected number* parameter described in Section 3.16 of ITU-T Recommendation Q.763.

* *accDelInfo*
  The name of an `SiAccDelInfo` structure that contains the *access delivery information* parameter described in Section 3.2 of ITU-T Recommendation Q.763.

* *cllHstry*
  The name of an `SiPropDly` structure that contains the *call history information* parameter described in Section 3.7 of ITU-T Recommendation Q.763.

* *redirNum*
  The name of an `SiRedirNum` structure that contains the *redirection number* parameter described in Section 3.46 of ITU-T Recommendation Q.763.

* *redirRstr*
  The name of an `SiRedirRestr` structure that contains the *redirection number restriction* parameter described in Section 3.47 of ITU-T Recommendation Q.763.

- `businessGrp`
  The name of an `SiBusinessGrp` structure that contains the *business group* parameter described in Section 3.3A of ANSI Standard T1.113.

- `infoInd`
  The name of an `SiInfoInd` structure that contains the *information indicators* parameter described in Section 3.22 of ANSI Standard T1.113.

- `carrierId`
  The name of an `SiCarrierId` structure that contains the *carrier identification* parameter described in Section 3.8A of ANSI Standard T1.113.

- `carSelInf`
  The name of the `SiCarrierSelInf` structure that contains the *carrier selection information* parameter described in Section 3.8B of ANSI Standard T1.113.

- `egress`
  The name of an `SiEgress` structure that contains the *egress service* parameter described in section Section 3.16A of ANSI Standard T1.113.

- `genAddr`
  The name of the `SiGenAddr` structure that contains the *generic address* parameter described in Section 3.20A of ANSI Standard T1.113.

- `infoReqInd`
  The name of the `SiInfoReqInd` structure that contains the *information request indicators* parameter described in Section 3.23 of ANSI Standard T1.113.

- `jurisInf`
  The name of the `SiJurisInf` structure that contains the *jurisdiction information* parameter described in Section 3.23A of ANSI Standard T1.113.

- `netTransport`
  The name of the `SiNetTransport` structure that contains the *network transport* parameter described in Section 3.24A of ANSI Standard T1.113.

- `specProcReq`
  The name of the `SiSpecProcReq` structure that contains the *special processing request* parameter described in Section 3.30A of ANSI Standard T1.113.

- `transReq`
  The name of the `SiTransReq` structure that contains the *transaction request* parameter described in Section 3.31B of ANSI Standard T1.113.

* *reasonNoNumInd*
  The name of the `SiNotifInd` structure that contains the *reason no number indicated* parameter described in Section 3.57 of Japan NTT Specification NTT-Q763-a. The value for this parameter can be any of the following:

`0x01`	User request
`0x02`	Service competition
`0x03`	Call is from a public telephone

* *contractorNum*
  The name of the `SiChargeNum` structure that contains the *contractor number* described in Section 3.55 of Japan NTT Specification NTT-Q763-a.

* *chgZoneInfo*
  The name of the `SiChgZoneInfo` structure that contains the *charging zone (toll area) information* described Section 3.51 of Japan NTT Specification NTT-Q763-a.

* *netFuncType*
  The name of the `SiNetFuncType` structure that contains the *network function type* described in Section 3.50 of Japan NTT Specification NTT-Q763-a.

* *genName*
  The name of the `SiGenName` structure that contains the *generic name* (type of information provided) described in Section 3.20C of ANSI Standard T1.113. Note that this parameter is allowed only if the environment variable `ISUP_GENERIC_NAME` is defined on the SINAP node. This variable defines the Generic Name parameter with the code `0xc7`.

Connection Status Event Structure (`SiCnStEvnt`)

The `SiCnStEvnt` structure contains information for an ISUP connection-status event. The following table presents the ISUP primitives used for these events. The column labeled "Primitive Type" lists the primitive name as it is defined in the `isup_msg_t.hdr` structure's `prim_ev` field, and the column labeled "Definition" indicates the type of action invoked by the primitive. The column labeled "Direction" indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	Direction
`EVTSITCNSTREQ`	Connection Status Request	Output
`EVTSITCNSTIND`	Connection Status Indication	Input

The following table lists ISUP message types associated with the primitives in the preceding table, followed by the ITU-T Recommendation Q.763 table, the ANSI Standard T1.113.3 table (if applicable), and the Japan NTT Specification NTT-Q763-a table (if applicable) in which the message is described.

ISUP Message Type	Table Reference ITU-T	ANSI	NTT
ACM	21	5	5
CPG	23	6A	7
SAM	35		
INR	31	13	
INF	30	12	
FOT	37	17	
NRM	46		
IDR	47		
IRS	48		
SGM	49		

In the 1988 edition of ITU-T Recommendation Q.763, the ISUP message types CMR (Call Modification Request), CMC (Call Modification Complete), and CMRJ (Call Modification Reject) are also associated with the preceding connection-status event primitives. However, these message types have not yet been carried forward to the 1993 edition of ITU-T Recommendation Q.763.

The `SiCnStEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **J**, **C** and/or **A** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT (ITU), or ANSI variant.

```
       typedef struct siCnStEvnt   /* Connection Status Event */
       {
[- - C]  SiSubNum subNum;              /* subsequent number */
[J A C]  SiBckCalInd bckCallInd;       /* backward call indicators */
[- A -]  SiChargeNum chargeNum;        /* charge number */
[- A -]  SiCallRefA callRefA;          /* call reference */
[- A -]  SiConnReqA connReqA;          /* connection request */
[J A C]  SiOptBckCalInd optBckCalInd;  /* optional bwd call indicators */
[J A C]  SiCauseDgn causeDgn;          /* cause indicators */
[- - C]  SiConnectedNum connNum;       /* connected number */
[J - C]  SiUsr2UsrInd usr2UsrInd;      /* user to user indicators */
[J - C]  SiUsr2UsrInfo usr2UsrInfo;    /* user to user information */
[J A C]  SiAccTrnspt accTrnspt;        /* access transport */
[- - C]  SiCalModInd calModInd;        /* call modification indicators */
[J A C]  SiEvntInfo evntInfo;          /* event information */
[- - C]  SiRedirNum redirNum;          /* redirection number */
[- - C]  SiInfoInd infoInd;            /* information indicators */
[- A C]  SiInfoReqInd infoReqInd;      /* info request indicators */
[- A C]  SiCgPtyCat cgPtyCat;          /* calling party category */
[- A C]  SiCgPtyNum cgPtyNum;          /* calling party number */
[- - C]  SiConnReq connReq;            /* connection request */
[J - C]  SiCallRef callRef;            /* call reference */
[- - C]  SiNotifInd notifInd;          /* notification indicator */
[- - C]  SiTxMedReq txMedUsed;         /* transmission medium used */
[- - C]  SiEchoCtl echoControl;        /* echo control */
[- - C]  SiAccDelInfo accDelInfo;      /* access delivery information */
[- - C]  SiGenNum genNmb;              /* generic number */
[- - C]  SiParmCompInfo parmCom;       /* parameter compatibility info */
[- - C]  SiCllDiverInfo cllDivr;       /* call Diversion information */
[- - C]  SiNetSpecFacil netFac;        /* network specific facility */
[- - C]  SiRemotOper remotOper;        /* remote operations */
[- - C]  SiServiceAct serviceAct;      /* service activation */
[- - C]  SiRedirRestr redirRstr;       /* redirection restriction */
[- - C]  SiMcidReqInd mcidReq;         /* MCID request indicators */
[- - C]  SiMcidRspInd mcidRsp;         /* MCID response indicators */
[- - C]  SiMsgCompInfo msgCom;         /* message compatibility info */
[- A -]  SiOrigLineInf origLineInf;    /* originating line information */
[- A -]  SiBusinessGrp businessGrp;    /* business group */
[- A -]  SiInfoInd infoInd2;           /* information indicators */
[- A -]  SiNetTransport netTransport;  /* network transport */
[- A -]  SiNotifInd notifInd1;         /* notification indicator */
[- A -]  SiRedirInfo redirInf;         /* redirection information */
       } SiCnStEvnt;
```

The following list describes the fields in the `SiCnStEvnt` structure. Note that all of the structures referenced in these fields are also defined in `issl_sit.h`.

* *subNum*
  The name of an `SiSubNum` structure that contains the *subsequent number* parameter described in Section 3.51 of ITU-T Recommendation Q.763.

* *bckCallInd*
  The name of an `SiBckCalInd` structure that contains the *backward call indicators* parameter described in Section 3.4 of Japan NTT Specification NTT-Q763-a, Section 3.3 of ANSI Standard T1.113, and Section 3.5 of ITU-T Recommendation Q.763.

* *chargeNum*
  The name of an `SiChargeNum` structure that contains the *charge number* parameter described in Section 3.10 of ANSI Standard T1.113.

* *callRefA*
  The name of an `SiCallRefA` structure that contains the *call reference* parameter described in Section 3.5 of ANSI Standard T1.113.

* *connReqA*
  The name of an `SiConnReqA` structure that contains the *connection request* parameter described in Section 3.15 of ANSI Standard T1.113.

* *optBckCalInd*
  The name of an `SiOptBckCalInd` structure that contains the *optional backward call indicators* parameter described in Section3.24 of Japan NTT Specification NTT-Q763-a, Section 3.24C of ANSI Standard T1.113, and Section 3.37 of ITU-T Recommendation Q.763.

* *causeDgn*
  The name of an `SiCauseDgn` structure that contains the *cause indicators* parameter described in Section 3.10 of Japan NTT Specification NTT-Q763-a, Section 3.9 of ANSI Standard T1.113, and Section 3.12 of ITU-T Recommendation Q.763.

* *connNum*
  The name of an `SiConnectedNum` structure that contains the *connected number* parameter described in Section 3.16 of ITU-T Recommendation Q.763.

* *usr2UsrInd*
  The name of an `SiUsr2UsrInd` structure that contains the *user-to-user indicators* parameter described in Section 3.37 of Japan NTT Specification NTT-Q763-a and Section 3.60 of ITU-T Recommendation Q.763.

* *usr2UsrInfo*
  The name of an `SiUsr2UsrInfo` structure that contains the *user-to-user information* parameter described in Section 3.38 of Japan NTT Specification NTT-Q763-a and Section 3.61 of ITU-T Recommendation Q.763.

* *accTrnspt*
  The name of an `SiAccTrnspt` structure that contains the *access transport* parameter described in Section 3.2 of Japan NTT Specification NTT-Q763-a, Section 3.1A of ANSI Standard T1.113, and Section 3.3 of ITU-T Recommendation Q.763.

* *calModInd*
  The name of an `SiCalModInd` structure that contains the *call modification indicators* parameter described in Section 3.5 of the 1988 edition of ITU-T Recommendation Q.763. This parameter is maintained for backward compatibility.

* *evntInfo*
  The name of an `SiEvntInfo` structure that contains the *event information* parameter described in Section 3.18 of Japan NTT Specification NTT-Q763-a, Section 3.17A of ANSI Standard T1.113, and Section 3.21 of ITU-T Recommendation Q.763.

* *redirNum*
  The name of an `SiRedirNum` structure that contains the *redirection number* parameter described in Section 3.46 of ITU-T Recommendation Q.763.

* *infoInd*
  The name of an `SiInfoInd` structure that contains the *information indicators* parameter described in Section 3.28 of ITU-T Recommendation Q.763.

* *infoReqInd*
  The name of an `SiInfoReqInd` structure that contains the *information request indicators* parameter described in Section 3.29 of ITU-T Recommendation Q.763.

* *cgPtyCat*
  The name of an `SiCgPtyCat` structure that contains the *calling party's category* parameter described in Section 3.11 of ITU-T Recommendation Q.763.

* *cgPtyNum*
  The name of an `SiCgPtyNum` structure that contains the *calling party number* parameter described in Section 3.10 of ITU-T Recommendation Q.763.

* *connReq*
  The name of an `SiConnReq` structure that contains the *connection request* parameter described in Section 3.17 of ITU-T Recommendation Q.763.

* *callRef*
  The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.8 of ITU-T Recommendation Q.763.

* *notifInd*
  The name of an `SiNotifInd` structure that contains the *generic notification* parameter described in Section 3.6 of Japan NTT Specification NTT-Q763-a and Section 3.25 of ITU-T Recommendation Q.763.

* *txMedUsed*
  The name of an `SiTxMedReq` structure that contains the *transmission medium used* parameter described in Section 3.56 of ITU-T Recommendation Q.763.

* *echoControl*
  The name of an `SiEchoCtl` structure that contains the *echo control information* parameter described in Section 3.19 of ITU-T Recommendation Q.763.

* *accDelInfo*
  The name of an `SiAccDelInfo` structure that contains the *access delivery information* parameter described in Section 3.2 of ITU-T Recommendation Q.763.

* *genNmb*
  The name of an `SiGenNum` structure that contains the *generic number* parameter described in Section 3.26 of ITU-T Recommendation Q.763.

* *parmCom*
  The name of an `SiParmCompInfo` structure that contains the *parameter compatibility information* parameter described in Section 3.41 of ITU-T Recommendation Q.763.

* *cllDivr*
  The name of an `SiCllDiverInfo` structure that contains the *call diversion information* parameter described in Section 3.6 of ITU-T Recommendation Q.763.

* *netFac*
  The name of an `SiNetSpecFacil` structure that contains the *network specific facilities* parameter described in Section 3.36 of ITU-T Recommendation Q.763.

* *remotOper*
  The name of an `SiRemotOper` structure that contains the *remote operations* parameter described in Section 3.48 of ITU-T Recommendation Q.763.

* *serviceAct*
  The name of an `SiServiceAct` structure that contains the *service activation* parameter described in Section 3.49 of ITU-T Recommendation Q.763.

* *redirRstr*
  The name of an `SiRedirRestr` structure that contains the *redirection number restriction* parameter described in Section 3.47 of ITU-T Recommendation Q.763.

* *mcidReq*
  The name of an `SiMcidReqInd` structure that contains the *MCID request indicator* parameter described in Section 3.31 of ITU-T Recommendation Q.763.

* *mcidRsp*
  The name of an `SiMcidRspInd` structure that contains the *MCID response indicator* parameter described in Section 3.32 of ITU-T Recommendation Q.763.

* *msgCom*
  The name of an `SiMsgCompInfo` structure that contains the *message compatibility information* parameter described in Section 3.33 of ITU-T Recommendation Q.763.

* *origLineInf*
  The name of an `SiOrigLineInf` structure that contains the *originating line information* parameter described in Section 3.26A of ANSI Standard T1.113.

* *businessGrp*
  The name of the `SiBusinessGrp` structure that contains the *business group* parameter described in Section 3.3A of ANSI Standard T1.113.

* *infoInd2*
  The name of the `SiInfoInd` structure that contains the *information indicators* parameter described in Section 3.22 of ANSI Standard T1.113.

* *netTransport*
  The name of the `SiNetTransport` structure that contains the *network transport* parameter described in Section 3.24A of ANSI Standard T1.113.

* *notifInd1*
  The name of the `SiNotifInd` structure that contains the *notification indicator* parameter described in Section 3.24B of ANSI Standard T1.113.

* *redirInf*
  The name of the `SiRedirInfo` structure that contains the *redirection information* parameter described in Section 3.29 of ANSI Standard T1.113

User-Information Event Structure (**SiInfoEvnt**)

The `SiInfoEvnt` structure contains information for an ISUP user-information event. The following table presents the ISUP primitives and messages used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” indicates the type of action invoked by the primitive. The column labeled “ISUP Message Table Reference” indicates the

ISUP messages associated with the primitive and the tables (in parentheses) in ITU-T Recommendation Q.763 and the ANSI Standard T1.113.3 in which the message is defined. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	ISUP Message Table Reference ITU-T	ANSI	Direction
EVTSITDATREQ	User Information Request	USR (36) PAM (43)	PAM (23)	Output
EVTSITDATIND	User Information Indication	USR (36) PAM (43)	PAM (23)	Input

The `SiInfoEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **C** and/or **A** shown to the left of each element in the structure indicate whether the element applies to the CCITT or ANSI ISUP versions.

```
         typedef struct siInfoEvnt   /* User Information Event */
         {
[- - C]  SiCallRef callRef;            /* call reference */
[- A C]  SiPassAlng passAlng;          /* pass along */
[- - C]  SiUsr2UsrInfo usr2UsrInfo;    /* user to user information */
[- - C]  SiAccTrnspt accTrnspt;        /* access transport */
         } SiInfoEvnt;
```

The following list describes the fields in the `SiInfoEvnt` structure.

* *callRef*
The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.8 of ITU-T Recommendation Q.763.

* *passAlng*
The name of an `SiPassAlng` structure that contains the ISUP protocol message to be embedded in a Pass-Along (PAM) message as described in Table 43 of ITU-T Recommendation Q.763 and in Table 23 of ANSI Standard T1.113.3.

* *usr2UsrInfo*
The name of an `SiUsr2UsrInfo` structure that contains the *user-to-user information* parameter described in Section 3.61 of ITU-T Recommendation Q.763.

* *accTrnspt*
The name of an `SiAccTrnspt` structure that contains the *access transport* parameter described in Section 3.3 of ITU-T Recommendation Q.763.

Call-Facility Event Structure (SiFacEvnt)

The SiFacEvnt structure contains information for an ISUP call-facility event. The following table presents the ISUP primitives and messages used for these events. The column labeled "Primitive Type" lists the primitive name as it is defined in the isup_msg_t.hdr structure's prim_ev field, and the column labeled "Definition" indicates the type of action invoked by the primitive. The column labeled "ISUP Message Table Reference" indicates the ISUP messages associated with the primitive and the table in ITU-T Recommendation Q.763 in which the message is defined. The column labeled "Direction" indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the prim_ev Field)	Definition	ISUP Message (Q.763 Table Reference)	Direction
EVTSITFACREQ	Call Facility Request	FAC (45) FAR (42)	Output
EVTSITFACIND	Call Facility Indication	FAC (45) FAR (42)	Input
EVTSITFACRSP	Call Facility Response	FAA (42) FRJ (29)	Output
EVTSITFACCFM	Call Facility Confirmation	FAA (42) FRJ (29)	Input

The SiFacEvnt structure is defined in the issl_sit.h include file, and its format is shown here. The letter **C** shown to the left of each element in the structure indicates that the element applies only to the CCITT variant.

```
typedef struct siFacEvnt  /* Call Facility Event */
    {
[- - C] SiFacInd facInd;/* facility indicator */
[- - C] SiUsr2UsrInd usr2UsrInd;/* user to user indicator */
[- - C] SiCallRef callRef; /* call reference */
[- - C] SiCauseDgn causeDgn;/* cause indicator */
[- - C]  SiMsgCompInfo msgCom;/* message compatibility info */
[- - C]  SiParmCompInfo parmCom; /* parameter compatibility info */
[- - C]  SiRemotOper remotOper; /* remote operations */
[- - C] SiServiceAct serviceAct;/* service activation */
[- - C]  SiConnReq connReq;/* connection request */
  }SiFacEvnt;
```

The following list describes the fields in the SiFacEvnt structure.

* *facInd*
  The name of an SiFacInd structure that contains the *facility indicator* parameter described in Section 3.22 of ITU-T Recommendation Q.763.

* *usr2UsrInd*
  The name of an `SiUsr2UsrInd` structure that contains the *user-to-user indicators* parameter described in Section 3.60 of ITU-T Recommendation Q.763.

* *callRef*
  The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.8 of ITU-T Recommendation Q.763.

* *causeDgn*
  The name of an `SiCauseDgn` structure that contains the *cause indicators* parameter described in Section 3.12 of ITU-T Recommendation Q.763.

* *msgCom*
  The name of an `SiMsgCompInfo` structure that contains the *message compatibility information* parameter described in Section 3.33 of ITU-T Recommendation Q.763.

* *parmCom*
  The name of an `SiParmCompInfo` structure that contains the *parameter compatibility information* parameter described in Section 3.41 of ITU-T Recommendation Q.763.

* *remotOper*
  The name of an `SiRemotOper` structure that contains the *remote operations* parameter described in Section 3.48 of ITU-T Recommendation Q.763.

* *serviceAct*
  The name of an `SiServiceAct` structure that contains the *service activation* parameter described in Section 3.49 of ITU-T Recommendation Q.763.

* *connReq*
  The name of the `SiConnReq` structure that contains the *connection request* parameter described in Section 3.17 of ITU-T Recommendation Q.763.

Call-Suspend Event Structure (`SiSuspEvnt`)

The `SiSuspEvnt` structure contains information for an ISUP call-suspend event. The following table presents the ISUP primitives and messages used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” indicates the type of action invoked by the primitive. The column labeled “ISUP Message Table Reference” indicates the ISUP messages associated with the primitive and the tables (in parentheses) in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and Japan NTT Specification NTT-Q763-a in which the message is defined. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	ISUP Message Table Reference ITU-T	ANSI	NTT	Direction
EVTSITSUSPREQ	Call Suspend Request	SUS (38)	SUS (17)A	SUS (13)	Output
EVTSITSUSPIND	Call Suspend Indication	SUS (38)	SUS (17)A	SUS (13)	Input

The `SiSuspEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **J**, **C**, and/or **A** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT, or ANSI ISUP version.

```
              typedef struct siSuspEvnt   /* Call Suspend Event */
              {
       [J A C]  SiSusResInd susResInd;       /* Suspend/Resume indicators */
       [- A -]  SiCallRefA callRefA;         /* call reference */
       [- - C]  SiCallRef callRef;           /* call reference */
              } SiSuspEvnt;
```

The following list describes the fields in the `SiSuspEvnt` structure.

* *`susResInd`*
  The name of an `SiSusResInd` structure that contains the *suspend/resume indicators* parameter described in Section 3.33 of the Japan NTT Specification NTT-Q763-a, Section 3.31A of ANSI Standard T1.113.3, and Section 3.52 of ITU-T Recommendation Q.763.

* *`callRefA`*
  The name of an `SiCallRefA` structure that contains the *call reference* parameter described in Section 3.5 of ANSI Standard T1.113.3.

* *`callRef`*
  The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.8 of ITU-T Recommendation Q.763.

Call-Resume Event Structure (`SiResmEvnt`)

The `SiResmEvnt` structure contains information for an ISUP call-resume event. The following table presents the ISUP primitives and messages used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” presents the type of action invoked by the primitive. The column labeled “ISUP Message Table Reference” indicates the ISUP messages associated with the primitive and the tables (in parentheses) in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and in Japan NTT Specification NTT-Q763-a in which the message is defined. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	ISUP Message Table Reference ITU-T	ANSI	NTT	Direction
EVTSITRESMREQ	Call Resume Request	RES (38)	RES (17A)	RES (13)	Output
EVTSITRESMIND	Call Resume Indication	RES (38)	RES (17A)	RES (13)	Input

The `SiResmEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **J**, **C**, and/or **A** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT, or ANSI ISUP version.

```
              typedef struct siResmEvnt   /* Call Resume Event */
              {
      [J A C]  SiSusResInd susResInd;       /* Suspend/Resume indicators */
      [- A -]  SiCallRefA callRefA;         /* call reference */
      [- - C]  SiCallRef callRef;           /* call reference */
              } SiResmEvnt;
```

The following list describes the fields in the `SiResmEvnt` structure.

* *`susResInd`*
  The name of an `SiSusResInd` structure that contains the *suspend/resume indicators* parameter described in Section 3.33 of Japan NTT Specification NTT-Q763-a, Section 3.31A of ANSI Standard T1.113.3, and Section 3.52 of ITU-T Recommendation Q.763.

* *`callRefA`*
  The name of an `SiCallRefA` structure that contains the *call reference* parameter described in Section 3.5 of ANSI Standard T1.113.3.

* *`callRef`*
  The name of an `SiCallRef` structure that contains the *call reference* parameter described in Section 3.8 of ITU-T Recommendation Q.763.

Connection-Release Event Structure (`SiRelEvnt`)

The `SiRelEvnt` structure contains information for an ISUP connection-release event. The following table presents the ISUP primitives and messages used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” presents the type of action invoked by the primitive. The column labeled “ISUP Message” indicates the ISUP messages associated with the primitive and the tables (in parentheses) in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and the Japan NTT Specification NTT-Q763-a in which the message is defined. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	ISUP Message Table Reference)			Direction
		ITU-T	ANSI	NTT	
`EVTSITRELREQ`	Connection Release Request	REL (33)	REL (14A)	REL (11)	Output
`EVTSITRELIND`	Connection Release Indication	REL (33)	REL (14A)	REL (11)	Input
`EVTSITRELRSP`	Connection Release Response	RLC (34)	RLC (18)	RLC (12)	Output
`EVTSITRELCFM`	Connection Release Confirmation	RLC (34)	RLC (18)	RLC (12)	Input

The `SiRelEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **J**, **C,** and/or **A** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT (ITU-T) or the ANSI ISUP version.

```
            typedef struct siRelEvnt    /* Connection Release Event */
            {
    [J A C]  SiCauseDgn causeDgn;          /* cause indicators */
    [- A C]  SiRedirInfo redirInfo;        /* redirection information */
    [- - C]  SiRedirNum redirNum;          /* redirection number */
    [- A -]  SiCallRefA callRefA;          /* call reference */
    [- - C]  SiSigPointCode sigPointCode;  /* signaling point code */
    [- A C]  SiAccTrnspt accTrnspt;        /* access transport */
    [J - C]  SiUsr2UsrInfo usr2UsrInfo;    /* user to user information */
    [- A C]  SiAutoCongLvl autoCongLvl;    /* automatic congestion level */
    [- - C]  SiAccDelInfo accDelInfo;      /* access delivery information */
    [- - C]  SiParmCompInfo parmCom;       /* parameter compatibility info */
    [- - C]  SiNetSpecFacil netFac;        /* network specific facility */
    [- - C]  SiRedirRestr redirRstr;       /* redirection restriction */
    [- - C]  SiUsr2UsrInd usr2UsrInd;      /* user to user indicators */
    [- A -]  SiChargeNum chargeNum;        /* charge number */
    [- A -]  SiGenAddr genAddr;            /* generic address */
            } SiRelEvnt;
```

The following list describes the fields in the `SiRelEvnt` structure.

* *causeDgn*
The name of an `SiCauseDgn` structure that contains the *cause indicators* parameter described in Section 3.10 of Japan NTT Specification NTT-Q763-a, Section 3.9 of ANSI Standard T1.113, and Section 3.12 of ITU-T Recommendation Q.763.

* *redirInfo*
The name of an `SiRedirInfo` structure that contains the *redirection information* parameter described in Section 3.20 of ANSI Standard T1.113 and Section 3.45 of ITU-T Recommendation Q.763.

* *redirNum*
The name of an `SiRedirNum` structure that contains the *redirection number* parameter described in Section 3.46 of ITU-T Recommendation Q.763.

* *callRefA*
The name of an `SiCallRefA` structure that contains the *call reference* parameter described in Section 3.5 of ANSI Standard T1.113.3.

* *sigPointCode*
The name of an `SiSigPointCode` structure that contains the *signalling point code* parameter described in Section 3.50 of ITU-T Recommendation Q.763.

* *`accTrnspt`*
  The name of an `SiAccTrnspt` structure that contains the *access transport* parameter described in section 3.1A of ANSI Standard T1.113 and Section 3.3 of ITU-T Recommendation Q.763.

* *`usr2UsrInfo`*
  The name of an `SiUsr2UsrInfo` structure that contains the *user-to-user information* parameter described in Section 3.38 of Japan NTT Specification NTT-Q763-a and Section 3.61 of ITU-T Recommendation Q.763.

* *`autoCongLvl`*
  The name of an `SiAutoCongLvl` structure that contains the *automatic congestion level* parameter described in Section 3.2A of ANSI Standard T1.113 and Section 3.4 of ITU-T Recommendation Q.763.

* *`accDelInfo`*
  The name of an `SiAccDelInfo` structure that contains the *access delivery information* parameter described in Section 3.2 of ITU-T Recommendation Q.763.

* *`parmCom`*
  The name of an `SiParmCompInfo` structure that contains the *parameter compatibility information* parameter described in Section 3.41 of ITU-T Recommendation Q.763.

* *`netFac`*
  The name of an `SiNetSpecFacil` structure that contains the *network specific facilities* parameter described in Section 3.36 of ITU-T Recommendation Q.763.

* *`redirRstr`*
  The name of an `SiRedirRestr` structure that contains the *redirection number restriction* parameter described in Section 3.47 of ITU-T Recommendation Q.763.

* *`usr2UsrInd`*
  The name of an `SiUsr2UsrInd` structure that contains the *user-to-user indicators* parameter described in Section 3.60 of ITU-T Recommendation Q.763.

* *`chargeNum`*
  The name of an `SiChargeNum` structure that contains the *charge number* parameter described in Section 3.10 of ANSI Standard T1.113.3.

* *`genAddr`*
  The name of an `SiGenAddr` structure that contains the *generic address* parameter described in Section 3.20A of ANSI Standard T1.113.3.

Circuit-Status Event Structure (`SiStaEvnt`)

The `SiStaEvnt` structure contains information for an ISUP circuit-status event. The following table presents the ISUP primitives used for these events. The column labeled “Primitive Type” lists the primitive name as it is defined in the `isup_msg_t.hdr` structure’s `prim_ev` field, and the column labeled “Definition” presents the type of action invoked by the primitive. The column labeled “Direction” indicates whether the primitive is being sent by the local application (output) or the remote application (input).

Primitive Type (Specified in the `prim_ev` Field)	Definition	Direction
`EVTSITSTAREQ`	Circuit Status Request	Output
`EVTSITSTAIND`	Circuit Status Indication	Input

The following table lists ISUP services message types associated with the primitives in the preceding table. The column labeled “ISUP Message Table Reference” indicates the tables in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and Japan NTT Specification NTT-Q763-a in which the message is defined.

Message Type	ISUP Messages Table Reference		
	ITU-T	ANSI	NTT
BLO	39	18	14
BLA	39	18	14
UBL	39	18	14
UBA	39	18	14
RSC	39	18	14
CCR	39	18	
COT	28	9	
CRM		6C	
CRA		18	
CVT		18	
LPA	39	18	
CGB	40	20	
CGBA	40	20	

Message Type	ISUP Messages Table Reference ITU-T	ANSI	NTT
CGU	40	20	
CGUA	40	20	
GRS	41	21	15
GRA	25	21	9
CQM	41	21	15
CQR	24	6B	8
CFN	26	8A	
UCIC	39	18	
CHG			16

The `SiStaEvnt` structure is defined in the `issl_sit.h` include file, and its format is shown here. The letters **J**, **C**, and/or **A** shown to the left of each element in the structure indicate whether the element applies to the Japan NTT, CCITT (ITU-T), or the ANSI ISUP version.

```
       typedef struct siStaEvnt    /* Circuit Status Event */
       {
[J A C]  SiRangStat rangStat;       /* range and status */
[- A C]  SiCirGrpSupMTypInd cgsmti; /* circ grp supvsn msg type ind */
[- A -]  SiCirStateInd cirSteIndA;  /* circuit state indicators */
[- A -] SiCirValRspInd valRspInd;   /* circuit validation response indicator */
[- A -] SiCirGrpCharInd grpCharInd; /* circuit group characteristic indicators */
[- A -] SiCLLI clli;                /* common language location identification */
[- A -] SiCirIdName cirIdName       /* circuit id name */
[J - C] SiCirStateInd cirStateInd;  /* circuit state indicators */
[- A C] SiContInd contInd;          /* continuity indicator */
[- A C] SiCauseDgn causeDgn;        /* cause indicators */
[- - C] SiParmCompInfo parmCom;     /* parameter compatibility info */
[- A -] SiNatConInd natConInd;      /* Nature of conn indicators */
[J - -] SiChgInfoType chgInfoType;  /* charge information type */
[J - -] SiChargeInfo chargeInfo;    /* charge information */
       } SiStaEvnt;
```

The following list describes the fields in the `SiStaEvnt` structure.

* *rangStat*
The name of a `SiRangStat` structure that contains the *range and status* parameter described in Section 3.27 of Japan NTT Specification NTT-Q763-a, Section 3.27 of ANSI Standard T1.113, and Section 3.43 of ITU-T Recommendation Q.763.

* *cgsmti*
The name of a `SiCirGrpSupMTypInd` structure that contains the *circuit group supervision message type indicator* parameter described in Section 3.11 of ANSI Standard T1.113 and Section 3.13 of ITU-T Recommendation Q.763.

* *cirSteIndA*
The name of a `SiCirStateInd` structure that contains the *circuit state indicator* parameter described in Section 3.11B of ANSI Standard T1.113.3.

* *valRspInd*
The name of the `SiCirValRspInd` structure that contains the *circuit validation response* indicator parameter described in Section 3.11C of ANSI Standard T1.113.3.

* *grpCharInd*
The name of the `SiCirGrpCharInd` structure that contains the *circuit group characteristic indicator* parameter described in Section 3.10A of ANSI Standard T1.113.3.

* *clli*
The common language location identification code of the node sending the circuit validation test (CVT) message. See Section 3.13A of ANSI Standard T1.113.3.

* *cirIDName*
The name of the `SiCirIdName` structure that contains the *circuit ID name* parameter described in Section 3.11A of ANSI Standard T1.113.

* *cirStateInd*
The name of a `SiCirStateInd` structure that contains the *circuit state indicator* parameter described in Section 3.12 of Japan NTT Specification NTT-Q763-a and Section 3.14 of ITU-T Recommendation Q.763.

* *contInd*
The name of a `SiContInd` structure that contains the *continuity indicators* parameter described in Section 3.16 of ANSI Standard T1.113 and Section 3.18 of ITU-T Recommendation Q.763.

* *causeDgn*
The name of a `SiCauseDgn` structure that contains the *cause indicators* parameter described in Section 3.9 of ANSI Standard T1.113 and Section 3.12 of ITU-T Recommendation Q.763.

* *parmCom*
The name of a `SiParmCompInfo` structure that contains the *parameter compatibility information* parameter described in Section 3.41 of ITU-T Recommendation Q.763.

* *natConInd*
The name of a `SiNatConInd` structure that contains the *nature of connection indicators* parameter described in Section 3.24 of ANSI Standard T1.113.3.

* *chgInfoType*
The name of a `SiChgInfoType` structure that contains the *charge information type* parameter described in Section 3.54 of Japan NTT Specification NTT-Q763-a.

* `chargeInfo`
The name of a `SiChgInfo` structure that contains the *charge information* parameter described in section 3.53 of Japan NTT Specification NTT-Q763-a.

ISUP Services Primitives

The following primitives are related to ISUP services messages. These primitives are divided into two types:

- Input operations (listed in Table 5-2)
- Output operations (listed in Table 5-3)

Table 5-2. ISUP Services Input Primitives

Input Operation Primitive	Primitive Event Code	Event Structure
Connection Setup Indication	`EVTSITCONIND`	`SiConEvnt`
Connection Setup Confirmation	`EVTSITCONCFM`	`SiConEvnt`
Connection Status Indication	`EVTSITCNSTIND`	`SiCnStEvnt`
User Information Indication	`EVTSITDATIND`	`SiInfoEvnt`
Call Facility Indication	`EVTSITFACIND`	`SiFacEvnt`
Call Facility Confirmation	`EVTSITFACCFM`	`SiFacEvnt`
Call Suspend Indication	`EVTSITSUSPIND`	`SiSuspEvnt`
Call Resume Indication	`EVTSITRESMIND`	`SiResmEvnt`
Connection Release Indication	`EVTSITRELIND`	`SiRelEvnt`
Connection Release Confirmation	`EVTSITRELCFM`	`SiRelEvnt`
Circuit Status Indication	`EVTSITSTAIND`	`SiStaEvnt`
Unsolicited Status Indication	`EVTLSISTAIND`	None

Table 5-3. ISUP Services Output Primitives

Output Operation Primitive	Primitive Event Code	Event Structure
Connection Setup Request	`EVTSITCONREQ`	`SiConEvnt`
Connection Setup Response	`EVTSITCONRSP`	`SiConEvnt`
Connection Status Request	`EVTSITCNSTREQ`	`SiCnStEvnt`
User Information Request	`EVTSITDATREQ`	`SiInfoEvnt`
Call Facility Request	`EVTSITFACREQ`	`SiFacEvnt`

Table 5-3. ISUP Services Output Primitives *(Continued)*

Output Operation Primitive	Primitive Event Code	Event Structure
Call Facility Response	EVTSITFACRSP	SiFacEvnt
Call Suspend Request	EVTSITSUSPREQ	SiSuspEvnt
Call Resume Request	EVTSITRESMREQ	SiResmEvnt
Connection Release Request	EVTSITRELREQ	SiRelEvnt
Connection Release Response	EVTSITRELRSP	SiRelEvnt
Circuit Status Request	EVTSITSTAREQ	SiStaEvnt
Connection Setup Request	EVTSITCONREQ	SiConEvnt

ISUP Services Include Files

Table 5-4 describes the ISUP services-specific include files that are located in the directory `$SINAP_HOME/Include`.

Table 5-4. ISUP Services Include Files

Include File	Description
`issl.h`	Contains all include files necessary for an ISUP services application to use the ISUP Services Support Library (ISSL). Contains a definition for the ISUP services message structure `isup_msg_t`.
`issl_lsi.h`	Contains circuit group information for ISUP services applications.
`issl_gen.h`	Contains the circuit identifier used to look up the ISUP services protocol internal state table.
`issl_sit.h`	Contains ISUP services structures.

Appendix A
ISUP Services Environment Variables

Table A-1 lists and describes the ISUP services environment variables used on the SINAP/SS7 system. All environment variables, including ISUP environment variables, that can be defined on a SINAP node are contained in the SINAP/SS7 environment file `$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`. For more detailed information on the use of environment variables, see Appendix B, “SINAP/SS7 Environment Variables,” in the *SINAP/SS7 User’s Guide* (R8051)

.

Table A-1. ISUP Services Environment Variables

Environment Variable	Description

`ISUP_FEATURE`	Activates a country-specific version of the ISUP services feature on a SINAP node.
	
	You must define the environment variable separately for each node on which the ISUP services feature is to be activated.
	
	The SINAP/SS7 software currently supports these country-specific versions of ISUP services: **Note that the ISUP version specified must be compatible with the network variant defined for the node.**
	
	For the ANSI network variant, the only valid ISUP version is:
	
	`ANSI`
	
	For the CCITT network variant, valid ISUP versions are:
	`ACIF_G500`
	`BELGIUM`
	`CCITT`
	`FRANCE1`
	`GERMANY`
	`ITALY`
	`ITU97`
	`MEXICO`
	`NETHERLANDS`
	`Q767`
	`SPAIN`
	`SWEDEN`
	`TAIWAN`
	`UK`
	
	For the China network variant, the only valid ISUP version is:
	`CHINA`
	
	For the NTT network variant, the only valid ISUP versions are:
	
	`NTT`
	`NTT_IC`

Table A-1. ISUP Services Environment Variables *(Continued)*

Environment Variable	Description
`ISUP_CGBA_PER_2CGB` (ANSI only)	Activates the ISUP services feature (ANSI 1992) that returns a Circuit Group Blocking Acknowledgment (CGBA) message to the originator of the Circuit Group Blocking (CGB) message whenever two CGB messages are received within a 5-second timer period. If the SINAP node receives the second CGB message after the timer expires, it discards the first CGB message received and waits to receive the second CGB message within the 5-second timer interval, then sends the CGBA message. When this environment variable is not set, the SINAP/SS7 system uses ANSI 1988 processing. In this case, the SINAP node waits to receive the second CGB message before the timer expires and then sends the CGBA message. If the node receives the second CGB message after the timer expires, it discards the first CGB message. The node starts the timer and waits to receive the second CGB message before sending the CGBA message to the originator of the CGB message.
`ISUP_RSC_BLO_PER_EXP` (ANSI only)	Activates the ISUP services feature to have the following events occur whenever the SINAP node sends a Blocking (BLO) message to a remote point code and starts timers IST12 and IST13 and the remote point code responds with a Reset Circuit (RSC) message: • The SINAP node immediately sends a Release Complete (RLC) message in response to the RSC message received from the remote point code • After timer IST12 expires, the SINAP node sends a BLO message to the remote point code • After timer IST13 expires, the SINAP node stops timer IST12 and uses timer IST13 expiration as the new schedule for sending BLO messages If you do not define this environment variable, the SINAP node immediately responds to an RSC message from the remote point code with a BLO message (without waiting for timer IST12 to expire), then sends an RLC message.

Table A-1. ISUP Services Environment Variables *(Continued)*

Environment Variable	Description
`ISUP_DBL_SEIZE_BITS` (ANSI only)	Enables you to set a value in the range 0-3 for the Double Seizing Control Indicator field of the Circuit Group Characteristics Indicator parameter in the Circuit Validation Response (CVR) message. The SINAP/SS7 system uses this defined value for all configured circuits on the SINAP node. If you do not define the environment variable or the value you specified for the variable is outside the valid range, the SINAP node uses the default value (`0x00`) for the Double Seizing Control Indicator.
`ISUP_REL_NO_ADD_ACC` (ANSI only)	Used to disable the Automatic Congestion Control (ACC) parameter in a Release (REL) message.
`ISUP_CQR_TRANS_FOR_UCIC` (ANSI only)	Causes the state for any unequipped circuits reported in a Circuit Query Response (CQR) message to be *transient*. This variable also prevents the circuits from going into the transient (maintenance) state when certain outage conditions exist (for example, when the ISUP services application or its process manager is not running). When this variable is not defined, the SINAP node reports the state *unequipped* in a CQR message. This action applies to all unequipped circuits, not just those managed by the application **CAUTION:** The actions of the SINAP node resulting from implementation of this feature do not comply with the ANSI standards. Stratus only provides this feature for special case-handling of some 5ESS switches.

Table A-1. ISUP Services Environment Variables *(Continued)*

Environment Variable	Description
`ISUP_NO_UCIC_REPLIES` (ANSI only)	Prohibits the SINAP node from sending Unequipped Circuit Identification Code (UCIC) messages to the remote node in response to messages destined for unequipped or unconfigured circuits. This affects *all* unequipped circuits, not just those managed by the application. Normally, for unequipped circuits, the SINAP node sends UCIC messages to the remote node. **CAUTION:** The actions of the SINAP node resulting from implementation of this feature do not comply with the ANSI standards. Stratus only provides this feature for special case-handling of some 5ESS switches.
`ISUP_GENERIC_NAME` (ANSI only)	Enables use of the Generic Name parameter in the Initial Address Message (IAM). When this variable is not defined, the SINAP node treats the Generic Name parameter as an unrecognized parameter whenever the node detects it in an IAM.
`ISUP_UPU_FEATURE`	This enables UPU message to be sent to each remote point code, for any incoming ISUP message, when ISUP application is down.

Appendix B

ISUP Services Messages and Parameters

This appendix contains tables that list ISUP services message types and parameters. For a complete list of ISUP services-related message types and parameters, see ITU-T Recommendation Q.763, ANSI Standard T1.113.3, and Japan NTT Specification NTT-Q763-a.

Table B-1 lists the ISUP services message-type acronyms. The table also lists the binary code for each message type (as it appears in the ITU-T Recommendation, ANSI Standard, or Japan NTT Specification) and the message's hexadecimal code (as it appears in the Message Type field in the BITE display of an ISUP services message).

NOTE

The message types in Table B-1are listed alphabetically by acronym spelling, which differs from the order in which they are listed in ITU-T Recommendation Q.763, ANSI Standard T1.113.3, or in Japan NTT Specification NTT-Q763-a.

Table B-1. ISUP Services Message Types

Acronym	Message Type	Table Reference ITU-T	ANSI	NTT	Binary	Hex
ACM	Address Complete	21	5	5	00000110	06
ANM	Answer	22	6	22	00001001	09
BLA	Blocking Acknowledgment	39	18	14	00010101	15
BLO	Blocking	39	18	14	00010011	13
CCR	Continuity Check Request	39	18		00010001	11
CFN	Confusion	26	8A		00101111	2F
CGB	Circuit Group Blocking	40	20		00011000	18
CGBA	Circuit Group Blocking Acknowledgment	40	20		00011010	1A
CGU	Circuit Group Unblocking	40	20		00011001	19

Table B-1. ISUP Services Message Types *(Continued)*

Acronym	Message Type	Table Reference ITU-T	ANSI	NTT	Binary	Hex
CGUA	Circuit Group Unblocking Acknowledgment	40	20		00011011	1B
CHG	Charge Information			16	11111110	FE
CON	Connect	27			00000111	07
COT	Continuity	28	9		00000101	05
CPG	Call Progress	23	6A	7	00101100	2C
CQM	Circuit Group Query	41	21	15	00101010	2A
CQR	Circuit Group Query Response	24	6B	8	00101011	2B
CRA	Circuit Reservation Acknowledgment		18		11101001	
CRM	Circuit Reservation		6C		11101010	
CVR	Circuit Validation Response		6D		11101011	
CVT	Circuit Validation Test		18		11101100	
EXM	Exit		9A		11101101	
FAA	Facility Accepted	42			00100000	20
FAC	Facility	45			00110011	33
FAR	Facility Request	42			00011111	1F
FOT	Forward Transfer	37	17		00001000	08
FRJ	Facility Reject	29			00100001	21
GRA	Circuit Group Reset Acknowledgment	25	21	9	00101001	29
GRS	Circuit Group Reset	41	21	15	00010111	17
IAM	Initial Address	32	14	14	00000001	01
IDR	Identification Request	47			00110110	36
INF	Information	30	12		00000100	04
INR	Information Request	31	13		00000011	03
IRS	Identification Response	48			00110111	37

Table B-1. ISUP Services Message Types *(Continued)*

Acronym	Message Type	Table Reference ITU-T	ANSI	NTT	Binary	Hex
LPA	Loopback Acknowledgment	39	18		00100100	24
NRM	Network Resource Management	46			00110010	32
OLM	Overload	39	18		00110000	30
PAM	Pass-Along	43	23		00101000	28
REL	Release	33	14A	11	00001100	0C
RES	Resume	38	17A	13	00001110	0E
RLC	Release Complete	34	18	12	00010000	10
RSC	Reset Circuit	39	18	14	00010010	12
SAM	Subsequent Address	35			00000010	02
SGM	Segmentation	49			00111000	38
SUS	Suspend	38	17	13	00001101	0D
UBA	Unblocking Acknowledgment	39	18	14	00010110	16
UBL	Unblocking	39	18	14	00010100	14
UCIC	Unequipped Circuit Identification Code	39	18		00101110	2E
UPA	User Part Available	44			00110101	35
UPT	User Part Test	44			00110100	34
USR	User-to-User Information	36			00101101	2D

Table B-2 presents a list of the parameters used in ISUP services messages. The column labeled “Subclause” indicates the section in ITU-T Recommendation Q.763, the section in ANSI Standard T1.113.3, and the section in Japan NTT Specification NTT-Q763-a in which the parameter is described. The column labeled “Decimal” indicates the numeric representation of the parameter as it is logged in BITE. For information on whether a parameter is mandatory or optional for a particular type of message, see the appropriate tables in ITU-T Recommendation Q.763, ANSI Standard T1.113.3 and Japan NTT Specification NTT-Q763-a.

Table B-2. ISUP Services Parameters

Parameter	Subclause ITU-T	ANSI	NTT	Decimal
Access delivery information	3.2			46
Access transport	3.3	3.1A	3.2	03
Automatic congestion level	3.4	3.2A		39
Backward call indicators	3.5	3.3	3.4	17
Business group		3.3A		
Call diversion information	3.6			54
Call history information	3.7			45
Call reference	3.8	3.5	3.6	01
Called party number	3.9	3.6	3.7	04
Calling party number	3.10	3.7	3.8	10
Calling party's category	3.11	3.8	3.9	09
Circuit state indicator	3.14	3.11B	3.12	38
Carrier identification		3.8A		
Carrier selection information		3.8B		
Cause indicators	3.12	3.9		18
Charge information			3.53	251
Charge information type			3.54	250
Charge number		3.10		
Charging zone (toll area) information			3.51	253
Circuit group supervision message type indicator	3.13	3.11		21
Closed user group interlock code	3.15		3.13	26
Connected number	3.16			33
Connection request	3.17	3.15		13
Continuity indicators	3.18	3.16		16

Table B-2. ISUP Services Parameters *(Continued)*

Parameter	Subclause ITU-T	Subclause ANSI	Subclause NTT	Decimal
Contractor number			3.55	249
Echo control information	3.19			55
Egress service		3.16A		
End of optional parameters	3.20			00
Event information	3.21	3.17A	3.18	36
Facility indicator	3.22			24
Forward call indicators	3.23	3.20	3.20	07
Freephone indicators (reserved)	See Note 1.			65
Generic address		3.20A		
Generic digits	3.24	3.20B		68
Generic name (See Note 2)		3.20C		199
Generic notification	3.25			44
Generic (Universal) number	3.26		3.56	67
Generic reference (reserved)	3.27			66
Hop counter (reserved)	See Note 1.			61
Information indicators	3.28	3.22		15
Information request indicators	3.29	3.23		14
Jurisdiction information		3.23A		
Location number	3.30			63
MCID request indicator	3.31			59
MCID response indicator	3.32			60
Message compatibility information	3.33			56
MLPP precedence	3.34			58
Nature of connection indicators	3.35	3.24	3.23	06
Network function type			3.50	254

Table B-2. ISUP Services Parameters *(Continued)*

Parameter	Subclause ITU-T	ANSI	NTT	Decimal
Network specific facilities	3.36			47
Network transport		3.24A		
Notification indicator		3.24B		
Optional backward call indicators	3.37	3.24C		41
Optional forward call indicators	3.38			08
Original called number	3.39	3.26		40
Origination ISC point code	3.40			43
Originating line information		3.26A		
Parameter compatibility information	3.41			57
Propagation delay counter	3.42			49
Range and status	3.43	3.27	3.27	22
Reason no number indicated			3.57	245
Redirecting number	3.44	3.27A	3.28	11
Redirection information	3.45	3.29	3.29	19
Redirection number	3.46			12
Redirection number restriction	3.47			64
Remote operations	3.48			50
Service activation	3.49	3.29C		51
Service code		3.29D		
Signalling point code	3.50			30
Special processing request		3.30A		
Subsequent number	3.51			05
Suspend/Resume indicators	3.52	3.31A	3.33	34
Transaction request		3.31B		
Transit network selection	3.53	3.31C		35

Table B-2. ISUP Services Parameters *(Continued)*

Parameter	Subclause ITU-T	ANSI	NTT	Decimal
Transmission medium requirement	3.54		3.35	02
Transmission medium requirement prime	3.55			62
Transmission medium used	3.56	3.32A		53
User service information	3.57	3.33	3.36	29
User service information prime	3.58			48
User teleservice information	3.59			52
User-to-user indicators	3.60		3.37	42
User-to-user information	3.61		3.38	32

NOTE

The format is not currently provided.

Appendix C
ISUP Services Error Messages

When a CASL function call is unsuccessful, the system returns a value in that function's `errno` field, corresponding to a probable reason for the failure. The `errno` value can then be cross-referenced to the messages contained in Table C-1 for a description of the error. The last column of the table "ISUP Function," lists the ISUP services functions which can return the error listed.

The range of `errno` values assigned to each type of ISUP services-related CASL function call errors are from 8000 to 8999 and are defined in the SINAP/SS7 `ca_error.h` include file.

Table C-1. ISUP Services Error Message Summary

`errno` Value	Error Message	Description	ISUP Function
8001	IS_ERR_TOO_MANY_PC	Indicates the system does not have enough memory to add the requested point code(s).	isup_add_rem_ssp() isup_cfg_process()
8002	IS_ERR_ALREADY_CONFIG	Indicates the application process is already configured.	isup_cfg_process()
8003	IS_ERR_CONFIG_FORMAT	Indicates there is an invalid configuration value in the internal message being used.	isup_cfg_process()
8004	IS_ERR_PROC_NOTCONFIG	Indicates the application process is not configured for ISUP services.	isup_add_circuit() isup_add_rem_ssp() isup_add_standby_rem_ssp() isup_alloc_msg() isup_cfg_circuit_info() isup_del_circuit() isup_del_rem_ssp() isup_get_circuit_status() isup_free_msg() isup_get_msg()

Table C-1. ISUP Services Error Message Summary *(Continued)*

errno Value	Error Message	Description	ISUP Function
8004 cont.	IS_ERR_PROC_NOTCONFIG		isup_purge_circuit() isup_put_msg() isup_restore_circuit_state() isup_start_utimer() isup_stop_utimer()
8005	IS_ERR_NO_ROUTESET	Indicates the point code defined in the command does not have a route set provisioned.	isup_add_rem_ssp() isup_add_standby_rem_ssp()
8006	IS_ERR_PC_OWNED	Indicates the requested point code is already owned by another ISUP process.	isup_add_rem_ssp() isup_add_standby_rem_ssp()
8007	IS_ERR_PC_NOT_CONFIG	Indicates that the requested point code is not in the table.	
8008	IS_ERR_NOT_REGISTERED	Indicates the calling process is not registered for ISUP.	isup_cfg_process()
8009	IS_ERR_CONFIG_PARAMETER	Indicates the command includes an invalid configuration parameter.	isup_cfg_process() isup_add_circuit() isup_cfg_circuit_info()
8010	IS_ERR_MEMORY_NOT_AVAIL	Indicates the system does not have enough memory to complete the requested task.	isup_alloc_msg() isup_cfg_process() isup_free_msg() isup_start_utimer()
8011	IS_ERR_ISMG_IPC_FAILURE	Indicates a failure in the Interprocess Communications (IPC) with ISMG	isup_add_rem_ssp() isup_cfg_process() isup_del_rem_ssp()
8012	IS_ERR_INTERNAL_FAILURE	Indicates an internal failure in ISUP.	isup_add_circuit() isup_cfg_process() isup_restore_circuit_state()
8013	IS_ERR_ALREADY_ADDED	Indicates that the configuration already contains the element you are trying to add.	isup_add_circuit() isup_add_rem_ssp() isup_add_standby_rem_ssp()

Table C-1. ISUP Services Error Message Summary *(Continued)*

errno Value	Error Message	Description	ISUP Function
8014	IS_ERR_LIMIT_EXCEEDED	Indicates the configuration limit is exceeded and no more elements can be added.	isup_add_circuit() isup_add_rem_ssp() isup_add_standby_rem_ssp()
8015	IS_ERR_PRIMITIVE_EVENT	Indicates the command contains an invalid primitive event code.	isup_alloc_msg() isup_put_msg()
8016	IS_ERR_UNKNOWN_REM_SSP	Indicates you have entered an unknown value for the remote Service Switching Point (SSP) and the command cannot execute.	isup_add_circuit() isup_del_rem_ssp()
8017	IS_ERR_INVALID_CIRCUIT	Indicates you have entered an invalid value for the circuit ID and the command cannot execute.	isup_add_standby_rem_ssp() isup_cfg_circuit_info() isup_change_pc_mode() isup_del_circuit() isup_get_circuit_status() isup_purge_circuit() isup_put_msg() isup_restore_circuit_state() isup_start_utimer()
8018	IS_ERR_CIRCUIT_IS_BUSY	Indicates that the selected circuit is busy.	isup_del_circuit()
8019	IS_ERR_CIRCUIT_DEFINED	Indicates that this circuit is not defined for the given SSP.	isup_del_rem_ssp()
8021	IS_ERR_UTIMER_DURATION	Indicates you have entered an invalid value for the user timer duration.	isup_start_utimer()
8022	IS_ERR_INVALID_UTIMER	Indicates you have entered an invalid value for the primitive, event type, or timer ID.	isup_stop_utimer()
8023	IS_ERR_NOT_SUPPORTED	Indicates the command si not supported for this version of ISUP services.	isup_cfg_circuit_info()

Table C-1. ISUP Services Error Message Summary *(Continued)*

errno Value	Error Message	Description	ISUP Function
	IS_ERR_PC_WRONG_MODE	Indicates the application change request specified a point code mode change to a state that already exists for the point code.	isup_change_pc_mode()
	IS_ERR_INVALID_CIRCUIT_STATE	Indicates the circuit state is not set to a valid state.	isup_restore_circuit_state()

Appendix D
Belgian Version of ISUP Services

The Belgian version of ISUP services feature adheres to the Signaling Procedures for the Signaling System No. 7 ISDN User Part for Belgacom ISUP, March 1999. This specification is based on the 1993 edition of the ITU-T Recommendations for ISUP (Q.761 -Q.764).

Implementing the Belgian Version of ISUP Services

To implement the Belgian version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh])` and set the variable to `BELGIUM`.

```
ISUP_FEATURE=BELGIUM
```

Differences Between the Belgian and CCITT Versions of ISUP Services

The Belgian version of ISUP services differs from the CCITT/ITU-T version of ISUP services that SINAP/SS7 currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences in the message types, message parameters, and parameter field values between the two versions.

For more detailed information on the differences in the Belgian version of ISUP services, see the ITU-T Recommendations for Switching and Signaling Specifications of SS7 - ISDN User Part (ISUP) Q.761 through Q.764.

Message Types

The Belgian version of ISUP services supports a restricted set of messages which are also supported in the CCITT version of ISUP services.

Supported Message Types

As directed by the Signaling Procedures for the Signaling System No. 7 ISDN User Part for Belgacom ISUP, the Belgian version of ISUP services supports a subset of the ITU-T message types described in Q.763. These are listed in Table D-1.

Table D-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CFN	Confusion
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CON	Connect
COT	Continuity
CPG	Call Progress
FAA	Facility Accepted
FAR	Facility Request
FRJ	Facility Rejected
GRA	Circuit Group Reset Acknowledge
GRS	Circuit Group Reset
IAM	Initial Address
INF	Information
INR	Information Request

Table D-1. Supported Message Types *(Continued)*

Message Type	Description
`LPA`	Loopback Acknowledgment
`REL`	Release
`RLC`	Release Complete
`RSC`	Reset Circuit
`SAM`	Subsequent Address
`SUS`	Suspend
`UBA`	Unblocking Acknowledgment
`UBL`	Unblocking
`UPA`	User Part Available
`UPT`	User Part Test
`USR`	User-to-User Information

Unsupported Message Types

The Belgian version of ISUP services treats the message types in Table D-2 (which are supported by the CCITT version of ISUP services) as unrecognized and ignores them.

Table D-2. Unsupported Message Types

Message	Description
`CHG`	Charge Information
`CQM`	Circuit Group Query
`CQR`	Circuit Group Query Response
`FAC`	Facility
`FOT`	Forward Transfer
`IDR`	Identification Request
`IRS`	Identification Response
`NRM`	Network Resource Management
`OLM`	Overload
`PAM`	Pass-Along

Table D-2. Unsupported Message Types *(Continued)*

Message	Description
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code

Message Parameters

Message parameters supported in the Belgian version of ISUP services are a subset of those supported in the CCITT version of ISUP services.

Supported Message Parameters

The Belgian version of ISUP services supports a subset of ITU-T parameters described in Q.763. These are:

- Access Transport
- Backward Call Indicators
- Call Diversion Information
- Call History Information
- Called Party Number
- Calling Party's Category
- Calling Party Number
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code
- Connected Number
- Continuity Indicators
- Echo Control Information
- End of Optional Parameters
- Event Information
- Facility Indicator
- Forward Call Indicators
- Generic Notification
- Generic Number
- Information Indicators
- Information Request Indicators

- Location Number
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- Nature of Connection Indicators
- Network Specific Facilities
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Original Called Number
- Origination ISC Point Code
- Parameter Compatibility Information
- Propagation Delay Counter
- Range and Status
- Redirecting Number
- Redirection Information
- Redirection Number
- Redirection Number Restriction
- Service Activation
- SignallingPoint Code
- Subsequent Number
- Suspend/Resume Indicators
- Transmission Medium Requirement
- User Service Information
- User Teleservice Information
- User-to-User Indicators
- User-to-User Information

Unsupported Message Parameters

The Belgian version of ISUP services does not support the message parameters contained in the following list (which are supported in the CCITT version of ISUP services).

- Access Delivery Information

- Automatic Congestion Level
- Call Reference
- Circuit State Indicator
- Connection Request
- Freephone Indicators
- Generic Digits
- MLPP Precedence
- Remote Operations
- Service Activation
- Transit Network Selection
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information Prime

Parameter Field Values

The Belgian version of ISUP services supports the parameter field values contained in Table D-3.

Table D-3. Parameter Field Values

Message Parameter	Value
Called Party Number	INN indicator always = `1`
Cause Indicators	Expired timer passed back in diagnostic field
Information Indicators	Always coded `0`: hold provided, charge information response and solicited information
Information Request Indicators	Always coded `1`: malicious call identification request

Disabled Options

In the Belgian version of ISUP services, only the following protocol options which are included in the CCITT version of ISUP services have been disabled:

- Message segmentation
- Sending Unequipped Circuit Identification Code (UCIC) message in reply to a request for an unequipped circuit

Appendix E
Chinese Version of ISUP Services

The Chinese version of ISUP services adheres to the Technical Regulations of the Ministry of Telecommunications of the People's Republic of China, Document #YDN 038-1997, Domestic Signaling System No. 7 Technical Standards, Integrated Services User Part (ISUP), Provisional Regulations, issued May 1, 1997. This specification is based on the 1993 edition of the ITU-T Recommendations Q.761 - Q.764, Specifications of Signalling System No. 7, ISDN User Part, Volume VI, Fascicle VI.8.

Implementing the Chinese Version of ISUP Services

To implement the Chinese version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh])` and set the variable to `China`.

```
ISUP_FEATURE=CHINA
```

NOTE

The China network variant must be configured on the SINAP node in order to activate the Chinese version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).

Differences Between the Chinese and CCITT Versions of ISUP Services

The Chinese version of ISUP services differs from the CCITT/ITU-T version of ISUP services that SINAP/SS7 currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences in the message types, message parameters, and parameter field values between the two versions.

For more detailed information on the differences in the Chinese version of ISUP services, see the ITU-T Recommendations for Switching and Signalling Specifications of SS7 - ISDN User Part (ISUP) Q.761 through Q.764.

NOTE

Section 5 of YDN 038 corresponds to ITU Q.762 and Section 6 of YDN 038 corresponds to ITU Q.763.

Message Types

This section describes the differences in messages between the Chinese version of ISUP services and the CCITT version of ISUP services which are supported by the current SINAP software release.

The message types supported by the Chinese version of ISUP services are described in YDN 038 Section 5.3 "Signaling Messages," Table 3/6 "Message Type Codes," and in the message format tables of Section 6.

Supported Message Types

As directed by the YDN 038 specification, the Chinese version of ISUP services supports a subset of the CCITT message types described in Q.763. These are listed in Table E-1.

Table E-1. Supported Message Types

Message Type	Description
`ACM`	Address Complete
`ANM`	Answer
`BLA`	Blocking Acknowledgment
`BLO`	Blocking
`CCR`	Continuity Check Request
`CFN`	Confusion
`CGB`	Circuit Group Blocking
`CGBA`	Circuit Group Blocking Acknowledgment
`CGU`	Circuit Group Unblocking
`CGUA`	Circuit Group Unblocking Acknowledgment
`CON`	Connect
`COT`	Continuity
`CPG`	Call Progress
`CQM`	Circuit Group Query

Table E-1. Supported Message Types *(Continued)*

Message Type	Description
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
GRA	Circuit Group Reset Acknowledge
GRS	Circuit Group Reset
IAM	Initial Address
IDR	Identification Request
INF	Information
INR	Information Request
IRS	Identification Response
NRM	Network Resource Management
PAM	Pass-Along
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SGM	Segmentation
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

New Message Types

In addition to the supported message types listed in Table E-1, which are also supported by the CCITT version of ISUP services, three new message types are defined in the Chinese version of ISUP services and are available for use when the ISUP services environment variable has been set. These new message types are listed in Table E-2 and are **not** supported by the CCITT version of ISUP services.

Table E-2. New Message Types

Message Type	Description
`CCL`	Calling Party Clear
`MPM`	Metering Pulse
`OPR`	Operator Signal

Unsupported Message Types

The Chinese version of ISUP services treats the message types in Table E-3 (which are supported by the CCITT version of ISUP services) as unrecognized messages and ignores them.

Table E-3. Unsupported Message Types

Message Type	Description
`CMC`	Call Modification Completed
`CMR`	Call Modification Request
`CMRJ`	Call Modification Reject
`LPA`	Loopback Acknowledgment
`OLM`	Overload
`UCIC`	Unequipped Circuit Identification Code

Message Parameters

Message parameters supported in the Chinese version of ISUP services are a subset of those supported in the CCITT version of ISUP services.

The message parameters supported by the Chinese version of ISUP services are described in YDN 038 Section 6.3 “ISDN User Part Parameters,” Table 4/6 “Parameter Names and Codes,” and in the message format tables of Section 6.

Supported Message Parameters

The Chinese version of ISUP services recognizes several message parameters that are also supported in the CCITT version of ISUP services. These are:

- Access Transport
- Automatic Congestion Level
- Backward Call Indicators
- Call Diversion Information
- Call History Information
- Called Party Number
- Calling Party Number
- Calling Party’s Category
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Circuit State Indicator
- Closed User Group Interlock Code
- Connected Number
- Continuity Indicators
- Echo Control Information
- Event Information
- Facility Indicators
- Forward Call Indicators
- Generic Digits
- Generic Number
- Information Indicators
- Information Request Indicators
- Location Number
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- Nature of Connection Indicators
- Network Specific Facilities
- Notification Indicator

- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Original Called Number
- Parameter Compatibility Information
- Pass-Along Information
- Propagation Delay Counter
- Range and Status
- Redirecting Number
- Redirection Information
- Redirection Number
- Redirection Number Restriction
- Subsequent Number
- Suspend/Resume Indicators
- Transit Network Selection
- Transmission Medium Requirement
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information
- User Service Information Prime
- User Teleservice Information
- User-to-User Indicators
- User-to-User Information

Two optional parameters supported by the CCITT version of ISUP services are defined with a different format for use in the Chinese version of ISUP services. The length of the following parameters has been changed from 2 to 3 bytes to accommodate the ANSI-style 24-bit point code used in the Chinese version of ISUP services:

- Origination ISC Point Code
- Signalling Point Code

New Message Parameters

In addition to the supported message parameters listed in the previous section, the following new parameter is supported in the Chinese version of ISUP services for the Metering Pulse (MPM) message. This parameter is mandatory and has a fixed length of two bytes. This parameter is not supported in the CCITT version of ISUP services.

- Charge Information

Unsupported Message Parameters

The Chinese version of ISUP services does not recognize the following message parameters, which are supported in the CCITT version of ISUP services. These are:

- Access Delivery Information
- Call Modification Indicators
- Call Reference
- Connection Request
- MLPP Precedence
- Remote Operations
- Service Activation

Allowed Parameter Values

The parameter values contained in Table E-4 are specified for the Chinese version of ISUP services and are described in YDN 038 Section 6.3 "ISDN User Part Parameters."

Table E-4. Allowed Parameter Values

Parameter	Field	Allowable Value(s)

Called Party Number	Nature of Address Indicator	`SUBSNUM (=1)`
		`NATNUM (=3)`
		`INTNATNUM (=4)`

Calling Party Number	Nature of Address Indicator	`ADDR_NOTPRSNT (=0)`
		`SUBSNUM (=1)`
		`NATNUM (=3)`
		`INTNATNUM (=4)`

Calling Party Number (cont.)	Screening Indicator	`USRPROV (=1)` `NETPROV (=3)`

Table E-4. Allowed Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
Calling Party's Category	Calling Party Category	CAT_UNKNOWN (=0) CAT_OPLANGFR (=1) CAT_OPLANGENG (=2) CAT_OPLANGGER (=3) CAT_OPLANGRUS (=4) CAT_OPLANGSP (=5) CAT_ADMIN1 (=6) [1] CAT_ADMIN2 (=7) CAT_ADMIN3 (=8) [2] CAT_NATOPR (=9) CAT_ORD (=10) CAT_PRIOR (=11) CAT_DATA (=12) CAT_TEST (=13) CAT_PAYPHONE (=15) CAT_ORDNOCHG (=240) [3] CAT_ORDPERIOD (=241) [3] CAT_ORDMETER (=242) [3] CAT_ORDPRINT (=243) [3] CAT_PRIORNOCHG (=244) [3] CAT_PRIORPERIOD (=245) [3] CAT_ORDLOCAL (=248) [3]
Cause Indicators	Cause Value	CCUNALLOC (=1) CCNORTTOTSFNET (=2) CCNORTTODEST (=3) CCSENDSPCLTONE (=4) CCMISDIALDTRNK (=5) CCQ850PREEMT (=8) [4] CCQ850PREECIRVD (=9) [4] CCCALLCLR (=16) CCUSRBSY (=17) CCNOUSRRSP (=18) CCNOANSWR (=19) CCSUBSCRABSENT (=20) CCCALLRJT (=21) CCNMBRCHG (=22) CCDESTOUTORD (=27) CCADDRINCOMP (=28) CCFACREJ (=29) CCNORMUNSPEC (=31) CCNOCIRCUIT (=34)

Table E-4. Allowed Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
	Cause Value (cont.)	CCACCINFDISC (=43) CCREQUNAVAIL (=44) CCPRECCALLBLKD (=46)[4] CCRESCUNAVAIL (=47) CCFACNOTSUB (=50) CCOGBARRDCUG (=53) CCINCBARRDCUG (=55) CCNOTAUTHBCAP (=57) CCBCAPUNAVAIL (=58) CCINCOGACCSUBCLS (=62) CCSERVUNAVAIL (=63) CCBCAPNOTIMP (=65) CCFACNOTIMP (=69) CCRESTDIG (=70) CCSERVNOTIMP (=79) CCCUNOTMEMBR (=87) CCINCOMPDEST (=88) CCNOTEXISTCUG (=90) CCINVTRNSTNET (=91) CCINVMSG (=95) CCNOMSGTYP (=97) CCNOPARAMDISC (=99) CCTMRRECOV (=102) CCNOPARAMPASS (=103)
	Location	CCNOPARAMDISCMSG (=110) CCPROTERR (=111) CCINTRWRK (=127) ILOC_USER (=0) ILOC_PRIVNETLU (=1) ILOC_PUBNETLU (=2) ILOC_TRANNET (=3) ILOC_PUBNETRUS (=4) ILOC_PRIVNETRUS (=5) ILOC_INTNET (=7) ILOC_NETINTER (=10)
Connected Number	Nature of Address	ADDR_NOTPRSNT (=0) SUBSNUM (=1) NATNUM (=3) INTNATNUM (=4)
	Screening Indicator	USRPROV (=1) NETPROV (=3)

Table E-4. Allowed Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
Generic Number	Number Qualifier Indicator	NQ_ADDCDMNB (=1) NQ_ADDCONMNB (=5) NQ_ADDCGNMB (=6) NQ_ORIGCDNMB (=7) NQ_ORIGRGDNMB (=8) NQ_ORIGRDNMB (=9) NQ_NETSVCNMB (=254) [3]
	Nature of Address Indicator	SUBSNUM (=1) UNKNOWN (=2) NATNUM (=3) INTNATNUM (=4) SPCLNETNUM (=126) [3]
Original Called Number	Nature of Address Indicator	ADDR_NOTPRSNT (=0) SUBSNUM (=1) NATNUM (=3) INTNATNUM (=4)
Redirecting Number	Nature of Address Indicator	ADDR_NOTPRSNT (=0) SUBSNUM (=1) NATNUM (=3) INTNATNUM (=4)
Redirection Number	Nature of Address Indicator	SUBSNUM (=1) NATNUM (=3) INTNATNUM (=4)
Location Number	Screening Indicator	USRPROV (=1) NETPROV (=3)

NOTES

1. Used for the Mandarin Chinese language.
2. Used for the Japanese language.
3. Additional value defined for the Chinese version of ISUP services.
4. Value is not supported by the CCITT version of ISUP services.

Protocol Procedures

This section describes the differences in protocol procedures between the Chinese and CCITT versions of ISUP services currently implemented on the SINAP/SS7 system.

For more information on the protocol procedures implemented in the Chinese version of ISUP services, see YDN 037, Section 7 (which corresponds to ITU-T Q.764).

Unrecognized Messages and Parameters

The procedures for handling unrecognized messages and message parameters are essentially the same in both the Chinese and CCITT versions of ISUP services. The only differences are in the criteria used to determine whether a message or message parameter is considered *unrecognized*.

Interpretation of Spare Field Values

The method of interpreting spare values in parameter fields is also essentially the same in the Chinese and CCITT versions of ISUP services.

The only differences lie in the criteria used to determine whether a field value is considered to be *unrecognized* or *spare* since many of the fields have different sets of allowable values in the Chinese version.

Disabled Options

In the Chinese version of ISUP services, only the following protocol option which is included in the CCITT version of ISUP services has been disabled.

- Sending Unequipped Circuit Identification Code (UCIC) message in reply to a request for an unequipped circuit

ISUP Services Timers

The Chinese version of ISUP services supports all the same ISUP services timers as the CCITT version of ISUP services.

API Support for Parameters with 24-bit Point Codes

As mentioned previously in the section “Supported Message Parameters,” the Chinese version of ISUP services changes the length of two optional message parameters from 2 bytes to 3 bytes to accommodate the 24-bit point code used by China’s SS7 stack. These parameters are the Origination ISC Point Code and the Signalling Point Code.

Origination ISC Point Code

The Origination ISC Point Code can appear in an Initial Address (IAM) message and is represented in the API by the `SiConEvnt` structure element:

```
SiSigPC24     orgPteCde24;
```

This applies to the connection setup primitives for sending and receiving and IAM.

Signaling Point Code

The Signaling Point Code parameter can appear in a Release (REL) message and is represented at the API by the `SiRelEvnt` structure element:

```
SiSigPC24     sigPointCode24;
```

This applies to the connection release primitives for sending and receiving an REL.

API Support for Sending and Receiving New Messages

The Chinese version of ISUP services define three new messages (previously listed in Table E-2) which are sent or received after a call has been established. Accordingly, three new event types for connection status primitives associated with sending and receiving these messages are defined. These are listed in Table E-5.

Table E-5. New Event Types

Event Type	Description
CALLINGCLR	Calling Party Clear
METERPULSE	Metering Pulse
OPERATORSIG	Operator Signal

The event types apply to the connection status primitives for sending and receiving messages and use the `SiCnStEvnt` structure to convey the parameters of the associated ISUP protocol messages. Each of the three new messages is defined with the Message Compatibility Information parameter as being optional, represented in the `SiCnStEvnt` structure:

```
SiMsgCompInfo     msgCom;
```

The Metering Pulse (MPM) message contains a 2-byte fixed length Charge Information parameter, which is represented in the API by the `SiCnStEvnt` structure element:

```
SiChgInfoMPM     chgInfoMPM;
```

Appendix F
France1 Version of ISUP Services

The France1 version of ISUP services adheres to the CNET 5450 specifications, which is based on the ITU-T Recommendations for ISUP (Q.763 and Q.764).

Implementing the France1 Version of ISUP Services

To implement the France1 version of ISUP services, uncomment the environment variable `ISUP_VERSION` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `FRANCE1`.

```
ISUP_FEATURE=FRANCE1
```

Key Features of the France1 Version of ISUP Services

When the France1 version of ISUP services is activated (and the CCITT network variant is specified) the following ISUP options are implemented:

- The SINAP node does not send automatically send an Unequipped Circuit Identification Code (UCIC) message to the remote point code upon receipt of a messages destined for an unequipped or unconfigured CIC.
- The SINAP/SS7 system checks for parameters restricted by the France1 version of ISUP services and treats them as *unrecognized*.
- Upon receipt of a Reset Circuit (RSC) message for a previously blocked circuit, the SINAP node clears ISUP timers IST14 and IST15 and changes the circuit state to *unblocked*.
- Upon receiving an Initial Address (IAM) message with a continuity check indicator set to `2` (continuity check on previous circuit required) and a receipt of a Continuity (COT) with failure message, the SINAP node performs the following tasks:
  - Stops timer IST8
  - Starts timer IST27 and waits for the Continuity Check Request (CCR) message
  - If a Continuity (COT) message with failure is received, the SINAP node stops timer IST8 but does not start timer IST27 (the node does not perform the continuity recheck

procedure). The SINAP node also sends a Release (REL) message with `Loc=4` and `cause=31` to the remote point code.

- The SINAP node discards an unrecognized message if the circuit state is `IDLE` and the message has no optional parameters.

  If the unrecognized message state is `BUSY` with no optional parameters, the SINAP node sends a Confusion (CFN) message.

  The France1 version of ISUP services handles unexpected messages in the following manner:

  — If the SINAP node receives an unexpected message while waiting for a Continuity (COT) message (timer IST36), it ignores the unexpected message and restarts timer IST36.

  — If a test-call is in progress, the SINAP node ignores the unexpected message.

- Upon receiving an `MTP-PAUSE`, the France1 version of ISUP services default `pauseActn` (`SI_PAUSE_CLRTRAN`) causes the SINAP node to send a Reset Circuit (RSC) message to the remote end only if the circuit is in a transient state. Circuits in stable states are not disturbed.

Differences Between the France1 and CCITT Versions of ISUP Services

The France1 version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences between the two versions.

For more detailed information on the differences in the France1 version of ISUP services, see the CNET 5450 specification.

Message Types

The France1 version of ISUP services supports a subset of messages that are supported in the CCITT version of ISUP services.

Supported Message Types

The France1 version of ISUP services recognizes and processes the subset of standard ITU-T message types listed in Table F-1. All other messages are treated as unrecognized.

Table F-1. Supported Message Types

Message Type	Description
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CFN	Confusion
CON	Connect
COT	Continuity
IAM	Initial Address
INF	Information
INR	Information Request
REL	Release
RSC	Reset Circuit
SGM	Segmentation
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The France1 version of ISUP services treats the message types listed in Table F-2 (also supported by the CCITT version of ISUP services) as unrecognized messages and ignores them.

Table F-2. Unsupported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment

Table F-2. Unsupported Message Types *(Continued)*

Message Type	Description
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CPG	Call Progress
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accept
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass-Along
RES	Resume
SUS	Suspend
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

Message Parameters

Message parameters supported in the France1 version of ISUP services are a subset of those supported in the CCITT version of ISUP services.

Supported Message Parameters

The France1 version of ISUP services supports the message parameters listed in Table F-3.

Table F-3. Supported Message Parameters

Message Parameter	Description
ACCTPORT	Access Transport
BACKCALLIND	Backward Call Indicators
CALDPARTNUM	Called Party Number
CALGPARTCAT	Calling Party Category
CALGPARTNUM	Calling Party Number
CAUSIND	Cause Indicators
CONTIND	Continuity Indicators
ENDOP	End of Optional Parameters
FWDCALLIND	Forward Call Indicator
GENNMB	Generic Number
HEDR	Mask for Message Header Element
INFOIND	Information Indicators
INFOREQIND	Information Request Indicators
LOCNMB	Location Number
MSGCOMP	Message Compatibility Information
NATCONIND	Nature of Connection Indicators
OPBACKCALLIND	Optional Backward Call Indicators
OPFWDCALLIND	Optional Forward Call Indicators
OPPTR	Pointer to Optional Parameters
ORIGCALDNUM	Original Called Number
PARCOMPIN	Parameter Compatibility Information
PASSALNG	Pass-Along Information

Table F-3. Supported Message Parameters *(Continued)*

Message Parameter	Description
`REDIRINFO`	Redirection Information
`SUBSEQNUM`	Subsequent Number
`TRANSMEDREQ`	Transmission Medium Requirement
`USR2USRINFO`	User-to-User Information
`USRSERVINFO`	User Service Information

Unsupported Message Parameters

The France1 version of ISUP services does not support the message parameters listed in Table F-4 (which are supported in the CCITT version of ISUP services).

Table F-4. Unsupported Message Parameters

Message Parameter	Description
`ACCDELINFO`	Access Delivery Information
`AUTOCONGLVL`	Automatic Congestion Level
`BACKVAD`	Backward VAD Indicator
`BUSINESSGRP`	Business Group
`CALLMODIND`	Call Modification Indicators
`CALLTRNSFRNMB`	Call Transfer Number
`CALLTRNSFRREF`	Call Transfer Reference
`CALREF`	Call Reference
`CARRIERID`	Carrier Identification
`CARSELINF`	Carrier Selection Information
`CCBSPARAM`	CCBS Parameter
`CGRPCHARIND`	Circuit Group Characteristic Indicator
`CGRPSUPMTYPIND`	Circuit Group Supervision Message Type Indicator
`CHARGENUM`	Charge Number
`CHRGEINFO`	Call Charge Information
`CHRGERTEINFO`	Charge Rate Information

Table F-4. Unsupported Message Parameters *(Continued)*

Message Parameter	Description
`CIRCSTEIND`	Circuit State Indicator
`CIRIDNAME`	Circuit ID Name
`CIRVALRSPIND`	Circuit Validation Response Indicator
`CLLDIVERS`	Call Diversion Information
`CLLHISTORY`	Call History Information
`CLLI`	Common Language Location Identification Code
`CLSDUGRPCHKRSP`	Closed User Group Check Response
`CLSDUGRPINTCDE`	Closed User Group Interlock Code
`CONNUMB`	Connected Number
`CONREQ`	Connection Request
`ECHOCNTRL`	Echo Control
`EGRESS`	Egress Service
`EVNTINFO`	Event Information
`FACIND`	Facility Indicator
`FACINDINFOR`	Facility Indicator Information
`FACINFIND`	Facility Information Indicator
`FREEPHIND`	Freephone Indicator
`FWDVAD`	Forward VAD Indicator
`GENADDR`	Generic Address
`GENDIGITS`	Generic Digits
`INDEX`	Index
`JCHARGEINFO`	Charge Information
`JCHGINFOTYPE`	Charge Information Type
`JCHGZONEINFO`	Charging Zone Information
`JCONTRCTRNUM`	Contractor Number
`JNETFUNCTYPE`	Network Function Type
`JREASNONUMIND`	Reason No Number Indicated

Table F-4. Unsupported Message Parameters *(Continued)*

Message Parameter	Description
JURISINF	Jurisdiction Information
LOOPPRVNTIND	Loop Prevention Indicator
MCIDREQ	Malicious Call Identification (MCID) Request Indicator
MCIDRSP	Malicious Call Identification (MCID) Response Indicator
MLPPPREC	Multilevel Precedence Preemption (MLPP) Precedence
NATPARFF	National Parameter: FF (Multitype)
NETSPFAC	Network Specific Facilities
NETTRANS	Network Transport
NOTIFIND	Notification Indicator
NOTIFINDC	Generic Notification
NPCHGPTYID	National Parameter: Charged Party IDN
NPCHRGINF	National Parameter: Charging Information
NPEXCHTRKID	National Parameter: Exchange and Trunk ID
NPINCDPNO	National Parameter: Intelligent NetCdPNo
NPSPV	National Parameter: Semipermanent Connection
NPSSCPRICLS	National Parameter: Subscriber Priority Class
NPSSP	National Parameter: Service Switching Point
NPTLN2TLNSG	National Parameter: TLN-to-TLN Signalling
NPUKK	National Parameter: Originating Party Information
ORGLINEINF	Originating Line Information
ORISCCDE	Origination ISC Point Code
OUTGTRKGRPNUM	Outgoing Trunk Group Number
PROPDLYCTR	Propagation Delay Counter
RANGSTAT	Range and Status
REDIRGNUM	Redirecting Number

Table F-4. Unsupported Message Parameters *(Continued)*

Message Parameter	Description
REDIRNUM	Redirection Number
REDIRRESTR	Redirection Number Restriction
REMOTOPER	Remote Operations
SERVACT	Service Activation
SERVACTA	Service Activation
SERVCODE	Service Code Indicator
SIGPTCDE	Signalling Point Code
SPECPROCREQ	Special Processing Request
SUSPRESIND	Suspend/Resume Indicators
TRANNETSEL	Transit Network Selection
TRANSMEDPRM	Transmission Medium Requirement Prime
TRANSMEDUSD	Transmission Medium Used
TRANSREQ	Transaction Request
TRNKOFF	Trunk Offering Information
USR2USRIND	User-to-User Indicators
USRSERVINFOPR	User Service Information Prime
USRSERVINFO	User Service Information

Handling Unknown Parameter Values

The method of handling unknown parameter values by the France1 version of ISUP services adheres to the CNET 5450 Specifications, Part IIIa, Section 2.9.4.8. The specific unknown values for various parameters are contained in Part II, Sections 3.2 through 3.61. Parameter values marked *spare*, *reserved*, *reserved for national use*, and those shaded in Part II of the CNET 5450 specification are considered unknown parameter values.

Table F-5 describes the inbound messages and their parameters (whose unknown values are processed for implementing the France1 version of ISUP services).

Table F-5. Processing Unknown Parameter Values for Inbound Messages

Message	Mandatory Parameters	Optional Parameters
IAM	Called Party Number (variable)	Calling Party Number • Nature of Address Indicator • Numbering Plan Indicator • Address Presentation Restricted Indicator • Screening Indicator • Address Signals
	Calling Party's Category (fixed) Forward Call Indicators (fixed) Nature of Connection Indicators (fixed) Transmission Medium Requirement (fixed)	Optional Forward Call Indicators
INF		Calling Party Number Calling Party's Category
REL	Cause Indicators (variable)	
RLC		Cause Indicator
CFN	Cause Indicator (variable)	

Table F-6 contains the unknown values for the mandatory parameters of the France1 version of ISUP services.

Table F-6. Unknown Values for Mandatory Parameters

Message Parameter	Parameter Fields	Unknown Values
Called Party Number	Nature of Address Indicator	`0x00, 0x01, 0x02,0x05, to 0x7f`
	Numbering Plan Indicator	`0x00, 0x02, 0x05, 0x06, 0x07`
	Address Signals	`0x0a, 0x0d, 0x0e`
Calling Party Number	Nature of Address Indicator	`0x00, 0x01, 0x02,0x05, to 0x7f`
	Numbering Plan Indicator	`0x00, 0x02, 0x05, 0x06, 0x07`
	Address Presentation Restricted Indicator	`0x03`
	Screening Indicator	`0x00, 0x02`
	Address Signals	`0x0a to 0x0f`
Calling Party's Category	Calling Party's Category Indicator	`0x0c, 0x0e, 0x10 to 0xdf, 0xe1 to 0xff`
Cause Indicators	Coding Standard	`0x01, 0x03`
	Location	`0x01, 0x02, 0x06, 0x08, 0x09, 0x0b to 0x0e`
	Cause Value	`5 to 15, 23 to 26, 30, 32, 33, 35 to 37, 39, 40, 45, 46, 48, 49, 51, 52, 54, 56, 59 to 61, 64, 66, 67, 68, 70, 71, to 78, 80 to 86, 89, 92 to 94, 96, 98, 100, 101, 103, 104, to 109, 112 to 126`

Table F-6. Unknown Values for Mandatory Parameters *(Continued)*

Message Parameter	Parameter Fields	Unknown Values
Forward Call Indicators	End-to-End Method Indicator	`0x00, 0x03`
	End-to-End Information Indicator	`0x00, 0x01`
	ISDN User Part Preference Indicator	`0x03`
	SCCP Method Indicator	`0x00, 0x03`
Nature of Connection Indicators	Satellite Indicator	`0x03`
	Continuity Check Indicators	`0x03`
Optional Forward Call Indicators	Closed User Group Indicator	`0x01 to 0x03`
Transmission Medium Requirement	Transmission Medium Requirement	`0x01, 0x04 to 0xff`

Differences in Protocol Procedures

The procedures for handling unrecognized messages and parameters are essentially the same in both the France1 and CCITT versions of ISUP services. The only differences are in the criteria used for determining whether the SINAP/SS7 system treats a message or parameter as *unrecognized*. The SINAP/SS7 system uses table entries in the ISUP services database to determine the criteria to use. When the France1 version of ISUP services is enabled, a restricted set of messages and parameters specific to the France1 version (which are subsets of those supported by the CCITT version of ISUP services) is initiated by the SINAP/SS7 system.

According to procedures defined in the CNET 5450 Specification, a SINAP/SS7 node processes an unknown parameter value in two stages:

1. All parameter values unknown to both the CCITT and France1 versions of ISUP services that resulted in decoding errors are processed first.
2. All unknown parameter values that are specific only to the France1 version of ISUP services are processed next. These parameter values are known values in CCITT.

For messages that are unknown only to the France1 version of ISUP services, the SINAP/SS7 node can take the following actions:

1. Discard the message
2. Discard the message after sending a Release (REL) message to the remote point code and a Release (REL) indication to the application

3. Continue processing the message using the existing CCITT parameter decode error processing
4. Continue processing the message skipping the existing CCITT parameter decode error processing
5. Process Resource Allocation failure
6. Process failure due to software error

Actions 3 and 4 result in processing of unknown parameters (if any) in the inbound ISUP message. The SINAP node processes unknown parameter values that are specific to France1 for all messages even if no decoding error exists.

IAM/INF Messages with Unknown Parameter Values

When the SINAP node receives an Initial Address (IAM) message or an Information (INF) message with an unknown value in a parameter field, the node performs the actions described in Table F-7.

Table F-7. Unknown Parameter Value Processing for IAM/INF Messages

Message Parameter	Parameter Fields	SINAP Action
Called Party Number (see Notes after table)	Nature of Address Indicator	Send REL message with cause value `28`
	Numbering Plan Indicator	
	Address Signals	
	Nature of Address Indicator	Set the Nature of Address Indicator field to the default value `0x03` (national number) Set the Address Presentation Restriction Indicator to the default value of `0x02` (address unavailable) Set the Screening Indicator to the default value `0x03` (network supplied) Reject the address signals

Table F-7. Unknown Parameter Value Processing for IAM/INF Messages *(Continued)*

Message Parameter	Parameter Fields	SINAP Action
Called Party Number (cont.)	Numbering Plan Indicator	Set the Numbering Plan Indicator field to the default value `0x01` (ISDN numbering plan) Set the Address Presentation Restriction Indicator to the default value `0x02` (address unavailable) Set the Screening Indicator to the default value `0x03` (network supplied) Reject the address signals
	Address Presentation Restricted Indicator	Set the Address Presentation Restriction Indicator to the default value `0x01` (presentation restricted)
	Screening Indicator	Set the Screening Indicator to the default value `0x03` (network supplied) Set the Address Presentation Restriction Indicator to the default value `0x02` (address unavailable) Reject the address signals
	Address Signals	Reject the address signals Set the Address Presentation Restriction Indicator to the default value `0x02` (address unavailable) Set the Screening Indicator to the default value `0x03` (network supplied)
Calling Party's Category (see Notes after table)	Calling Party's Category Indicator	Set to default value `0x0a` (ordinary calling subscriber)

Table F-7. Unknown Parameter Value Processing for IAM/INF Messages *(Continued)*

Message Parameter	Parameter Fields	SINAP Action
Forward Call Indicators	End-to-End Method Indicator	Set to default value `0x00` (no end-to-end method available)
	End-to-End Information Indicator	Set to default value `0x00` (no end-to-end information available)
	ISDN User Part Preference Indicator	Send REL message with cause value `111`
	SCCP Method Indicator	Set to default value of `0x00` (No indication)
Nature of Connection Indicators	Satellite Indicator	Set to default value `0x02` (two satellites in the connection)
	Continuity Check Indicators	Set to default value `0x00` (continuity check not required)
Optional Forward Call Indicators	Closed User Group Indicator	Set to default value `0x00` (non-CUG call)
Transmission Medium Requirement	Transmission Medium Requirement	Send REL message with cause value `65`

NOTES

1. When the SINAP node receives an Initial Address (IAM) message with an odd number of Address Signals in the Called Party Number parameter, the node sets the Filler to the default value of `0000`.
2. When the SINAP node receives an Initial Address (IAM) or Information (INF) message with an odd number of Address Signals in the Calling Party Number parameter, the node sets the Filler to the default value of `0000`.

REL/CFN/RLC Messages with Unknown Parameter Values

When a Release (REL), Confusion (CFN), or Release Complete (RLC) message with an unknown value is received in a parameter field the SINAP/SS7 software performs the actions described in Table F-8.

Table F-8. Unknown Parameter Value Processing for REZL/CFN/RLC Messages

Message Parameter	Parameter Fields	SINAP Action
Cause Indicators	Coding Standard	Set to default value `0x00` (standardized ITU-T code)
	Location	Set to default value `0x0a` (network beyond interworking point)
	Cause Value	If unknown value is class `000` or `001` then set to default value `31` (normal, not specified) If unknown value is class `010` then set to default value `47` (resource unavailable, unspecified) If unknown value is class `011` then set to default value `63` (service or option not available, not specified) If unknown value is class `100` then set to default value `79` (service or option not implemented, unspecified) If unknown value is class `101` then set to default value `95` (invalid message, unspecified) If unknown value is class `110` then set to default value `111` (protocol error, not specified) If unknown value is class `111` then set to default value `127` (interworking, not specified)

Appendix G
G500 Version of ISUP Services

The G500 version of ISUP services adheres to the Australian Communications Industry Forum (ACIF) Specification G500 “Signalling System No. 7 Interconnect ISUP,” version 1.0, February 1998. This specification is based on the 1993 edition of the ITU-T Recommendations for ISUP (Q.761 - Q.764).

Implementing the G500 Version of ISUP Services

To implement the G500 version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `ACIF_G500`.

```
ISUP_FEATURE=ACIF_G500
```

Differences Between G500 and CCITT ISUP Services

The G500 version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences in the message types, message parameters, and parameter field values between the two versions.

For more information on the differences in message types supported in the G500 version of ISUP services, see the ACIF G500 specification (sections B.1, “Signalling Messages,” Annex B, Table B-2, “General Function of Messages and Signals,” and Annex C, Table C-3, “Formats and Codes”).

Message Types

The G500 version of ISUP services supports a restricted set of messages which are also supported in the CCITT version of ISUP services.

Supported Message Types

The G500 version of ISUP services supports only messages types listed in Table G-1.

Table G-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLO	Blocking
BLA	Blocking Acknowledgment
CFN	Confusion
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
CPG	Call Progress
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The G500 version of ISUP services treats the message types in Table G-2 (which are supported by the CCITT version of ISUP services) as unrecognized messages and ignores them.

Table G-2. Unsupported Message Types

Message Type	Description
CCR	Continuity Check Request
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
COT	Continuity
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
INF	Information
INR	Information Request
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass -Along
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

Message Parameters

Message parameters supported in the G500 version of ISUP services are a subset of the message parameters supported in SINAP/SS7's CCITT version of ISUP services.

Supported Message Parameters

The G500 version of ISUP services supports the message parameters listed in Table G-3. Note that wherever the G500 ISUP services parameter field values differ from those of the CCITT version of ISUP services, the chart lists the values that are specific to G500 ISUP. Field values that are the same in both versions are left blank.

Differences in parameter field values are address by ACIF G500, Section C.3, "ISDN User Part Parameters" of Annex C on "Formats and Codes."

Table G-3. Supported Message Parameter Values

Message Parameter	Parameter Fields	Field Values Allowed
Access Transport		
Automatic Congestion Level		
Backward Call Indicators	Called Party's Status	`CADSTAT_NOIND =0` `CADSTAT_SUBFREE =1`
	End-to-End Method Indicator	`E2EMTH_NOMETH =0`
	End-to-End Information Indicator	`E2EINF_NOINFO =0`
	Holding Indicator	`HOLD_NOTREQD =0`
	SCCP Method Indicator	`SCCPMTH_NOIND =0`
Called Party Number	Nature of Address Indicator	`UNKNOWN =2`
	Numbering Plan Indicator	`ISDNNUM =1`
Calling Party Number	Nature of Address Indicator	`NATNUM =3`
	Number Incomplete Indicator	`NBMCMLTE =0`
	Numbering Plan Indicator	`ISDNNUM =1`
	Address Presentation Restricted	`PREASALLOW =0` `PRESREST =1`
	Screening Indicator	`USRPROV =1` `NETPROV =3`

Table G-3. Supported Message Parameter Values *(Continued)*

Message Parameter	Parameter Fields	Field Values Allowed
Calling Party's Category	Calling Party's Category	CATG500_INTNAT =10 CATG500_INHDIV =239 CATG500_INTNATOP =241 CATG500_NATOP =242 CATG500_CUSTCL =243 CATG500_PARTCLI =244 CATG500_CUSTCLIMM =245 CATG500_MOBILE =247 CATG500_CCRCUST =249 CATG500_CCRCUSTMM =250 CATG500_TEST =251 CATG500_PAYCLI =253 CATG500_PAYCLIMM =254
Cause Indicators	Coding Standard	CSTD_CCITT =0 ILOC_USR =0 ILOC_PRIVNETLU =1 ILOC_PUBNETLU =2 ILOC_PUBNETRUS =4 ILOC_PRIVNETRUS =5 ILOC_INTNET =7 ILOC_NETINTER =10
	Cause-Code Value	CCUNALLOC =1 CCNORTTODEST =3 CCMISDIALDTRNK =5 CCCALLCLR =1 CCUSRBSY =17 CCNOUSRRSP =18 CCNOANSWR =19 CCCALLRJT =21 CCNMBRCHG =22 CCDESTOUTORD =27 CCADDRINCOMP =28 CCFACREJ =29 CCNORMUNSPEC =31 CCNOCIRCUIT =34 CCNETAOL =38 CCTMPFAIL =41 CCSWTCHCONG =42 CCACCINFDISC =43 CCREQUNAVAIL =44 CCRESCUNAVAIL =47 CCNOTAUTHBCAP =57 CCBCAPUNAVAIL =58

Table G-3. Supported Message Parameter Values *(Continued)*

Message Parameter	Parameter Fields	Field Values Allowed
Cause Indicators (cont.)	Cause-Code Value (cont.)	CCSERVUNAVAIL =63 CCBCAPNOTIMP =65 CCSERVNOTIMP =79 CCINCOMPDEST =88 CCINVMSG =95 CCNOMSGTYP =97 CCNOPARAMDISC =99 CCTMRRECOV =102 CCNOPARAMPASS =103 CCNOPARAMDISCMSG =110 CCPROTERR =111 CCINTRWRK =127
Circuit Group Supervision Message Type Indicator		
Event Information	Event	EV_ALERT =1 EV_PROGRES =2 EV_INBAND =3
	Event Presentation Restricted Indicator	EVPR_NOIND =0
Forward Call Indicators	End-to-End Method Indicator	E2EMTH_NOMETH =0
	End-to-End Information Indicator	E2EINF_NOINFO =0
	SCCP Method Indicator	SCCPMTH_NOIND =0
Message Compatibility Information		
Nature of Connection Indicators	Continuity Check Indicators	CONTCHK_NOTREQ =0 CONTCHK_PREV =2
Optional Backward Call Indicators		
Original Called Number	Nature of Address Indicator	UNKNOWN =2
	Numbering Plan Indicator	ISDNNUM =1
	Address Presentation Restricted	PREASALLOW =0 PRESREST =1
Parameter Compatibility Information		

Table G-3. Supported Message Parameter Values *(Continued)*

Message Parameter	**Parameter Fields**	**Field Values Allowed**
Range and Status		

Redirecting Number	Nature of Address Indicator
	Numbering Plan Indicator
	Address Presentation Restricted
		`NATNUM =3`
		`ISDNNUM =1`
		`PREASALLOW =0`
		`PRESREST =1`

Redirection Information		
Subsequent Number		
Suspend/Resume Indicators		

Transmission Medium Requirement	Transmission Medium Requirement	`TMR_SPEECH =0`
		`TMR_64KBITS =2`
		`TMR_31KHZ =3`

Table G-3. Supported Message Parameter Values *(Continued)*

Message Parameter	Parameter Fields	Field Values Allowed

User Service Information	Information Transfer Capability	`ITC_SPEECH =0`
		`ITC_UNRDIG =8`
		`ITC_RESDIG =9`
		`ITC_A31KHZ =16`
		`ITC_A7KHZ =17`

	Information Transfer Rate	`ITR_PKT =0` `ITR_64KBIT =16`

	User Rate	`UR_EINI460 =0`
		`UR_600 =1`
		`UR_1200 =2`
		`UR_2400 =3`
		`UR_3600 =4`
		`UR_4800 =5`
		`UR_7200 =6`
		`UR_8000 =7`
		`UR_9600 =8`
		`UR_8000 =7`
		`UR_9600 =8`
		`UR_14400 =9`
		`UR_16000 =10`
		`UR_19200 =11`
		`UR_32000 =12`
		`UR_48000 =14`
		`UR_56000 =15`
		`UR_64000 =16`
		`UR_134 =21`
		`UR_100 =22`
		`UR_75-1200 =23`

User Service Information (cont.)	User Rate (cont.)	`UR_1200_75 =24`
		`UR_50 =25`
		`UR_75 =26`
		`UR_110 =27`
		`UR_150 =28`
		`UR_200 =29`
		`UR_300 =30`
		`UR_12000 =31`

User-to-User Indicators	Service 2	0
User-to-User Information		

Unsupported Message Parameters

The differences in parameters supported in the G500 version of ISUP services as compared to the CCITT version are addressed in ACIF G500 Table B-1 of Annex B "General Function of Messages and Signals," Section C.3 "ISDN User Part Parameters," and Table C-4 of Annex C "Formats and Codes."

The message parameters not supported in the G500 version of ISUP services are:

- Access Delivery Information
- Call Diversion Information
- Call History Information
- Call Modification Indicators
- Call Reference
- Circuit State Indicator
- Closed User Group Interlock Code
- Connected Number
- Connection Request
- Continuity Indicators
- Echo Control Information
- Facility Indicator
- Generic Digits
- Generic Number
- Information Indicators
- Information Request Indicators
- Location Number
- Malicious Call Identification (MCID) Request Indicator
- Malicious Call Identification (MCID) Response Indicator
- Multilevel Precedence Preemption (MLPP)
- Network Specific Facilities
- Notification Indicator
- Optional Forward Call Indicators
- Origination ISC Point Code
- Pass-Along Information
- Propagation Delay Counter
- Redirection Number

- Redirection Number Restriction
- Remote Operations
- Service Activation
- Signaling Point Code
- Transit Network Selection
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information Prime
- User Teleservice Information

Differences in Protocol Procedures

The procedures for handling unrecognized messages and parameters are essentially the same in the G500 and CCITT version of ISUP services. The only differences are in the criteria used for determining whether the SINAP/SS7 system treats a message or parameter as *unrecognized* and are defined in Section D.2.10.5.3 of ACIF G500 Annex D "Signalling Procedures." In compliance with these directives, the SINAP/SS7 system uses table entries in the ISUP services database to determine which criteria to use to distinguish between recognized and unrecognized messages and parameters.

When the G500 version of ISUP services feature set is enabled, the SINAP/SS7 system activates a restricted set of messages and parameters specific to G500 (which are subsets of those supported by the CCITT version of ISUP services).

Interpretation of Spare Field Values

The SINAP/SS7 system interprets spare values in parameter fields based on the ACIF G500 Specification, Annex C.A.

The interpretation of spare values in parameter fields in the G500 and CCITT version of ISUP services are essentially the same, with the exception that the SINAP/SS7 system replaces the CCITT Calling Party Category default interpretation with an indication to fail the call in the G500 version of ISUP services.

The major difference between the two versions is the criteria used for determining whether a field value is considered *unrecognized* (or spare), since many parameter fields have different sets of allowable values, as specified in G500 version of ISUP services.

When the SINAP/SS7 system detects an unrecognized value in a field that requires default interpretation, a decoding flag in the ISUP database indicates that the default value of the field should be passed via the ISUP services application programming interface (API) to the user application, where the appropriate decoding function is performed. The SINAP/SS7 system passes default values to the user application for decoding if the parameter input contains values other than the allowed values for parameter fields shown in Table G-4.

Table G-4. Allowed Parameter Field Values

Message Parameter	Field
Backward Call Indicators	Charge Indicator Called Party's Status Indicator Called Party's Category Indicator
Cause Indicators	Location Cause Value
Nature of Connection Indicators	Satellite Indicator Continuity Check Indicators
Calling Party Number	Address Presentation Restricted Indicator
Original Called Number	Address Presentation Restricted Indicator
Redirecting Number	Address Presentation Restricted Indicator
Redirection Information	Redirecting Indicator Original Redirection Reasons Redirection Counter Redirecting Reason
User-to-User Indicators	Service 1 Service 2 Service 3

Disabled Options in G500 Version of ISUP Services

In the G500 version of ISUP services, the following protocol options that are included in the CCITT version of ISUP services, are disabled:

- Automatic transmission of an unequipped circuit identification code (UCIC) message in reply to a message received for an unequipped circuit. The SINAP/SS7 system discards messages it receives for unequipped CICs.
- Use of protocol procedures for continuity checking.
- The ability to provide segmentation of overlength messages for certain message types. The optional Backward Call Indicators parameter has no segmentation or MLPP indicator fields defined. These fields can either be ignored and set to `0` by the SINAP/ISUP system or the ISUP user application.

ISUP Services Timers

The timers used in the G500 version of ISUP services are described in the ACIF G500 Specification, Annex D Table D-4. The following sections describe the ISUP services timers that are supported and unsupported in the G500 version of ISUP services.

Supported Timers

The G500 version of ISUP services supports the subset of CCITT ISUP services timers listed in Table G-5.

Table G-5. Supported G500 ISUP Services Timers

Timer	Description
IST1	Release (REL) message sent
IST2	Suspend (SUS) user message received
IST5	Initial Release (REL) message sent
IST6	Suspend (SUS) network message received
IST7	Latest address message sent
IST9	Receipt of Address Complete (ACM) message
IST12	Blocking (BLO) message sent
IST13	Initial Blocking (BLO) message sent
IST14	Unblocking (UBL) message sent
IST15	Initial Unblocking (UBL) message sent
IST16	Reset Circuit (RSC) message (not due to IST5 expiry) sent
IST17	Initial Reset Circuit (RSC) message sent
IST18	Circuit Group Blocking (CGB) message sent
IST19	Initial Circuit Group Blocking (CGB) message sent
IST20	Circuit Group Unblocking (CGU) message sent
IST21	Initial Circuit Group Unblocking (CGU) message sent
IST22	Circuit Group Reset (GRS) message sent
IST23	Initial Circuit Group Reset (GRS) message sent

NOTE

The timer entry IST1 doubles as the value used for both T1 and T16. The timer entry IST5 doubles as the value for T5 and T17.

Unsupported Timers

The G500 version of ISUP services does not support the timers listed in Table G-6, which are supported in the in the CCITT version of ISUP services.

Table G-6. Unsupported G500 ISUP Services Timers

Timer	Description
`IST4`	Remote user unavailable MTP status event received
`IST8`	Receipt of Initial Address message (IAM) requiring continuity check
`IST27`	Receipt of continuity check failure indication
`IST33`	Information Request (INR) message sent
`IST34`	Receipt of segmented message
`IST36`	Continuity recheck timer

Appendix H
German Version of ISUP Services

The German version of ISUP services adheres to the following standards:

- Interface Specification “Signalling in ZZN7,” version 3.0.0 (approved final version 9 August 1998)
- ITU-T Recommendations Q.761 through Q.764, “ Specifications of Signaling System No.7, ISDN User Part, Volume VI” Fascicle VI.8, March 1993
- ETS 300 356-1, ISDN User Part (ISUP) version 2 for the international interface, Part 1 “Basic Services,” February 1995
- ES 201 296 V1.1.2; Integrated Services Digital Network (ISDN); Signalling System No. 7; ISDN User Part (ISUP); Signalling aspects of charging, September 1998
- EN 301 069-1 V1.2.2, Integrated Services Digital Network (ISDN); Signaling System No. 7; ISDN User Part (ISUP); Application transport mechanism; Part 1: Protocol specification [ITU-T Recommendation Q.765, modified], July 1998.

Implementing the German Version of ISUP Services

To implement the German version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `GERMANY`.

```
ISUP_FEATURE=GERMANY
```

Differences Between the German and CCITT Versions of ISUP Services

The German version of ISUP services differs from the base CCITT/ITU-T version of ISUP services, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe these differences.

For more detailed information on the differences in the German version of ISUP services, see the documents referenced at the beginning of this appendix.

Message Types

The following sections describe the messages that are and are not supported in the German version of ISUP services.

Supported Message Types

The German version of ISUP services supports a subset of the CCITT message types described in Q.763. These are listed in Table H-1. The SINAP node treats all other messages as unrecognized and discards them.

Table H-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CFN	Confusion
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress
FAA	Facility Accepted
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
IDR	Identification Request

Table H-1. Supported Message Types*(Continued)*

Message Type	Description
`IRS`	Identification Response
`NRM`	Network Resource Management
`REL`	Release
`RES`	Resume
`RLC`	Release Complete
`RSC`	Reset Circuit
`SAM`	Subsequent Address
`SGM`	Segmentation
`SUS`	Suspend
`UBA`	Unblocking Acknowledgment
`UBL`	Unblocking
`UPA`	User Part Available
`UPT`	User Part Test
`USR`	User-to-User Information

New Message Types

In addition to the supported message types listed in Table H-1, the German version of ISUP services also supports the message types listed in Table H-2. These messages are available for use whenever the ISUP environment variable (`ISUP_FEATURE`) has been set. These new message types are **not** supported by the CCITT version of ISUP services.

Table H-2. New Message Types

Message Type	Description
`APM`	Application Transport. This message is sent in either direction and contains information that allows peer-to-peer communications between application transport mechanism user applications.
`PRI`	Pre-Release Information. This message is used with the Release (REL) message to transport information when sending that information in the REL message itself would cause compatibility problems with ISUP '92 and later versions of the ISUP protocol. Note that the German version of ISUP services does **not** support segmentation of PRI messages.

See the interface specification “Signalling in ZZN7” for a complete description of the function and purpose of these new message types.

Unsupported Message Types

The German version of ISUP services does not support the message types listed in Table H-3 (which **are** supported by the CCITT version of ISUP services). The SINAP node treats these messages as unrecognized and discards them.

Table H-3. Unsupported Message Types

Message Type	Description
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CQM	Circuit Group Query
CQR	Circuit Group Query Response
CRG	Charge Information
FAC	Facility
INF	Information
INR	Information Request
LPA	Loopback Acknowledgment
OLM	Overload
PAM	Pass-Along
UCIC	Unequipped Circuit Identification Code

Supported Message Parameters

The German version of ISUP services recognizes the following message parameters that are also supported in the CCITT version of ISUP services.

- Access Delivery Information
- Access Transport
- Automatic Congestion Level
- Backward Call Indicators
- Call Diversion Information
- Call History Information

- Called Party Number
- Calling Party Number
- Calling Party's Category
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed Circuit Group Interlock Code
- Connected Number
- Connection Request
- Continuity Indicators
- Echo Control Information
- End of Optional Parameters
- Event Information
- Facility Indicator
- Forward Call Indicators
- Generic Notification
- Generic Number
- Information Request Indicators
- Location Request Indicator
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- MLPP Precedence
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Original Called Number
- Origination ISC Point Code
- Parameter Compatibility Information
- Propagation Delay Counter
- Range and Status
- Redirecting Number
- Redirection Information

- Redirection Number
- Redirection Number Restriction
- Subsequent Number
- Suspend/Resume Indicators
- Transmission Medium Requirement
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information
- User Service Information Prime
- User Teleservice Information
- User-to-User Indicators
- User-to-User Information

New Message Parameters

In addition to the supported message parameters listed in the previous section, the German version of ISUP services also supports the following message parameters.

- Application Transport
- Carrier Selection Information
- Hop Counter
- Multi-Carrier Environment
- Subscriber Priority Class

The following API structures accommodating these parameters are given.

Application Transport Parameter

```
typedef struct appTrans                     /* Application Transport */
{
ElmtHdr eh;                                 /* element header */
TknU8 appContextId;                         /* application context identifier (ACI) */
TknU8 attiRelCallInd;                       /* ATTI release call indicator */
TknU8 attiSendNotifInd;                     /* ATTI send notification indicator */
TknU8 apmSegmInd;                           /* APM segmentation Indicator */
TknU8 seqInd;                               /* sequence indicator (SI) */
```

```
TknU8 segmLocRef;                      /* segmentation local reference (SLR) */
TknStr encapAppInfo;                   /* encapsulated application information */
} SiAppTrans;
```

Hop Counter Parameter

```
typedef struct hopCounter                 /* Hop Counter*/
{
ElmtHdr eh:                               /* element header */
TknU8 hopCounter;                         /* hop counter */
} SiHopCounter;
```

Multi-Carrier Environment Parameter

```
typedef struct multCarrEnv                /* Multi Carrier Environment */
{
ElmtHdr eh:                               /* element header */
TknU8 prsntnInd;                          /* presentation indicator */
} SiMultCarrEnv;
```

Subscriber Priority Class Parameter

```
typedef struct subscrPriorClass           /* Multi Carrier Environment */
{
ElmtHdr eh:                               /* element header */
TknU8 priority;                           /* priority */
} SiSubscrPriorClass;
```

Connection Setup Event Structure

```
typedef struct siConEvnt                 /* Connection Setup Event */
{
SiNatConInd natConInd;                   /* Nature of Connection indicators */
SiFwdCallInd fwdCallInd;                 /* Forward Call indicators */
___
___
___
```

```
SiGenName genName;                              /* Generic Name */
          /* Additions for Germany */
SiAppTrans appTrans;                            /* Application Transport */
SiTranNetSel carrierSel;                        /* Carrier Selection */
SiHopCounter hopCounter;                        /* Hop Counter */
SiMultCarrEnv multCarrEnv;                      /* Multi-Carrier Environment */
SiSubscrPriorClass subscrPriorClass;            /* Subscriber Priority Class */
} SiConEvnt;
```

Connection Status Event Structure

```
typedef struct siConEvnt                       /* Connection Status Event */
{
SiSubNum subNum;                               /* Subsequent Number */
SiBckCallInd bckCallInd;                       /* Backward Call indicators */
___
___
___
SiGenName genName;                              /* Generic Name */
          /* Additions for Germany */
SiAppTrans appTrans;                            /* Application Transport */
SiOpFwdCallInd opFwdCallInd;                    /* Optional Forward Call indicators */
} SiCnSTEvnt;
```

See the interface specification “Signalling in ZZN7” for a complete description of the function and purpose of these new message parameters.

Unsupported Message Parameters

The German version of ISUP services does not support the following message parameters (which **are** supported in the CCITT version of ISUP services):

- Call Reference
- Circuit State Indicator
- Freephone Indicators
- Generic Digits
- Generic Reference
- Information Indicators
- Network-Specific Facilities

- Remote Operations
- Service Activation
- Signalling Point Code
- Transit Network Selection

Differences in Parameter Field Values

The parameters supported in the German version of ISUP services and their corresponding range of values are addressed in the interface specification "Signalling in ZZN7," section 4.3.1.3 "Formats and Codes." Note that this specification defines a different range of values than the CCITT version of ISUP services for the Called Party Number parameter. The differences for this parameter are provided in Table H-4.

Table H-4. Differences in Parameter Field Values

Message Parameter	Field	Required Value(s)
Called Party Number	Address Signal `teAddrSig`	Ported Number Prefix `1101`

NOTE

The SINAP node does not validate the address signal value. The call control application must perform this validation.

Call History & Propagation Delay Counter Message Parameters

Unlike the 1993 ITU-T recommendations for the field definitions of the Call History Information and Propagation Delay Counter message parameters, wherein the fields are not specifically defined, the German version of ISUP services explicitly defines these fields in the manner shown in Figure H-1. This is in keeping with the 1997 ITU-T recommendations for these message parameters where the fields are specifically defined and it is directed that the least significant bits (`A` through `H`) be transmitted in the second octet.

This has been interpreted in the German version of ISUP services to mean that this parameter value is always sent in two octets. For values less than 256, Octet 1 contains all zeros.

Bit Position:	8	7	6	5	4	3	2	1
Octet 1:	P	O	N	M	L	K	J	I
Octet 2:	H	G	F	E	D	C	B	A

Figure H-1. Call History and Propagation Delay Counter Field Definitions

Parameter Compatibility Information Message Parameter

Section 4.3.1.3 “Formats and Codes” of the Interface Specification “Signalling in ZZN7,” indicates that bits `G` and `F` of the Parameter Compatibility Information message parameter (which are designated as ‘Spare’ in Q.763) are used to indicate what action to take should the “pass on functionality” not be possible. These bits are labelled as “pass on not possible indicator” and are represented by the fields `passNtPoss1`, `passNtPoss2`, and `passNtPoss3` in the structure `SiParmCompInfo`. When these bits are `10` and “pass on” is not possible, the SINAP node discards the parameter.

Differences in Protocol Procedures

This section describes the differences in protocol procedures between the German version of ISUP services and the CCITT version of ISUP services currently implemented on the SINAP/SS7 system.

For more information on the differences in protocol procedures between these two versions of ISUP, see the documents described at the beginning of this appendix.

Unrecognized Messages and Message Parameters

There are no differences in handling unrecognized messages and message parameters between the German and CCITT versions of ISUP services. The SINAP node discards unrecognized messages and handles unrecognized parameter values per Table A.2, Q.763 (03/93).

Disabled Capability-Based Protocol Options

The German version of ISUP services does not support automatically sending a Unequipped Circuit Identification Code (UCIC) message as a reply to a message received for an unequipped circuit. The capability-based option bit `ISUP_OPT_NO_UCIC_REPLIES` is automatically set whenever the German version of ISUP services has been enabled.

ISUP Services Timers

The German version of ISUP services supports all the same ISUP services timers as the CCITT version of ISUP services.

API Support for New Message Types

The following sections describe the API support provided for the newly defined message types.

Application Transport (APM) message

The SINAP node sends the new Application Transport (APM) message type after it receives an IAM. The APM is unexpected in all other instances. In addition, it can be sent in both directions.

To accommodate the APM, the following event type has been defined for the Connection Status primitives:

`APPTRANS (=26)`

Since this message can be sent and received, the connection status events `EVTSITCNSTREQ` and `EVTSITCNSTIND` allow the `APPTRANS` event type to be passed to the `isup_put_msg()` and `isup_get_msg()` functions to enable APM to be sent and received.

Pre-Release Information (PRI) Message

The SINAP node sends the Pre-Release Information (PRI) message after it receives/sends an IAM and before it sends/receives a REL message.

To accommodate the PRI, the German version of ISUP services supports the following event type for the Connection Status primitives:

`PRERELINFO(=27)`

API Support for New Message Parameters

Of the five new message parameters defined in the German version of ISUP services, all are optional for the Initial Address (IAM) message. Therefore, since the IAM is sent and received via the Application Programming Interface (API) through the Connection Setup primitive, elements for each of these message parameters are contained in the `SiConEvnt` structure.

The Application Transport message parameter is optional in the following messages:

- Address Complete (ACM)
- Answer (ANM)
- Call Progress (CPG)
- Connect (CON)
- Application Transport (APM)
- Pre-Release Information (PRI)

Because the ACM, CPG, APM, and PRI messages are sent and received via the API through the Connection Status primitive, elements for this message parameter are contained in the `SiCnStEvnt` structure. Since the elements for the Application Transport message parameter are added in the `SiConEvnt` structure for the IAM, the ANM and CON messages can make use of the updated `SiConEvnt` structure for the elements of the Application Transport message parameter.

Sample Program Files

Two sample programs (`grm_isup_send.c` and `grm_isup_recv.c`) are provided to assist in testing the message types and message parameters that are specific to the German version of ISUP services. These sample programs are modified versions of the base isup_send.c and isup_recv.c sample programs provided with the SINAP/SS7 software.

The message flow between the German ISUP sample programs is shown in Figure H-2.

An additional sample program (grm_isupMtpSend.c) has been provided to facilitate testing of the German version of ISUP services. This new sample program is used to send ISUP messages (bypassing SINAP ISUP) across to the remote application (running on SINAP ISUP). The remote application can be either of the previous two sample programs, depending on the test to be conducted.

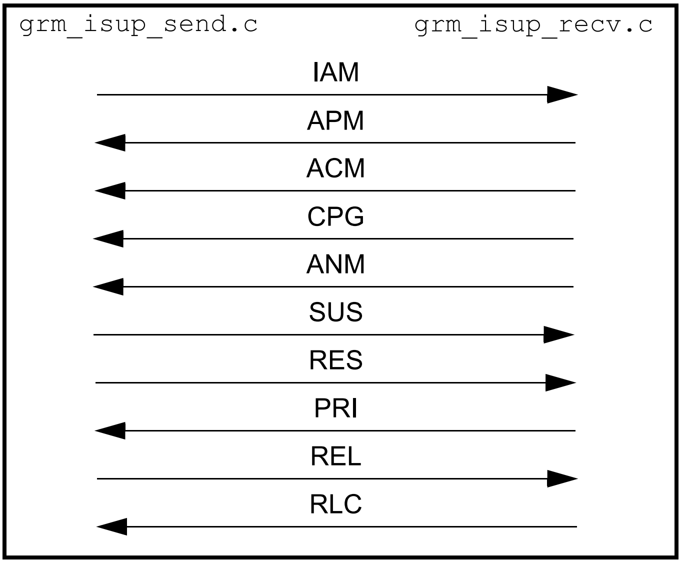

Figure H-2. Message Flow Between `grm_isup_send.c` and `grm_isup_recv.c`

Appendix I
Italian Version of ISUP Services

The Italian version of ISUP services adheres to the following standards:

- Technical Specification N.763, “Technical Characteristics of the Interconnection between Telecommunications Networks”, Release 1, April 1998.
- ETS 300 121, “Integrated Services Digital Network (ISDN); Application of the ISDN User Part (ISUP) of CCITT Signaling System No. 7 for International ISDN Interconnection”, December 1992
- ITU-T Recommendation Q.767, “Application of the ISDN User Part of CCITT No. 7 for International ISDN Connections”, 1991
- ITU-T Recommendations Q.761 through Q.764, “Specifications of Signaling System No. 7. ISDN USer Part, Volume VI, Fascicle VI.8”, November 1988

NOTE

Although the Introduction to N.763 states that Interfaces A through G are considered, the role of the SINAP/SS7 system as a Service Control Point (SCP) in the SS7 network requires that only Part1A (the interface between fixed networks) of Interface A be supported in this Italian version of ISUP services.

Implementing the Italian Version of ISUP Services

To implement the Italian version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `ITALY`.

```
ISUP_FEATURE=ITALY
```

NOTE

The CCITT network variant must be configured on the SINAP node in order to activate the Italian version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).

Differences Between the Italian and CCITT Versions of ISUP Services

The Italian version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences in the message types, message parameters, and parameter field values between the two versions.

For more detailed information on the differences in the Italian version of ISUP services, see the documents referenced at the beginning of this appendix.

Message Types

The following sections describe the messages that are and are not supported in the Italian version of ISUP services.

Supported Message Types

The Italian version of ISUP services supports the message types listed in Table I-1

Table I-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress
FOT	Forward Transfer

Table I-1. Supported Message Types *(Continued)*

Message Type	Description
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The Italian version of ISUP services does not support the message types in Table I-2. The SINAP node treats these messages as unrecognized and ignores them.

Table I-2. Unsupported Message Types

Message Type	Description
CFN	Confusion
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CQM	Circuit Group Query
COM *	Call Offering
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request

Table I-2. Unsupported Message Types *(Continued)*

Message Type	Description
FRJ	Facility Rejected
IDR *	Identification Request
INF	Information
INR	Information Request
IRS *	Identification Response
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass-Along
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

NOTE

Although defined by paragraphs 3.1.5.1 and 3.1.5.4 of the N.763 specification, the message types indicated by an asterisk (*) are nonetheless not supported in this version of Italian ISUP services.

Section 3.1.5 of the N.763 specification states that the information provided in these paragraphs is not intended to be normative, but is only provided for the purpose of facilitating interconnected operators in case some of these exceptions might be necessary to support additional services on the interface.

Message Parameters

This section describes the messages parameters that are and are not supported in the Italian version of ISUP services.

Supported Message Parameters

The Italian version of ISUP recognizes the following message parameters.

For more detailed information on these parameters, see the documents referenced at the beginning of this appendix.

- Access Transport
- Automatic Congestion Level
- Backward Call Indicators
- Called Party Number *
- Calling Party Number *
- Calling Party's Category
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code
- Connected Number *
- Continuity Indicators
- End of Optional Parameters
- Event Information
- Forward Call Indicators
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Range and Status
- Redirection Information
- Subsequent Number *
- Suspend/Resume Indicators
- Transmission Medium Requirement
- User Service Information
- User-to-User Indicators
- User-to-User Information

NOTE

In addition to the values specified in Q.767 for the Address Signal field of the message parameters marked with an asterisk

(*), N.763 allows the use of values `1011` (code 11) and `1100` (code 12) on the interface subject to an agreement between the service providers.

Unsupported Message Parameters

The Italian version of ISUP services does not recognize the following message parameters.

- Access Delivery Information *
- Call Diversion Information
- Call History Information
- Call Modification Indicators
- Call Reference
- Circuit State Indicator
- Connection Request
- Echo Control Information
- Facility Indicator
- Generic Digits
- Generic Notification
- Generic Number
- Incoming Trunk Identity Code *
- Information Indicators
- Information Request Indicators
- Location Number
- MCID Request Indicator *
- MCID Response Indicator *
- Message Compatibility Information *
- MLPP Precedence
- Network Specific Facilities
- Original Called Number
- Origination ISC Point Code
- Parameter Compatibility Information *
- Propagation Delay Counter
- Redirecting Number
- Redirection Number

- Redirection Number Restriction
- Remote Operations
- Service Activation
- Signalling Point Code *
- Transmission Medium Requirement Prime
- Transmission Medium Used
- Transit Network Selection
- User Service Information Prime
- User Teleservice Information

NOTE

Although defined by paragraphs 3.1.5.1 through 3.1.5.3 of the N.763 specification, the message parameters indicated by an asterisk (*) are nonetheless not supported in this version of Italian ISUP services.

Section 3.1.5 of the N.763 specification states that the information provided in these paragraphs is not intended to be normative, but is only provided for the purpose of facilitating interconnected operators in case some of these exceptions might be necessary to support additional services on the interface.

Differences in Parameter Field Values

The Italian version of ISUP services, which is primarily based upon Technical Specification N.763, supports essentially the same parameter field values as those defined in the Q.767 (1991) specification for ISUP services. The differences between the two specifications are described in Table I-3.

Table I-3. Differences in Parameter Field Values Between N.763 and Q.767

Parameter	Field	Value Differences
Called Party Number	Nature of Address Indicator	`0001` Subscriber number
Calling Party Number	Nature of Address Indicator	`0011` National (significant) number

Table I-3. Differences in Parameter Field Values Between N.763 and Q.767*(Continued)*

Parameter	Field	Value Differences
	Calling Party Number Incomplete Indicator (NI)	`Incomplete (=1)`
Calling Party's Category		`0000` Unknown at this time `1001` National operator
Cause Indicators	Location	`0001` Private network serving the local user `0010` Public network serving the local user
Connected Number	Nature of Address Indicator	`0011` National (significant) number

Differences in Protocol Procedures

This section describes the differences in protocol procedures between the Italian (N.763) and ITU-T/CCITT (1991 Q.767) versions of ISUP services.

Unrecognized Messages and Parameters

The Italian N.763 version of ISUP services does not specify any exception from the manner in which the ITU-T/CCITT (1991 Q.767) version of ISUP services handles unrecognized message types and message parameters. However, there are differences in the manner in which unrecognized message types and message parameters are handled between the ITU-T/CCITT (1991 Q.767) version of ISUP services and the ITU-T/CCITT (1993 Q.764) version of ISUP services that the SINAP/SS7 system currently supports.

The procedures for handling unrecognized message types and message parameters in ITU-T/CCITT (1991 Q.767) are much simpler than those found in ITU-T/CCITT (1993 Q.764); Q.767 does not support either the Message Compatibility Information parameter or the Parameter Compatibility Information parameter, whereas Q.764 does. Further, Q.767 does not support use of the Confusion (CFN) message. As a result, Q.767 discards and ignores unrecognized message types and message parameters.

ISUP Services Timers

The Italian version of ISUP services supports all the same ISUP services timers as the CCITT version of ISUP services.

Appendix J
ITU97 Version of ISUP Services

The 1997 version of ITU-T ISUP services (hereafter referred to as the ITU97 version of ISUP services) adheres to the following standards:

- ITU-T Recommendations Q.761 through Q.764, "Specifications of Signaling System No. 7, ISDN User Part", September 1997
- ITU-T Recommendation Q.763, "Specifications of Signaling System No. 7, Addendum 1, ISDN User Part Formats and Codes", May 1998
- ITU-T Recommendation Q.765, "Specifications of Signaling System No. 7, Application Transport Mechanism", May 1998.

Implementing the ITU97 Version of ISUP Services

To implement the ITU97 version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `ITU97`.

```
ISUP_FEATURE=ITU97
```

NOTE

The CCITT network variant must be configured on the SINAP node in order to activate the ITU97 version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).

ITU97 ISUP Services

The ITU97 version of ISUP services are an extension of the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which are based on the 1993 edition of ITU-T Recommendations for ISUP services.

The following sections describe the message types, message parameters, and parameter field values supported in the ITU97 version of ISUP services.

Message Types

This section describes the messages supported in the ITU97 version of ISUP services.

Supported Message Types

The ISU97 version of ISUP services supports all of the CCITT message types described in Q.763 (1993). These are listed in Table J-1. The SINAP node treats all other messages as unrecognized and discards them.

Table J-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CFN	Confusion
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
GRA	Circuit Group Reset Acknowledgment

Table J-1. Supported Message Types *(Continued)*

Message Type	Description
GRS	Circuit Group Reset
IAM	Initial Address
IDR	Identification Request
INF	Information
INR	Information Request
IRS	Identification Response
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass-Along
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SGM	Segmentation
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

New Message Types

In addition to the supported message types listed in Table J-1, three new message types are defined in the ITU97 version of ISUP services and are available for use when the ISUP services environment variable has been properly set. These new message types are listed in Table J-2 and are **not** supported by the CCITT (1993) version of ISUP services.

Table J-2. New Message Types

Message Type	Description
APM	Application Transport
LOP	Loop Prevention
PRI	Pre-Release Information

Message Parameters

This section describes the messages parameters supported in the ITU97 version of ISUP services.

Supported Message Parameters

The ITU97 version of ISUP services supports all but one of the message parameters described in Q.763 (1993). The exception is the Generic Reference parameter, which is marked as ‘Reserved’ in the ITU97 version of ISUP services.

In addition, twenty new optional message parameters are defined for use in the ITU97 version of ISUP services. The complete list of message parameters supported by the ITU97 version of ISUP services is provided below.

- Access Delivery Information
- Access Transport
- Application Transport*
- Automatic Congestion Level
- Backward Call Indicators
- Backward Global Virtual Network Services (GVNS)*
- Call Diversion Information
- Call Diversion Treatment Indicators*
- Call History Information
- Call Offering Treatment Indicators*
- Call Reference
- Call Transfer Number*
- Call Transfer Reference*
- Called IN Number*

- Called Party Number
- Calling Party Number
- Calling Party’s Category
- Cause Indicators
- Call Completion Service Set-up (CCSS)*
- Circuit Assignment Map*
- Circuit Group Supervision Message Type Indicator
- Circuit State Indicator
- Closed User Group Interlock Code
- Collect Call Request*
- Conference Treatment Indicators*
- Connected Number
- Connection Request
- Continuity Indicators
- Correlation Id*
- Display Information*
- Echo Control Information
- Event Information
- Facility Indicator
- Forward Call Indicators
- Forward Global Virtual Network Service (GVNS)*
- Generic Digits
- Generic Notification
- Generic Number
- Hop Counter*
- Information Indicators
- Information Request Indicators
- Location Number
- Loop Prevention Indicators*
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information

- MLPP Precedence
- Nature of Connection Indicators
- Network Management Controls*
- Network Specific Facilities
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Original Called Number
- Original ISC Point Code
- Parameter Compatibility Information
- Propagation Delay Counter
- Range and Status
- Redirect Capability
- Redirect Counter
- Redirecting Number
- Redirection Information
- Redirection Number
- Redirection Number Restriction
- Remote Operations
- Service Control Function (SCF) Id*
- Service Activation
- Signalling Point Code
- Subsequent Number
- Suspend/Resume Indicators
- Transit Network Selection
- Transmission Medium Requirement
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Interaction Dialog (UID) Action Indicators*
- User Interaction Dialog (UID) Capability Indicators*
- User Service Information
- User Service Information Prime
- User Teleservice Information

- User-to-User Indicators
- User-to-User Information

NOTE

Parameters marked with an asterisk (*) are the new message parameters implemented for the ITU97 version of ISUP services.

Unsupported Message Parameters

The ITU97 version of ISUP services does not recognize the following message parameter which is defined in Q.763 (1993). This parameter is marked as 'Reserved' in the ITU97 version of ISUP services.

- Generic Reference

Allowed Parameter Values

The following sections contain the values for the message parameters supported by the ITU97 version of ISUP services.

Differences with CCITT (1993)

Table J-3 contains the values for parameters supported by the ITU97 version of ISUP services, but which differ from those described in the 1993 CCITT version of ISUP services.

Table J-3. Parameters Values Differing from CCITT

Parameter	Field	Allowable Value(s)

Called Party Number	Nature of Address Indicator	`UNKNOWN (= 0)`
		`SUBSNUM (= 1)`
		`NATNUM (= 3)`
		`INTNATNUM (= 4)`
		`NETSPECNUM (= 5)`

Cause Indicators	Cause Value	`CCUNALLOC (= 1)`
		`CCNORTTOTSFNET (= 2)`
		`CCNORTTODEST (= 3)`
		`CCSENDSPCLTONE (= 4)`
		`CCMISDIALDTRNK (= 5)`
		`CCREDIRNEWDEST (= 23)`
		`CCEXCHRTERR (= 25)`

Table J-3. Parameters Values Differing from CCITT*(Continued)*

Parameter	Field	Allowable Value(s)
	Cause Value (cont.)	CCCALLCLR (= 16) CCUSRBSY (= 17) CCNOUSRRSP (= 18) CCNOANSWR (= 20) CCCALLRJT (= 21) CCNMBRCHG (= 22) CCDESTOUTORD (= 27) CCADDRINCOMP (= 28) CCFACREJ (= 29) CCNORMUNSPEC (= 31) CCNOCIRCUIT (= 34) CCNETAOL (= 38) CCTMPFAIL (= 41) CCSWTCHCONG (= 42) CCACCINFDISC (= 43) CCREQUNAVAIL (= 44) CCRESCUNAVAIL (= 47) CCFACNOTSUB (= 50) CCOGBARRDCUG (= 53) CCINCBARRDCUG (= 55) CCNOTAUTHBCAP (= 57) CCBCAPUNAVAIL (= 58) CCINCOGACCSUBCLS (= 62) CCSERVUNAVAIL (= 63) CCBCAPNOTIMP (= 65) CCFACNOTIMP (= 69) CCRESTDIG (= 70) CCSERVNOTIMP (= 79) CCCUNOTMEMBR (= 87) CCINCOMPDEST (= 88) CCNOTEXISTCUG (= 90) CCINVTRNSTNET (= 91) CCINVMSG (= 95) CCNOMSGTYP (= 97) CCNOPARAMDISC (= 99) CCTMRRECOV (=102) CCNOPARAMPASS (=103) CCNOPARAMDISCMSG (=110) CCPROTERR (=111) CCINTRWRK (=127)
Echo Control Information	Incoming Echo Control Information	ECHCDEV_NOINFOINCL (= 0) ECHCDEV_NOTINCL (= 1) ECHCDEV_INCL (= 2) ECHCDEV_AVAIL (= 3)

Table J-3. Parameters Values Differing from CCITT*(Continued)*

Parameter	Field	Allowable Value(s)

Echo Control Information (cont.)	Outgoing Echo Control Information	`ECHCDEV_NOINFOINCL (= 0)`
		`ECHCDEV_NOTINCL (= 1)`
		`ECHCDEV_INCL (= 2)`
		`ECHCDEV_AVAIL (= 3)`

Generic Number	Number Qualifier Indicator	`NQ_ADDCDMNB (= 1)`
		`NQ_ADDCONMNB (= 5)`
		`NQ_ADDCGNMB (= 6)`

Redirecting Number	Nature of Address Indicator	`ADDR_NOTPRSNT (= 0)`
		`SUBSNUM (= 1)`
		`NATNUM (= 3)`
		`INTNATNUM (= 4)`

Transmission Medium Requirement		`TMR_SPEECH (= 0)`
		`TMR_64KBITS (= 2)`
		`TMR_31KHZ (= 3)`
		`TMR_ALTSPEECH (= 4)`
		`TMR_ALT64KBITS (= 5)`
		`TMR_64KBSPREF (= 6)`
		`TMR_2X64KBITS (= 7)`
		`TMR_384KBITS (= 8)`
		`TMR_1536KBITS (= 9)`
		`TMR_1920KBITS (= 10)`
		`TMR_3X64KBITS (= 16)`
		`TMR_4X64KBITS (= 17)`
		`TMR_5X64KBITS (= 18)`
		`TMR_7X64KBITS (= 20)`
		`TMR_8X64KBITS (= 21)`
		`TMR_9X64KBITS (= 22)`
		`TMR_10X64KBITS (= 23)`
		`TMR_11X64KBITS (= 24)`
		`TMR_12X64KBITS (= 25)`
		`TMR_13X64KBITS (= 26)`
		`TMR_14X64KBITS (= 27)`
		`TMR_15X64KBITS (= 28)`
		`TMR_16X64KBITS (= 29)`
		`TMR_17X64KBITS (= 30)`
		`TMR_18X64KBITS (= 31)`
		`TMR_19X64KBITS (= 32)`
		`TMR_20X64KBITS (= 33)`
		`TMR_21X64KBITS (= 34)`
		`TMR_22X64KBITS (= 35)`
		`TMR_23X64KBITS (= 36)`
		`TMR_25X64KBITS (= 38)`

Table J-3. Parameters Values Differing from CCITT*(Continued)*

Parameter	Field	Allowable Value(s)
Transmission Medium Requirement (cont.)		TMR_26X64KBITS (= 39) TMR_27X64KBITS (= 40) TMR_28X64KBITS (= 41) TMR_29X64KBITS (= 42)

New ITU97 Parameter Values

Table J-4 contains the values for the new parameters used in the ITU97 version of ISUP services. These parameters are described in Q.763 (1997).

Table J-4. New ITU97 Parameter Values

Parameter	Field	Allowable Value(s)
Application Transport	Application Context Identifier	ACI_UCEHASE (= 0) ACI_PSS1ASE (= 1) ACI_CHARGINGASE (= 3)
	ATII Release Call Indicator	ATIIRCI_DONOTREL (= 0) ATIIRCI_RELCALL (= 1)
	ATII Send Notification Indicator	ATIISNI_DONOTSEND (= 0) ATIIRCI_SENDNOTIF (= 1)
	Sequence Indicator	SQI_SUBSEQTOFIRSTSEG (= 0) SQI_NEWSEQUENCE (= 1)
Backward Global Virtual Network Services (GVNS)	Terminating Access Indicator	BGVNS_NOINFO (= 0) BGVNS_DEDTERMACC (= 1) BGVNS_SWTTERMACC (= 2)
Call Diversion Treatment Indicators	Call to be Diverted Indicator	CALDIV_NOIND (= 0) CALDIV_ALLOW (= 1) CALDIV_NOTALLOW (= 2)
Call Offering Treatment Indicators	Call to be Offered Indicator	CALOFF_NOIND (= 0) CALOFF_NOTALLOW (= 1) CALOFF_ALLOW (= 2)
Call Transfer Number	Nature of Address Indicator	SUBSNUM (= 1) UNKNOWN (= 2) NATNUM (= 3) INTNATNUM (= 4)
	Numbering Plan Indicator	ISDNNUM (= 1) DATANUM (= 3) TELEXNUM (= 4) PRIVATENUMPLAN (= 5)

Table J-4. New ITU97 Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
Call Transfer Number (cont.)	Address Presentation Restricted Indicator	PRESALLOW (= 0) PRESREST (= 1)
	Screening Indicator	USRPROVNOTVER (= 0) USRPROV (= 1) USRPROVVERFASIL (= 2) NETPROV (= 3)
Called IN Number	Nature of Address Indicator	SUBSNUM (= 1) UNKNOWN (= 2) NATNUM (= 3) INTNATNUM (= 4)
	Numbering Plan Indicator	NP_UNK (= 0) ISDNNUM (= 1) DATANUM (= 3) TELEXNUM (= 4)
	Address Presentation Restricted Indicator	PRESALLOW (= 0) PRESREST (= 1) ADDRNOAVAIL (= 2)
Call Completion Service Set-up (CCSS)	CCSS Call Indicator	CCSS_NOIND (= 0) CCSS_CALL (= 1)
Circuit Assignment Map	Map Type	FMT_1544BITS (= 1) FMT_2048BITS (= 2)
Collect Call Request	Collect Call Request Indicator	COLCAL_NOIND (= 0) COLCAL_CALL (= 0)
Conference Treatment Indicators	Conference Acceptance Indicator	CONF_NOIND (= 0) CONF_ACCEPT (= 1) CONF_REJECT (= 2)
Forward Global Virtual Network Service (GVNS)	Nature of Address Indicator	SUBSNUM (= 1) UNKNOWN (= 2) NATNUM (= 3) INTNATNUM (= 4) NETSPECNUM (= 5)
Loop Prevention Indicators	Type	LPPRV_TYPE_REQ (= 0) LPPRV_TYPE_RESP (= 1)
	Response Indicator	LPRSP_INSUFFINFO (= 0) LPRSP_NOLOOP (= 1) LPRSP_SIMTRNSFR (= 2)

Table J-4. New ITU97 Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
Network Management Controls	Temporary Alternative Routing Indicator	`TAR_NOIND (= 0)` `TAR_CTRLCALL (= 1)`
User Interaction Dialog (UID) Action Indicators	Through-Connection Instruction Indicator	`THRUCON_NOIND (= 0)` `THRUCON_BOTHDIR (= 1)`
	T9 Timer Instruction Indicator	`T9_NOIND (= 0)` `T9_STOP (= 1)`
User Interaction Dialog (UID) Capability Indicators	Through-Connection Indicator	`THRUCON_NOIND (= 0)` `THRUCON_BOTHDIR (= 1)`
	T9 Timer Instruction Indicator	`T9_NOIND (= 0)` `T9_STOP (= 1)`

Protocol Procedures

The ITU97 version of ISUP services supports the same set of protocol procedures as the 1993 CCITT version of ISUP services currently implemented in the SINAP/SS7 platform.

ISUP Services Timers

The ITU97 version of ISUP services supports all the same ISUP services timers as the 1993 CCITT version of ISUP services with the following exception.

In the 1993 CCITT version of ISUP services, if a request for a re-connection or a Resume (RES) message is not received within the time allocated by timers T6 or T38, then the exchange where either timer has been started will initiate the release procedure on both ends. A `Cause` value of `0x102` will be returned in the Release (REL) message. However, in the ITU97 version of ISUP services, a `Cause` value of `0x16` (normal call clearing) is returned in the REL message on the expiry of timer T6 and a `Cause` value of `0x102` (recovery on timer expiry) is returned in the REL message on the expiry of timer T38.

API Support for Sending and Receiving New Messages

The ITU97 version of ISUP services defines three new message types (listed in Table J-2). Accordingly, three new event types for connection status primitives associated with sending and receiving these messages are defined. These are listed in Table J-5.

Table J-5. New Event Types

Event Type	Description
`APPTTRANS`	Application Transport
`LOOPPRVNT`	Loop Prevention

Table J-5. New Event Types *(Continued)*

Event Type	Description
PRERELINFO	Pre-Release Information

These event types apply to the connection status primitives for sending and receiving messages and use the `SiCnStEvnt` structure to convey the parameters of the associated ISUP protocol messages.

Application Transport (APM) Message

A SINAP node can send an Application Transport (APM) message after receiving an Initial Address (IAM) message. The APM message is unexpected in all other instances.

The APM message is defined with the following optional parameters:

- Message Compatibility Information
- Parameter Compatibility Information
- Application Transport

These parameters are represented in the `SiCnStEvnt` structure as:

```
SiMsgCompInfo     msgCom;
SiParmCompInfo    parmCom;
SiAppTrans        appTrans;
```

Pre-Release Information (PRI) Message

A SINAP node can send an Pre-Release Information (PRI) message after receiving or sending an Initial Address (IAM) message and before receiving or sending a Release (REL) message.

The PRI message is defined with the following optional parameters:

- Optional Forward Call Indicators
- Optional Backward Call Indicators
- Message Compatibility Information
- Parameter Compatibility Information
- Application Transport

These parameters are represented in the `SiCnStEvnt` structure as:

```
SiOptBckCalInd     SiOptBckCalInd;
SiOptFwdCalInd     SiOptFwdCalInd;
SiMsgCompInfo      msgCom;
SiParmCompInfo     parmCom;
```

```
SiAppTrans          appTrans;
```

Loop Prevention (LOP) Message

The Loop Prevention (LOP) message can be sent after an Initial Address (IAM) message and before a Release (REL) message.

The LOP message is defined with the following optional parameters:

- Call Transfer Reference
- Loop Prevention Indicators
- Message Compatibility Information
- Parameter Compatibility Information

These parameters are represented in the `SiCnStEvnt` structure as:

```
SiCallTRef          callTranRef;
SiLoopPrvnt         loopPrvntInd;
SiMsgCompInfo       msgCom;
SiParmCompInfo      parmCom;
```

Sample Program Files

Two sample programs (`itu97_isup_send.c` and `itu97_isup_recv.c`) are provided to assist in testing the message types and message parameters that are specific to the ITU97 version of ISUP services. These sample programs are modified versions of the base `isup_send.c` and `isup_recv.c` sample programs provided with the SINAP/SS7 software.

Appendix K
Netherlands Version of ISUP Services

The Netherlands version of ISUP services adheres to the following standards:

- ETS 300 121, "Integrated Services Digital Network (ISDN); Application of the ISDN User Part (ISUP) of CCITT Signaling System No. 7 for International ISDN Interconnection", December 1992
- ETS 300 303, "Integrated Services Digital Network (ISDN); ISDN - Global System for Mobile Communications (GSM) Public Land Mobile Network (PLMN) Signalling Interface", July 1994
- ETS 300 303, "Integrated Services Digital Network (ISDN); ISDN - Global System for Mobile Communications (GSM) Public Land Mobile Network (PLMN) Signalling Interface", January 1996
- ITU-T Recommendation Q.767, "Application of the ISDN User Part of CCITT No. 7 for International ISDN Connections", 1991.

Implementing the Netherlands Version of ISUP Services

To implement the Netherlands version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `NETHERLANDS`.

```
ISUP_FEATURE=NETHERLANDS
```

NOTES

1. The CCITT network variant must be configured on the SINAP node in order to activate the Netherlands version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).
2. The Netherlands version of ISUP services requires that the Network indicator in the MTP routing table be set to 11

(reserved for National use). Use the MML command `CREATE-OSP` to set the `NETWORK` field to `11`.

Differences Between the Netherlands and CCITT Versions of ISUP Services

The Netherlands version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which is based on the 1991 edition of ITU-T Recommendation Q.767 for ISUP services. The following sections describe the differences in the message types, message parameters, and parameter field values between the two versions.

For more detailed information on the differences in the Netherlands version of ISUP services, see the documents referenced at the beginning of this appendix.

Message Types

The following sections describe the messages that are and are not supported in the Netherlands version of ISUP services.

Supported Message Types

The Netherlands version of ISUP services supports the message types listed in Table K-1.

Table K-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress

Table K-1. Supported Message Types *(Continued)*

Message Type	Description
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The Netherlands version of ISUP services does not support the message types in Table K-2. The SINAP node treats these messages as unrecognized and ignores them.

Table K-2. Unsupported Message Types

Message Type	Description
CFN	Confusion
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FRJ	Facility Rejected

Table K-2. Unsupported Message Types *(Continued)*

Message Type	Description
FOT	Forward Transfer
IDR	Identification Request
INF	Information
INR	Information Request
IRS	Identification Response
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass-Along
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

Message Parameters

This section describes the messages parameters that are and are not supported in the Netherlands version of ISUP services.

Supported Message Parameters

The Netherlands version of ISUP recognizes the following message parameters.

For more detailed information on these parameters, see the documents referenced at the beginning of this appendix.

- Access Transport
- Automatic Congestion Level
- Backward Call Indicators
- Called Party Number
- Calling Party Number
- Calling Party's Category

- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code
- Connected Number
- Continuity Indicators
- End of Optional Parameters
- Event Information
- Forward Call Indicators
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Range and Status
- Redirection Information*
- Subsequent Number
- Suspend/Resume Indicators
- Transmission Medium Requirement
- User Service Information
- User-to-User Indicators
- User-to-User Information

NOTE

The parameter indicated by an asterisk (*) is applicable to the Initial Address (IAM) message only.

Unsupported Message Parameters

The Netherlands version of ISUP services does not recognize the following list of message parameters:

- Access Delivery Information
- Call Diversion Information
- Call History Information
- Call Modification Indicators
- Call Reference
- Circuit State Indicator
- Connection Request

- Echo Control Information
- Facility Indicator
- Generic Digits
- Generic Notification
- Generic Number
- Information Indicators
- Information Request Indicators
- Location Number
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- MLPP Precedence
- Network Specific Facilities
- Original Called Number
- Origination ISC Point Code
- Parameter Compatibility Information
- Propagation Delay Counter
- Redirecting Number
- Redirection Number
- Redirection Number Restriction
- Remote Operations
- Service Activation
- Signalling Point Code
- Transmission Medium Requirement Prime
- Transmission Medium Used
- Transit Network Selection
- User Service Information Prime
- User Teleservice Information

Allowed Parameter Values

The parameter values contained in Table K-3 are specified for the Netherlands version of ISUP services and are described in the 1991 edition of ITU-T Recommendation Q.767 for ISUP services.

Table K-3. Allowed Parameter Values

Parameter	Field	Allowable Value(s)
Calling Party Number	Nature of Address Indicator	`000 0011` National number `000 0100` International number
Cause Indicators	Cause Value	`000 0001` Unallocated (unassigned) number `000 0011` No route to destination `000 0100` Send special information tone `001 0000` Normal call clearing `001 0001` User busy `001 0010` No user responding `001 0011` No answer from user (user alerted) `001 0100` Absent subscriber `001 0101` Call rejected `001 0110` Number changed `001 1011` Destination out of order `001 1100` Address incomplete `001 1101` Facility rejected `001 1111` Normal unspecified `010 0010` No circuit available `010 0110` Network out of order `010 1001` Temporary failure `010 1010` Switching equipment congestion `010 1100` Requested channel not available `010 1111` Resource unavailable - unspecified `011 0111` Incoming calls barred within CUG `011 1001` Bearer capability not authorized `011 1010` Bearer capability not presently available `011 1111` Service/option not available - unspecified `100 0001` Bearer capability not implemented `100 1111` Service or option not implemented - unspecified `101 0111` User not member of CUG `101 1000` Incompatible destination `101 1111` Invalid message - unspecified `110 0110` Recovery on timer expiry `110 1111` Protocol error - unspecified `111 1111` Interworking unspecified

Table K-3. Allowed Parameter Values *(Continued)*

Parameter	Field	Allowable Value(s)
Connected Number	Nature of Address Indicator	`000 0000` Address not Present `000 0011` National (significant) number `000 0100` International number

The parameter values contained in Table K-4 are specified for the Netherlands version of ISUP services, but are described in the 1988 edition of ITU-T Recommendation Q.763 for ISUP services.

Table K-4. Allowed Parameter Values

Parameter	Field	Allowable Value(s)
Redirection Information	Redirecting Indicator	Bits: `CBA` `011` Call diverted (Default) `100` Call diverted; all redirection information presentation restricted.
	Original Redirection Reasons	Bits: `HGFE` `0000` Unknown / not available
	Redirecting Reason	Bits: `POMN` `0000` Unknown / not available (Default) `0001` User busy `0010` No reply `0011` Unconditional `0100` Deflection during alerting `0101` Deflection immediate response `0110` Mobile subscriber not reachable

Differences in Protocol Procedures

This section describes the differences in protocol procedures between the Netherlands and CCITT versions of ISUP services currently implemented on the SINAP/SS7 system.

Unrecognized Messages and Parameters

The procedures for handling unrecognized messages and message parameters are essentially the same in both versions of ISUP services. The only differences are in the criteria used to determine whether a message or message parameter is considered *unrecognized*.

Interpretation of Spare Field Values

The method of interpreting spare values in parameter fields is also essentially the same in the Chinese and CCITT versions of ISUP services with the following exception.

In the CCITT version of ISUP services, the Redirecting Indicator field of the Redirection Information message parameter defaults to `RI_CALLFWDALLRIPRESRES` (Bits `CBA`: `100`). In the Netherlands version of ISUP services, the default value for the Redirecting Indicator field of the Redirection Information message parameter is `RI_CALLFWD` (Bits `CBA`: `011`).

Disabled Options

There are no disabled options in the Netherlands version of ISUP services.

ISUP Services Timers

The Netherlands version of ISUP services supports all the same ISUP services timers as the CCITT version of ISUP services.

Appendix L
NTT-IC Version of ISUP Services

The NTT-InterConnect (NTT-IC) version of ISUP services adheres to the following industry specifications:

- ITU-T “White Book” Recommendations Q.761 through Q.764, “Specifications of Signaling System No.7, ISDN User Part, Volume VI” Fascicle VI.8, March 1993
- “Specifications for Multi-Carrier InterConnection Interface” (Technical Conditions for Connection) by Nippon Telegraph and Telephone Corp., InterConnection Promotion Department, December 22, 1998
- NTT-Q763 and NTT-Q763-1, Technical Requirements Appendix Table 4 and Table 4.1 (of NTT-IC), by Nippon Telegraph and Telephone
- Telecommunications Technology Committee (TTC) Standard JT-Q763, “Formats and Codes” (for ISUP), Version 14.1, September 8, 1999

Implementing the NTT-IC version of ISUP Services

To implement the NTT-IC version of ISUP services, un-comment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `NTT_IC`.

```
ISUP_FEATURE=NTT_IC
```

NOTE

The NTT network variant must be configured on the SINAP node in order to activate the NTT-IC version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).

Differences Between the NTT-IC, NTT, and CCITT Versions of

ISUP Services

The NTT-IC version of ISUP services differs from both the NTT version of ISUP services that the SINAP/SS7 software currently supports and the base CCITT/ITU-T version of ISUP services, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences between these versions.

For more detailed information on the differences in the NTT-IC version of ISUP services, see the documents referenced at the beginning of this chapter.

Message Types

The following sections describe the messages which are and are not supported in the NTT-IC version of ISUP services. Additionally, any similarities and differences between the NTT-IC version of ISUP services and the NTT and CCITT/ITU versions of ISUP services will be described.

Supported Message Types

The NTT-IC version of ISUP services supports the same set of messages as the NTT version of ISUP services. This includes the Charge Information (CHG) message, which is not recognized by the CCITT version of ISUP services.

The messages supported by the NTT-IC version of ISUP services are listed in Table L-1. The SINAP node treats all other messages as unrecognized and discards them.

Table L-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CHG	Charge Information
CPG	Call Progress
CQM	Circuit Group Query
CQR	Circuit Group Query Response
GRS	Circuit Group Reset
GRA	Circuit Group Reset Acknowledgment
IAM	Initial Address
REL	Release

Table L-1. Supported Message Types*(Continued)*

Message Type	Description
RLC	Release Complete
RSC	Reset Circuit
RES	Resume
SUS	Suspend
UBL	Unblocking
UBA	Unblocking Acknowledgment

Unsupported Message Types

As with the existing NTT version of ISUP services, the NTT-IC version of ISUP services does not support the message types listed in Table L-2 (which **are** supported by the CCITT version of ISUP services). These messages are treated as unrecognized by the SINAP node and discarded. No Confusion (CFN) message is sent to the remote node upon receipt of an unrecognized message.

Table L-2. Unsupported Message Types

Message Type	Description
CCR	Continuity Check Request
CFN	Confusion
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity Test
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Rejected
IDR	Identification Request

Table L-2. Unsupported Message Types *(Continued)*

Message Type	Description
IRS	Identification Response
INF	Information
INR	Information Request
LPA	Loopback Acknowledgment
NRM	Network Resource Management
OLM	Overload
PAM	Pass-Along
SAM	Subsequent Address
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

Message Parameters

Message parameters supported in the NTT-IC version of ISUP services are a subset of those supported in the CCITT version of ISUP services, in addition to those specified by the documents described at the beginning of this chapter.

Supported Message Parameters

The NTT-IC version of ISUP services recognizes several message parameters (which **are not** supported in the CCITT version of ISUP services). These are:

- Additional User Type *
- Carrier Information Transport *
- Charge Information
- Charge Information Delay *
- Charge Information Type
- Charging Zone Area Information
- Contractor (Subscriber) Number
- Domestic Redirection Reason *

- End Information Transport *
- Network Function Type
- PHS Terminal ID Number *
- Reason No Number Indicated

NOTE

Parameters marked with an asterisk (*) are also not supported in the NTT version of ISUP services. Of these additions, all (with the exception of End Information Transport, which is NTT-IC proprietary) are defined in the NTT-IC specifications, as well as in the TTC standard JT-Q763.

In addition to the parameters listed above, the NTT-IC version of ISUP services also supports the following message parameters (which **are** supported in the CCITT version of ISUP services). These are:

- Access Transport
- Backward Call Indicators
- Call Reference
- Called Party Number
- Calling Party Number
- Calling Party's Category
- Cause Indicators
- Circuit State Indicator
- Event Information
- Forward Call Indicators
- Generic (Universal) Number
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Original Called Number
- Range and Status
- Redirecting Number
- Redirection Information
- Redirection Number *
- Signalling Point Code *
- Suspend/Resume Indicators

- Transmission Medium Requirement
- User Service Information
- User-to-User Indicators
- User-to-User Information

NOTES

1. Parameters marked with an asterisk (*) are not supported in the NTT version of ISUP services. Of these additions, the Signaling Point Code parameter is not defined in the NTT-IC specifications.
2. Of those parameters that are supported in the NTT version of ISUP services, all but the Call Reference parameter are specified in the NTT-IC specifications and the TTC standard JT-Q763.

Unsupported Message Parameters

The NTT-IC version of ISUP services does not support the following message parameters (which **are** supported in the CCITT version of ISUP services):

- Access Delivery Information
- Automatic Congestion Level
- Call Diversion Information
- Call History Information
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code *
- Connected Number
- Connection Request
- Continuity Indicators
- Echo Control Information
- Facility Indicator
- Generic Digits
- Generic Notification
- Information Indicators
- Information Request Indicators
- Location Number

- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- MLPP Precedence
- Network Specific Facilities
- Optional Forward Call Indicators *
- Origination ISC Point Code
- Parameter Compatibility Information
- Pass-Along Information
- Propagation Delay Counter
- Redirection Number Restriction
- Remote Operations
- Service Activation
- Subsequent Number
- Transit Network Selection
- Transmission Medium Used
- User Service Information Prime
- User Teleservice Information

NOTE

Parameters marked with an asterisk (*) are supported in the NTT version of ISUP services.

Differences in Parameter Field Values

Table L-3 contains the differences in allowable parameter field values that are specific to the NTT-IC version of ISUP services.

Table L-3. Required Parameter Values

Message Parameter	Field	Required Value(s)
Backward Call Indicators	End-to-End Method Indicator	`E2EMTH_NOMETH` (=0)
	SCCP Method Indicator	`SCCPMTH_NOIND` (=0)

Table L-3. Required Parameter Values ***(Continued)***

Message Parameter	Field	Required Value(s)
Charging Zone Information	Charge Area Type	CHGAREA_CACODE (=1)
Domestic Redirection Reason	Redirecting Reason	DOMREDIRREAS_ROAMING (=126)
Forward Call Indicators	End-to-End Method Indicator	E2EMTH_NOMETH (=0)
	SCCP Method Indicator	SCCPMTH_NOIND (=0)
Calling Party's Category	Calling Party's Category	CAT_NATOPR (=9) CAT_ORD (=10) CAT_PRIOR (=11) CAT_TEST (=13) CAT_PAYPHONE (=15) CAT_TOPPRIOR[1] (=240)
Called Party Number	Address Signal Field	See Note 2.
	Nature of Address Indicator	NATNUM (=3) SPCLNETNUM (=126)
	Numbering Plan Indicator	ISDNNUM (=1)
Calling Party Number	Nature of Address Indicator	NATUM (=3) INTNATNUM (=4) SPCLNETNUM (=126)
	Address Presentation Restriction	PRESALLOW (=0) PRESREST (=1)
	Numbering Plan Indicator	ISDNNUM (=1)
Generic Number	Nature of Address Indicator	NATNUM (=3)
	Address Presentation Restriction	PRESALLOW (=0) PRESREST (=1)
	Numbering Plan Indicator	ISDNNUM (=1)
	Number Qualifier Indicator	NQ_ADDCGNMB (=6) NQ_ORIGCDNMB (=7)

Table L-3. Required Parameter Values ***(Continued)***

Message Parameter	Field	Required Value(s)
Original Called Number	Nature of Address Indicator	NATNUM (=3)
	Address Presentation Restriction	PRESALLOW (=0) PRESREST (=1)
	Numbering Plan Indicator	ISDNNUM (=1)
Redirecting Number	Nature of Address Indicator	NATNUM (=3)
	Address Presentation Restriction	PRESALLOW (=0) PRESREST (=1)
	Numbering Plan Indicator	ISDNNUM (=1)
Redirection Number	Nature of Address Indicator	NATNUM (=3)
	Numbering Plan Indicator	ISDNNUM (=1)
Contractor Number	Nature of Address Indicator	NATNUM (=3)
	Numbering Plan Indicator	ISDNNUM (=1)
Cause Indicators	Location	ILOC_USER (=0) ILOC_TRANNET (=3) ILOC_PUBNETRUS (=4) ILOC_PRIVNETRUS (=5) ILOC_INTNET (=7) ILOC_NETINTER (=10)
	Cause	CCUNALLOC (=1) CCNORTTOTSFNET (=2) CCNORTTODEST (=3) CCSENDSPCLTONE (=4) CCMISDIALDTRNK (=5) CCCALLCLR (=16) CCUSRBSY (=17) CCNOUSRRSP (=18) CCNOANSWR (=19) CCSUBSCRABSENT (=20) CCCALLRJT (=21) CCNMBRCHG (=22) CCREDIRNEWDEST[3] (=23) CCDESTOUTORD (=27) CCADDRINCOMP (=28) CCFACREJ (=29) CCNORMUNSPEC (=31) CCNOCIRCUIT (=34) CCNETAOL (=38) CCTMPFAIL (=41)

Table L-3. Required Parameter Values *(Continued)*

Message Parameter	Field	Required Value(s)
	Cause (cont.)	CCSWTCHCONG (=42) CCACCINFDISC (=43) CCREQUNAVAIL (=44) CCRESCUNAVAIL (=47) CCFACNOTSUB (=50) CCNOTAUTHBCAP (=57) CCBCAPUNAVAIL (=58) CCSERUNAVAIL (=63) CCBCAPNOTIMP (=65) CCFACNOTIMP (=69) CCRESTDIG (=70) CCSERVNOTIMP (=79) CCINCOMPDEST (=88) CCINVTRNSTNET (=91) CCINVMSG (=95) CCNOMSGTYP (=97) CCNOPARAMDISC (=99) CCNOPARAMPASS (=103) CCPROTERR (=111) CCINTRWRK (=127)
Nature of Connection Indicators	Continuity Check Indicators	CONTCHK_NOTREQ (=0)
	Satellite Indicator	SAT_NONE (=0) SAT_ONE (=1) SAT_TWO[3] (=2)
Redirection Information	Original Redirection Reasons	REAS_USRBUSY (=1) REAS_NOREPLY (=2) REAS_UNCOND (=3) REAS_DFLCDURALRT (=4) REAS_DFLCIMMDRSP (=5) REAS_MBLSUBNOTRCHBL[3] (=6)
	Redirecting Reason	REAS_USRBUSY (=1) REAS_NOREPLY (=2) REAS_UNCOND (=3) REAS_DFLCDURALRT (=4) REAS_DFLCIMMDRSP (=5) REAS_MBLSUBNOTRCHBL[3] (=6)
User-to-User Indicators	See Note 4.	Type: A bit (=1) Service 1: CB bits (=1)

NOTES

1. This field is provided to support the NTT-IC version of ISUP services, but are not defined in the NTT-IC specifications or in the TTC standard JT-Q763.
2. This field can accommodate a maximum length of 13 bytes, allowing it to represent phone numbers of up to 26 digits.
3. These fields are provided to support the NTT-IC version of ISUP services and are defined in the NTT-IC specifications or in the TTC standard JT-Q763.
4. This parameter is only recognized in the Address Complete (ACM) message.

Allowable Parameter Values

Table L-4 contains a list of the parameters and field values that are used in the NTT-IC version of ISUP services.

Table L-4. Allowable Parameter Values

Message Parameter	Where Used	Field	Allowable Value(s)
Access Transport	ACM[1], ANM[1], CPG[1], IAM[1]	[only one]	arbitrary series of bytes
Additional User Type	ACM[1], IAM[1]	[only one]	arbitrary series of bytes (up to 16)
Backward Call Indicators	ACM[2], ANM[1], CPG[1]	Charge Indicator (bits BA)	`CHRG_NOIND` =0 `CHRG_NOCHRG` =1 `CHRG_CHRG` =2
		Called Party's Status Indicator (bits DC)	`CADSTAT_NOIND` =0 `CADSTAT_SUBFREE` =1
		Called Party's Category Indicator (bits FE)	`CADCAT_NOIND` =0 `CADCAT_ORDSUBS` =1 `CADCAT_PAYPHONE` =2
		End-to-End Method Indicator (bits HG)	`E2EMTH_NOMETH` =0
		Interworking Indicator (bit I)	`INTIND_NOINTW` =0 `INTIND_INTW` =1
		End-to-End Information Indicator (bit J)	`E2EINF_NOINF` =0

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Backward Call Indicators (cont.)	ACM[2], ANM[1], CPG[1] (cont.)	ISDN User Part Indicator (bit K)	`ISUP_NOTUSED =0` `ISUP_USED =1`
		Holding Indicator (bit L)	`HOLD_NOTREQD =0`
		ISDN Access Indicator (bit M)	`ISDNACC_NONISDN =0` `ISDNACC_ISDN =1`
		Echo Control Device Indicator (bit N)	`ECHOCDEV_NOTINCL =0` `ECHOCDEV_INCL =1`
		SCCP Method Indicator (bits OP)	`SCCPMTH_NOIND =0`
Call Reference	ACM[1], ANM[1], CPG[1], IAM[1]	Call Identity	three arbitrary bytes
		Point Code	two arbitrary bytes
Called Party Number	IAM[2]	Even/Odd Indicator	`NMB_EVEN =0` `NMB_ODD =1`
		Nature of Address Indicator	`NATNUM =3` `SPCLNETNUM =126`
		Internal Network Number Indicator	`INN_ALLOW =0`
		Numbering Plan Indicator	`ISDNNUM =1`
		Address Signal	arbitrary series of bytes (up to 13)

Table L-4. Allowable Parameter Values *(Continued)*

Message Parameter	Where Used	Field	Allowable Value(s)
Calling Party Number	IAM[1]	Even/Odd Indicator	`NMB_EVEN =0` `NMB_ODD =1`

		Nature of Address Indicator	`NATNUM =3`
			`INTNATNUM =4`
			`SPCLNETNUM =126`

		Number Incomplete Indicator	`NMB_COMPLTE =0`
		Numbering Plan Indicator	`ISDNNUM =1`
		Address Presentation Restricted Indicator	`PREASALLOW =0` `PRESREST =1`
		Screening Indicator	`USRPROV =1` `NETPROV =3`
Calling Party Number (cont.)	IAM[1] (cont.)	Address Signal	arbitrary series of bytes (up to 10)

Calling Party's Category	IAM[2]	[only one]	`CAT_NATOPR =9`
			`CAT_ORD =10`
			`CAT_PRIOR =11`
			`CAT_TEST =13`
			`CAT_PAYPHONE =15`
			`CAT_TOPPRIOR =240`

Carrier Information Transport	ACM[1], CPG[1], IAM[1]	[only one]	arbitrary series of bytes (up to 97)

Table L-4. Allowable Parameter Values *(Continued)*

Message Parameter	Where Used	Field	Allowable Value(s)
Cause Indicators	ACM[2], CPG[2], REL[1]	Coding Standard	CSTD_CCITT =0 CSTD_INT =1 CSTD_NAT =2 CSTD_SPECLOC =3
		Location	ILOC_USER =0 ILOC_TRANNET =3 ILOC_PUBNETRUS =4 ILOC_PRIVNETRUS =5 ILOC_INTNET =7 ILOC_NETINTER =10
		Cause Value	CCUNALLOC =1 CCNORTTOTSFNET =2 CCNORTTODEST =3 CCSENDSPCLTONE =4 CCMISDIALDTRNK =5 CCCALLCLR =16 CCUSRBSY =17 CCNOUSRRSP =18 CCNOANSWR =19 CCSUBSCRABSENT =20 CCCALLRJT =21 CCNMBRCHG =22 CCREDIRNEWDEST =23 CCDESTOUTORD =27 CCADDRINCOMP =28 CCFACREJ =29 CCNORMUNSPEC =31 CCNOCIRCUIT =34 CCNETAOL =38 CCTMPFAIL =41 CCSWTCHCONG =42 CCACCINFDISC =43

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Cause Indicators (cont.)	ACM[2], CPG[2], REL[1] (cont.)	Cause Value (cont.)	CCREQUNAVAIL =44 CCRESCUNAVAIL =47 CCFACNOTSUB =50 CCNOTAUTHBCAP =57 CCBCAPUNAVAIL =58 CCSERVUNAVAIL =63 CCBAPNOTIMP =65 CCFACNOTIMP =69 CCRESTDIG =70 CCSERVNOTIMP =79 CCINCOMPDEST =88 CCINVTRNSTNET =91 CCINVMSG =95 CCNOMSGTYP =97 CCNOPARAMDISC =99 CCNOPARAMPASS =103 CCPROTERR =111 CCINTRWRK =127
		Diagnostic(s)	arbitrary series of bytes
Charge Information Delay	ACM[1]	[only one]	up to 2 arbitrary bytes
Charging Zone (Area) Information	ACM[1], ANM[1], CHG[1], IAM[1]	Odd/Even Indicator	NMB_EVEN =0 NMB_ODD =1
		Charge Area Type	CHGAREA_CACODE =1
		Address Signal	arbitrary series of bytes (up to 3)
Charge Information	ACM[1], CHG[2]	[only one]	arbitrary series of bytes
Charge Information Type	ACM[1], CHG[2]	[only one]	one arbitrary byte
Circuit State Indicator	CQR[2]	[only one]	arbitrary series of bytes (See Note 3)
Contractor Number	IAM[2]	Odd/Even Indicator	NMB_EVEN =0 NMB_ODD =1
		Nature of Address Indicator	NATNUM =3

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Contractor Number (cont.)	IAM[2] (cont.)	Numbering Plan Indicator	`ISDNNUM =1`
		Address Signal	arbitrary series of bytes (up to 8)
Domestic Redirection Reason	IAM[1], REL[1]	[only one]	`DOMREDIRREAS_ ROAMIONG =126`
End Information Transport	IAM[1]	[only one]	arbitrary series of bytes (up to 126)

Event Information	CPG[2]	Event Indicator	`EV_ALERT =1`
			`EV_PROGRESS =2`
			`EV_INBAND =3`
			`EV_FWDONBUSY =4`
			`EV_FWDONNOREP =5`
			`EV_FWDUNCONDIT =6`

		Event Presentation Restricted Indicator	`AVPR_NOIND =0` `EVPR_PRESRES =1`
Forward Call Indicators	IAM[2]	National / International Call Indicator (bit A)	`CALL_NAT =0` `CALL_INTERNAT =1`
		End-to-End Method Indicator (bits CB)	`E2EMTH_NOMETH =0`
		Interworking Indicator (bit D)	`INTIND_NOINTW =0` `INTIND_INTW =1`
		End-to-End Information Indicator (bit E)	`E2EINF_NOINFO =0`
		ISDN User Part Indicator	`ISUP_NOTUSED =0` `ISUP_USED =1`

		ISDN User Part Preference Indicator (bits HG)	`PREF_PREFAW =0`
			`PREF_NOTREQAW =1`
			`PREF_REQAW =2`

		ISDN Access Indicator (bit I)	`ISDNACC_NONISDN =0` `ISDNACC_ISDN =1`
		SCCP Method Indicator (bits KJ)	`SCCPMTH_NOIND =0`

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Generic (Universal) Number	IAM[1]	Number Qualifier Indicator	NQ_ADDCGNMB =6 NQ_ORIGCDNMB =7
		Odd/Even Indicator	NMB_EVEN =0 NMB_ODD =1
		Nature of Address Indicator	NATNUM =3
		Number Incomplete Indicator	NMB_COMPLTE =0
		Numbering Plan Indicator	ISDNNUM =1
		Address Presentation Restricted Indicator	PREASALLOW =0 PRESREST =1
		Screening Indicator	USRPROV =1 NETPROV =3
		Address Signal	arbitrary series of bytes (up to 9)
Nature of Connection Indicators	IAM[2]	Satellite Indicator	SAT_NONE =0 SAT_ONE =1 SAT_TWO =2
		Continuity Check Indicators (bits DC)	CONTCHK_NOTREQ =0
		Echo Control Device Indicator (bit E)	ECHOCDEV_NOTINCL =0 ECHOCDEV_INCL =1
Network Function Type	IAM[1]	[only one]	one arbitrary byte
Optional Backward Call Indicators	ACM[1], CPG[1]	In-Band Information Indicator (bit A)	INBND_NOIND =0 INBND_AVAIL =1
		Call Forwarding May Occur Indicator (bit B)	CAFWD_NOIND =0
Original Called Number	IAM[2]	Odd/Even Indicator	NMB_EVEN =0 NMB_ODD =1
		Nature of Address Indicator	NATNUM =3
		Numbering Plan Indicator	ISDNNUM =1

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Original Called Number (cont.)	IAM[2] (cont.)	Address Presentation Restricted Indicator	`PREASALLOW =0` `PRESREST =1`
		Address Signal	arbitrary series of bytes (up to 10)
PHS Terminal ID Number	IAM[1]	Odd/Even Indicator	`NMB_EVEN =0` `NMB_ODD =1`
		Nature of Address Indicator	`NATNUM =3`
		Numbering Plan Indicator	`ISDNNUM =1`
		Address Signal	arbitrary series of bytes (up to 8)
Range and Status	CQM[2], CQR[2], GRA[2], GRS[2]	Range	`CQM/CQR 0 - 31` `GRA/GRS 0 - 11`
		Status	arbitrary series of bits

Reason No Number Indicated	IAM[1]	[only one]	`RNNI_USERREQUEST =1`
			`RNNI_COMPETITION =2`
			`RNNI_PUBLICPHONE =3`

Redirecting Number	IAM[1]	Odd/Even Indicator	`NMB_EVEN =0` `NMB_ODD =1`
		Nature of Address Indicator	`NATNUM =3`
		Numbering Plan Indicator	`ISDNNUM =1`
		Address Presentation Restricted Indicator	`PREASALLOW =0` `PRESREST =1`
		Address Signal	arbitrary series of bytes (up to 10)

Redirection Information	IAM[1]	Redirecting Indicator (bits CBA)	`RI_CALLFWD =3`
			`RI_CALLFWDALLRI`
			`PRESRES =4`

		Original Redirection Reasons (bits HGFE)	`REAS_USRBUSY =1`
			`REAS_NOREPLY =2`
			`REAS_UNCOND =3`
			`REAS_DFLCDURALRT =4`

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Redirection Information (cont.)	IAM[1] (cont.)	Original Redirection Reasons (bits HGFE) (cont.)	`REAS_DFLCIMMDRSP =5` `REAS_MBLSUBNOTRCHBL =6`
		Redirection Counter (bits KJI)	`1 - 5`

		Redirecting Reason (bits POMN)	`REAS_USRBUSY =1`
			`REAS_NOREPLY =2`
			`REAS_UNCOND =3`
			`REAS_DFLCDURALRT =4`
			`REAS_DFLCIMMDRSP =5`
			`REAS_MBLSUBNOTRCHBL =6`

Redirection Number	REL[1]	Odd/Even Indicator	`NMB_EVEN =0` `NMB_ODD =1`
		Nature of Address Indicator	`NATNUM =3`
		Internal Network Number Indicator	`INN_ALLOW=0`
		Numbering Plan Indicator	`ISDNNUM =1`
		Address Signal	arbitrary series of bytes (up to 10)
Signalling Point Code	REL[1]	[only one]	two arbitrary bytes[4]
Suspend/Resume Indicators	RES[2], SUS[2]	Suspend/Resume Indicator (bit A)	`SR_ISDNSUBINIT =0` `SR_NETINIT =1`

Transmission Medium Requirement	IAM[2]	[one only]	`TMR_SPEECH =0`
			`TMR_64KBITS =2`
			`TMR_31KHZ =3`

Table L-4. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
User Service Information	IAM[1]	Coding Standard	CSTD_CCITT =0 CSTD_INT =1 CSTD_NAT =2 CSTD_NET =3
		Information Transfer Capability	ITC_SPEECH =0 ITC_UNRDIG =8 ITC_A31KHZ =16
		Transfer Mode	TM_CIRCUIT =0 TM_PACKET =2
User Service Information (cont.)	IAM[1] (cont.)	Information Transfer Rate	ITR_PKT =0 ITR_64KBIT =16 ITR_384KBIT =19 ITR_1536KBIT =21
		See Note 5.	
User-to-User Indicators	ACM[1]	Type (bit A)	1 for Response
		Service 1 (bits CB)	1 for Not Provided
		Service 2 (bits ED)	0
		Service 3 (bits GF)	0
User-to-User Information	ACM[1], ANM[1], CPG[1], IAM[1], REL[1]	[only one]	arbitrary series of bytes

NOTES

1. Parameter is optional in this message.
2. Parameter is mandatory in this message.
3. These bytes are interpreted according to Q.763 to represent unequipped or transient circuit states, or combinations of the call processing and maintenance blocking states.
4. This parameter will always be sent regardless of whether or not it is supplied by the user. This feature is not defined in the NTT-IC specifications.
5. Several additional fields are defined in optional extension bytes per Q.931.

Pointer to the Start of the Optional Part

The format of a message for the NTT-IC version of ISUP services, as well as for the NTT version of ISUP services, is defined to always include a pointer to the start of the optional part of a message, regardless of whether or not the message can have optional parameters. The pointer is all zeros whenever there is no optional part in the message.

A CCITT ISUP message definition does not include a pointer to the start of the optional part for messages which cannot have an optional part.

API Support for Additional Parameters and Fields

Each of the new parameters and fields supported by the NTT-IC version of ISUP services is visible to the application program at the API. Structures that have been added or modified to accommodate these parameters and fields are defined in the header file `issl_sit.h`. In addition, elements have been added to the event structures defined in this file. The affected structures are explained in more detail in the following section.

NOTE

Each of the modifications that were made to the structures listed in the following table are thoroughly commented in the `issl_sit.h` header file to help identify the changes required for the NTT-IC version of ISUP services.

NTT-IC API Structures

The API structures which have been modified in the `issl_sit.h` header file for the NTT-IC version of ISUP services are listed in Table L-5. The notes following the table briefly describe the changes made to the structures. Refer to the comments contained in the header file pertaining to the structures for a complete description.

Table L-5. New API Structure List

Structure	Description
`chgInfoDelay`	Charge Information Delay
`addUserType`	Additional User Type
`carInfoType`	Carrier Information Transport
`domRedirReas`	Domestic Redirection Reason
`endInfoTrans`	End Information Transport
`chargeNum`[1]	Charge Number
`chgZoneInfo`[2]	Charge Zone Information

Table L-5. New API Structure List*(Continued)*

Structure	Description
`siConEvnt`[3]	Connection Setup Event
`siCnStEvnt`[4]	Connection Status Event
`siRelEvnt`[5]	Connection Release Event
`siStaEvnt`[6]	Circuit Status Event

NOTES

1. This structure is currently used in the NTT version of ISUP services to support the Contractor Number parameter. In the NTT-IC version of ISUP services this structure is also used to support the new PHS Terminal ID Number parameter.
2. The existing NTT ISUP structure has been expanded to add a new field (which was formerly spare) to accommodate NTT-IC ISUP.
3. Elements have been added to this structure to support the additional parameters that can be sent and received at the API for the Connection Setup primitives.
4. Elements have been added to this structure to support the additional parameters that can be sent and received at the API for the Connection Status primitives.
5. An element has been added to this structure to support the additional parameter that can be sent and received at the API for the Connection Release primitives.
6. An element has been added to this structure to support the additional parameter that can be sent and received at the API for the Connection Status primitives.

Message Parameter Allocation

Table L-6 lists each of the messages for which different parameters are applicable, along with the parameters supported by that message.

Table L-6. Differences in Message Parameters

Message	Parameters
Initial Address (IAM)	Nature of Connection Indicators[1]
	Forward Call Indicators[1]
	Calling Party's Category[1]
	Transmission Medium Requirement[1]
Initial Address (IAM) (cont.)	Called Party Number[1]
	Call Reference
	Calling Party Number
	Redirecting Number
	Redirection Information
	Original Called Number
	User-to-User Information
	Access Transport
	User Service Information
	Generic (Universal) Number
	Reason No Number Indicated
	Contractor (Subscriber) Number
	Charging Zone (Area) Information
	Network Function Type
	PHS Terminal ID Number[2]
	Additional User Type[2]
	Carrier Information Transport[2]
	Domestic Redirection Reason[2]
	End Information Transport[2]
	Backward Call Indicators
Address Complete (ACM)	Optional Backward Call Indicators
	Cause Indicators

Table L-6. Differences in Message Parameters*(Continued)*

Message	Parameters
	User-to-User Indicators
	User-to-User Information
	Access Transport
	Call Reference
	Charge Information Type[2]
Address Complete (ACM) (cont.)	Charge Information[2]
	Charging Zone (Area) Information[2]
	Charge Information Delay[2]
	Additional User Type[2]
	Carrier Information Transport[2]
Call Progress (CPG)	Event Information[1]
	Backward Call Indicators
	Optional Backward Call Indicators
	Cause Indicators
	User-to-User Information
	Access Transport
	Charging Zone (Area) Information[2]
	Carrier Information Transport[2]
Answer (ANM)	Backward Call Indicators
	User-to-User Information
	Access Transport
	Charging Zone (Area) Information[2]
Charge Information (CHG)	Charge Information Type[1]
	Charge Information[1]
	Charging Zone (Area) Information[2]
Release (REL)	Cause Indicators[1]
	Redirection Number[2]

Table L-6. Differences in Message Parameters*(Continued)*

Message	Parameters
	Signaling Point Code[2]
	User-to-User Information
	Domestic Redirection Reason[2]
Release Complete (RLC)[3]	None

NOTES

1. These parameters are mandatory in the NTT-IC version of ISUP services.
2. These parameters are not supported in the NTT version of ISUP services.
3. The Signalling Point Code parameter will always be sent regardless of whether or not it is supplied by the user. This feature is not defined in the NTT-IC specifications.

Differences in Protocol Procedures

The procedures for handling unrecognized parameters in the NTT-IC version of ISUP services are essentially the same as those in the NTT version of ISUP services. The only differences lie in the criteria used for determining whether a parameter is considered unrecognized.

Any differences between the two versions and the CCITT/ITU version of ISUP services are listed in the following sections.

Unrecognized Messages

All unrecognized messages received by the NTT-IC version of ISUP services are discarded. A Confusion (CFN) message is **not** sent in reply.

Interpretation of Spare Field Values

The procedures for interpreting spare values in parameter fields in the NTT-IC version of ISUP services are essentially the same as those in the NTT version of ISUP services. The main difference is the criteria used for determining whether a field value is considered unrecognized or spare, since many field have different sets of allowable values in the NTT-IC version of ISUP services.

The fields for which a default value will be passed up to the user if the input is not one of the allowable values are contained in Table L-7.

Table L-7. Parameter Differences List

Message Parameter	Field
Backward Call Indicators	Charge Indicator
	Called Party's Status Indicator
	Called Party's Category Indicator
	End-to-End Method Indicator
	End-to-End Information Indicator
	SCCP Method Indicator
Calling Party Number	Address Presentation Restricted Indicator
	Screening Indicator
Cause Indicators	Location
	Cause Value
Forward Call Indicators	End-to-End Method Indicator
	End-to-End Information Indicator
	SCCP Method Indicator
Generic (Universal) Number	Address Presentation Restricted Indicator
	Screening Indicator
Nature of Connection Indicators	Satellite Indicator
	Continuity Check Indicator
Original Called Number	Address Presentation Restricted Indicator
Redirecting Number	Address Presentation Restricted Indicator
Redirection Information	Redirecting Indicator
	Original Redirection Reason
	Redirection Counter
	Redirecting Reason

Table L-7. Parameter Differences List *(Continued)*

Message Parameter	Field
User-to-User Indicators	Service 2
	Service 3

Disabled Options

Three protocol features which are available in the CCITT version of ISUP services are not supported by the NTT-IC version of ISUP services. These are:

- Unequipped Circuit Identification Code (UCIC) messages are **not** automatically sent as a reply to a message that is received for an unequipped circuit;
- Protocol procedures are not used for continuity checking; and
- There is **no** ability to perform segmentation of over-length messages for certain message types.

When the NTT-IC version of ISUP services is configured, the following capability-based option bits are set to disable the optional features listed above:

```
ISUP_OPT_NO_UCIC_REPLIES
ISUP_OPT_NO_CONT_CHECKING
ISUP_OPT_NO_SEGMENTATION
```

The protocol behavior of these option bits is the same in the NTT version of ISUP services.

ISUP Services Timers

The NTT-IC version of ISUP services supports the same set of protocol timers as those supported in the NTT version of ISUP services. A list of supported timers is contained in Table L-8.

Table L-8. Supported NTT-IC ISUP Services Timers

Timer	Description
T1[1]	REL message sent
T2	SUS (user) message received
T5[2]	Initial REL message sent
T6	SUS (network) message received
T7	Latest address message sent
T12	BLO message sent

Table L-8. Supported NTT-IC ISUP Services Timers ***(Continued)***

Timer	Description
`T13`	Initial BLO message sent
`T14`	UBL message sent
`T15`	Initial UBL message sent
`T16`[1]	RSC message (not due to T5 expiry) sent
`T17`[2]	Initial RSC message sent
`T22`	GRS message sent
`T23`	Initial GRS message sent

NOTES

1. The IST1 timer is used for these timers.
2. The IST5 timer is used for these timers.

Appendix M
Swedish Version of ISUP Services

The Swedish version of ISUP services, with minor modification, adheres to the Switching and Signalling Specifications of SS7 ISDN User Part (ISUP) Q.761 through Q.764, dated March 1993.

Implementing the Swedish Version of ISUP Services

To implement the Swedish version of ISUP services, un-comment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `SWEDEN`.

```
ISUP_FEATURE=SWEDEN
```

NOTES

1. The CCITT network variant must be configured on the SINAP node in order to activate the Swedish version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).
2. The Swedish version of ISUP services requires that the Network indicator in the MTP routing table be set to 11 (reserved for National use). Use the MML command `CREATE-OSP` to set the `NETWORK` field to `11`.

Differences Between the Swedish and CCITT Versions of ISUP Services

The Swedish version of ISUP services differs from the CCITT/ITU-T version of ISUP services that SINAP/SS7 currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences between the two versions.

For more detailed information on the differences in the Swedish version of ISUP services, see the ITU-T Recommendations for Switching and Signalling Specifications of SS7 - ISDN User Part (ISUP) Q.761 through Q.764.

Message Types

The Swedish version of ISUP services supports a restricted set of messages which are also supported in the CCITT version of ISUP services.

Supported Message Types

The Swedish version of ISUP services recognizes and processes the subset of standard ITU-T message types listed in Table M-1. The SINAP node treats all other messages as unrecognized and discards them.

Table M-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
REL	Release
RES	Resume
RLC	Release Complete
RSC	Reset Circuit

Table M-1. Supported Message Types*(Continued)*

Message Type	Description
SAM	Subsequent Address
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The Swedish version of ISUP services treats the message types listed in Table M-2 (supported by the CCITT version of ISUP services) as unrecognized. The SINAP node discards these messages and does not send a Confusion (CFN) message to the remote node.

Table M-2. Unsupported Message Types

Message Type	Description
CCR	Continuity Check Request
CFN	Confusion
CHG	Charge Information
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CQM	Circuit Group Query
CQR	Circuit Group Query Response
DRS	Delayed Release
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FOT	Forward Transfer
FRJ	Facility Reject
IDR	Identification Request
INF	Information
INR	Information Request

Table M-2. Unsupported Message Types*(Continued)*

Message Type	Description
IRS	Identification Response
LPA	Loopback Acknowledgment
NRM	Network Resource Managment
OLM	Overload
PAM	Pass-Along
SGM	Segmentation
UCIC	Unequipped Circuit Identification Code
UPA	User Part Available
UPT	User Part Test
USR	User-to-User Information

Message Parameters

Message parameters supported in the Swedish version of ISUP services are a subset of those supported in the CCITT version of ISUP services.

Supported Message Parameters

The Swedish version of ISUP services supports the following message parameters:

- Access Transport
- Backward Call Indicators
- Call Diversion Information
- Called Party Number
- Calling Party Number
- Calling Party’s Category
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code
- Connected Number
- Continuity Indicators
- End of Optional Parameters
- Event Information

- Forward Call Indicators
- Generic Notification
- Location Number
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Original Called Number
- Parameter Compatibility Information
- Range and Status
- Redirecting Number
- Redirection Information
- Redirection Number
- Redirection Number Restriction
- Subsequent Number
- Suspend/Resume Indicators
- Transmission Medium Requirement
- User Service Information
- User-to-User Indicators
- User-to-User Information

Unsupported Message Parameters

The Swedish version of ISUP services does not support the following message parameters (which are supported in the CCITT version of ISUP services):

- Access Delivery Information
- Automatic Congestion Level
- Call History Information
- Call Modification Indicators
- Call Reference
- Circuit State Indicator
- Connection Request
- Echo Control Information
- Facility Indicator

- Generic Digits
- Generic Number
- Information Indicators
- Information Request Indicators
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- MLPP Precedence
- Network Specific Facilities
- Origination ISC Point Code
- Pass-Along Information
- Propagation Delay Counter
- Remote Operations
- Service Activation
- Signalling Point Code
- Transit Network Selection
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information Prime
- User Teleservice Information

Parameter Field Values

The parameter values listed in Table M-3 are specified for the Swedish version of ISUP services.

Table M-3. Required Parameter Values

Message Parameter	Field	Required Value(s)
Backward Call Indicators	Called Party's Status	`CADSTAT_NOIND (=0)` `CADSTAT_SUBFREE (=1)`
	Holding Indicator	`HOLD_NOTREQD (=0)`

Table M-3. Required Parameter Values ***(Continued)***

Message Parameter	Field	Required Value(s)
Backward Call Indicators (cont.)	End-to-End Information indicator	`E2EINF_NOINF` (=0)
	End-to-End Method Indicator	`E2EMTH_NOMETH` (=0)
	SCCP Method Indicator	`SCCPTH_NOIND` (=0)
Forward Call Indicators	End-to-End Information Indicator	`E2EINF_NOINFO` (=0)
	End-to-End Method Indicator	`E2EMTH_NOMETH` (=0)
	SCCP Method Indicator	`SCCPMTH_NOIND` (=0)
Called Party Number	Nature of Address Indicator	`SUBSNUM` (=2) `NATNUM` (=3) `INTNATNUM` (=4)
	Numbering Plan Indicator	`ISDNNUM` (=1)
Calling Party Number	Nature of Address Indicator	`NATUM` (=3) `INTNATNUM` (=4)
	Calling Party Number Incomplete	`NBMCMLTE` (=0)
	Calling Party Numbering Plan	`ISDNNUM` (=1)
	Screening Indicator	`USRPROV` (=1) `NETPROV` (=3)
	Address Presentation Restriction	`PRESALLOW` (=0) `PRESREST` (=1)

Table M-3. Required Parameter Values ***(Continued)***

Message Parameter	Field	Required Value(s)
Calling Party's Category		CAT_OPLANGFR (=1) CAT_OPLANGENG (=2) CAT_OPLANGGER (=3) CAT_OPLANGRUS (=4) CAT_OPLANGSP (=5) CAT_ADMIN1 (=6) CAT_ADMIN2 (=7) CAT_ADMIN3 (=8) CAT_ORD (=10) CAT_PRIOR (=11) CAT_DATA (=12) CAT_TEST (=13) CAT_PAYPHONE (=15)

Cause Indicators	Coding Standard	CSTD_CCITT (=0)
	Location	ILOC_USER (=0) ILOC_TRANNET (=3) ILOC_PUBNETRUS (=4) ILOC_PRIVNETRUS (=5) ILOC_LOCALIF (=6) ILOC_INTNET (=7) ILOC_NETINTER (=10)
Connected Number	Nature of Address Indicator	NATUM (=3) INTNATNUM (=4)
Location Number	Presentation Restriction	See note 1.
	Screening Indicator	See note 1.
Nature of Connection Indicators	Continuity Check Indicators	CONTCHK_NOTREQ (=0) CONTCHK_PREV (=2)
Original Called Number	Nature of Address Indicator	See note 2.
	Presentation Restriction	See note 1.
	Numbering Plan Indicator	See note 2.
Parameter Compatibility Information	Pass on not possible[3]	RELCAL (=0) DISCRDMSG (=1) DISCRDPARM (=2)

Table M-3. Required Parameter Values ***(Continued)***

Message Parameter	Field	Required Value(s)
Redirecting Number	Nature of Address Indicator	See note 2.
	Presentation Restriction	See note 2.
	Numbering Plan Indicator	See note 2.
Redirection Number	Nature of Address Indicator	See note 2.
	Numbering Plan Indicator	See note 2.
Transmission Medium Requirement		`TMR_SPEECH` (=0) `TMR_64KBITS` (=2) `TMR_31KHZ` (=3)
User-to-User Indicators		`Type: A bit` (=1) `Service 1: CB` (=00) `Service 2: ED` (=00) `Service 3: GF` (=00)
	Network Discard Indicator[4]	`H` bit (=1)

NOTES

1. See *Calling Party Number*.
2. See *Called Party Number*.
3. If the value `3` is received for this field, it must be interpreted as `0`.
4. This field is not currently implemented.

Differences in Protocol Procedures

The procedures for handling unrecognized messages and parameters are essentially the same as in the CCITT version of ISUP services. Any exceptions between the two versions are listed in the following sections.

Unrecognized Messages

Unrecognized messages received by the Swedish version of ISUP services are discarded. A Confusion (CFN) message is **not** sent in reply.

Unknown Parameter Values

Upon receiving unrecognized message parameters, the Swedish version of ISUP services essentially adheres to the "Echo Control Procedure" as defined in Section 2.7 of the ITU-T Q.764 Signalling Procedures recommendation and Annex C.A of the ITU-T Q.767

recommendation. Differences between the Swedish version of ISUP services and these recommendations are described in Table M-4.

NOTE

Either the action `ignore` or `No default` will pass on the value to call control in any condition.

Table M-4. Parameter Differences List

Message Parameter	Swedish ISUP Version	ITU-T ISUP Version
Backward Call Indicators[1] Holding Indicator (national use) Bit L:	Ignored	Default: 0 "holding not requested"
Cause Indicators[2] Location	Default: `0111` "International Network"	International: Default "international network" National: Default "beyond an interwork point"
Event Information[3] Event Indicator: bits G-A	No default	Discard message
Optional Backward Call Indicators[4] Call diversion may occur: bit B	Default `0`	No description
User-to-User Indicators Type bit A: Network Discard Indicator bit H:	Default `1` "response" Default `1` "Network discard"	No description No description

NOTES

1. Since the default action is to ignore the spare field, all values are allowed to pass to call control. The allowable

enum-table for this parameter (`teHoldInd`) is `teHoldIndEnums0[]`

```
HOLD_NOTREQ (=0)

HOLD_REQD   (=1)
```

2. For the token `teLocation`, the default value should be set to `ILOC_INTNET`. The global variable `isup_rel_location_dflt` should be set to `ILOC_INTNET` since this will also be the default value for the release cause.
3. Since `No default` is the action for the spare field value, all values are allowed to pass to call control. The allowable enum-table is `teDefEnums7[]`.
4. This is a single-bit filed with both `0` and `1` as allowable values.

Disabled Options

Three protocol features which are available in the ITU-T version of ISUP services are not supported by the Swedish version of ISUP services. These are:

- Unequipped Circuit Identification Code (UCIC) messages are **not** automatically sent as replies to messages that are received for unequipped or un-configured circuits
- There is **no** ability to perform segmentation of over-length messages for certain message types
- A Confusion (CFN) message is **not** automatically sent upon receipt of an unrecognized message

ISUP Services Timers

The Swedish version of ISUP services supports all ITU-T timers, with the exception of those described in Table M-5.

Table M-5. Supported Swedish ISUP Services Timers

Timer	Description
`T4`	Remote user unavailable MTP status event received
`T27`	Receipt of continuity check failure indication
`T33`	Information Request (INR) message sent

Table M-5. Supported Swedish ISUP Services Timers

Timer	Description
T34	Receipt of segmented message
T36	Continuity recheck timer

NOTE

After receiving the Continuity (COT) with failure message after the Initial Address (IAM) message with continuity check indicator, the SINAP node sends a Release (REL) message to the remote node. This is due to the fact that the Swedish version of ISUP services does not support timer T36 (continuity recheck timer).

Appendix N
Taiwanese Version of ISUP Services

The Taiwanese version of ISUP services adheres to the Specifications for Signalling System No. 7 ME 0001-3, Jan 97. This specification is based on the 1993 edition of the ITU-T Recommendations for ISUP (Q.761 -Q.764).

Implementing the Taiwanese Version of ISUP Services

To implement the Taiwanese version of ISUP services, un-comment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `TAIWAN`.

```
ISUP_FEATURE=TAIWAN
```

Differences Between the Taiwanese and CCITT Versions of ISUP Services

The Taiwanese version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports which is based on the 1993 edition of ITU-T Recommendations for ISUP services. The following sections describe the differences between the two versions.

For more detailed information on the differences in the Taiwanese version of ISUP services, see the ITU-T Recommendations for Switching and Signalling Specifications of SS7 - ISDN User Part (ISUP) Q.761 through Q.764.

Message Types

The Taiwanese version of ISUP services supports a subset set of messages which are also supported in the CCITT version of ISUP services.

Supported Message Types

As directed in Section 6.1 of ME 0001-3, the Taiwanese version of ISUP services supports a subset of ITU-T message types described in Q.763. The supported message types are listed in Table N-1.

Table N-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CGB	Circuit Group Blocking
CCR	Continuity Check Request
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CFN	Confusion
CON	Connect
COT	Continuity
CPG	Call Progress
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FRJ	Facility Reject
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
INF	Information
INR	Information Request
PAM	Pass-Along
REL	Release

Table N-1. Supported Message Types *(Continued)*

Message Type	Description
`RES`	Resume
`RLC`	Release Complete
`RSC`	Reset Circuit
`SAM`	Subsequent Address
`SGM`	Segmentation
`SUS`	Suspend
`UBA`	Unblocking Acknowledgment
`UBL`	Unblocking
`USR`	User-to-User Information

Unsupported Message Types

The Taiwanese version of ISUP services treats the message types listed in Table N-2 (supported by CCITT ISUP services) as unrecognized messages and ignores them.

Table N-2. Unsupported Message Types

Message Type	Description
`CHG`	Charge Information
`CMC`	Call Modification Completed
`CMR`	Call Modification Request
`CMRJ`	Call Modification Reject
`FOT`	Forward Transfer
`IDR`	Identification Request
`IRS`	Identification Response
`LPA`	Loopback Acknowledgment
`NRM`	Network Resource Management
`OLM`	Overload
`UCIC`	Unequipped Circuit Identification Code

Table N-2. Unsupported Message Types *(Continued)*

Message Type	Description
UPA	User Part Available
UPT	User Part Test

Message Parameters

Message parameters supported in the Taiwanese version of ISUP services are a subset of the message parameters supported in SINAP/SS7's CCITT version of ISUP services.

Supported Message Parameters

As directed in Section 7 of ME 0001-3, the Taiwan ISUP services supports a subset of ITU-T parameters described in Q.763. The supported message parameters are listed in Table N-3.

Table N-3. Supported Message Parameters

Message Parameter	Description
ACCDELINFO	Access Delivery Information
ACCTPORT	Access Transport
AUTOCONGLVL	Automatic Congestion Level
BACKCALLIND	Backward Call Indicators
CALGPARTCAT	Calling Party's Category
CALDPARTNUM	Called Party Number
CALGPARTNUM	Calling Party Number
CAUSIND	Cause Indicator
CGRPSUPMTYPIND	Circuit Group Supervision Message Type Indicator
CIRCSTEIND	Circuit State Indicator
CLLDIVERS	Call Diversion Information
CLSDUGRPINTCDE	Closed User Group Interlock Code
CONNUMB	Connected Number
CONTIND	Continuity Indicators
ENDOP	End of Optional Parameters
EVNTINFO	Event Information
FACIND	Facility Indicator

Table N-3. Supported Message Parameters *(Continued)*

Message Parameter	Description
FWDCALLIND	Forward Call Indicator
GENDIGITS	Generic DIgits
GENNMB	Generic Number
INFOIND	Information Indicators
INFOREQIND	Information Request Indicators
LOCNMB	Location Number
MLPPPREC	MLPP Precedence
MSGCOMP	Message Compatibility Information
NATCONIND	Nature of Connection Indicators
NETSPFAC	Network Specific Facilities
NOTIFINDC	Generic Notification
OPBACKCALLIND	Optional Backward Call Indicators
OPFWDCALLIND	Optional Forward Call Indicators
ORIGCALDNUM	Original Called Number
ORISCCDE	Origination ISC Point Code
PARCOMPIN	Parameter Compatibility Information
RANGSTAT	Range and Status
REDIRGNUM	Redirecting Number
REDIRINFO	Redirection Information
REDIRNUM	Redirection Number
REDIRRESTR	Redirection Number Restriction
REMOTOPER	Remote Operations
SERVACT	Service Activation
SIGPTCDE	Signalling Point Code
SUBSEQNUM	Subsequent Number
SUSPRESIND	Suspend/Resume Indicators
TRANSMEDPRM	Transmission Medium Requirement Prime

Table N-3. Supported Message Parameters *(Continued)*

Message Parameter	Description
TRANSMEDREQ	Transmission Medium Requirement
TRANSMEDUSD	Transmission Medium Used
TSRVINFO	User Teleservice Information
USR2USRIND	User-to-User Indicators
USR2USRINFO	User-to-User Information
USRSERVINFO	User Service Information
USRSERVINFOPR	User Service Information Prime

Unsupported Message Parameters

The Taiwanese version of ISUP services does not support the message parameters listed in Table N-4 (which are supported in the CCITT version of ISUP services).

Table N-4. Unsupported Message Parameters

Message Parameter	Description
CLLHISTORY	Call History Information
CALREF	Call Reference
CONREQ	Connection Request
ECHOCNTRL	Echo Control
FREEPHIND	Freephone Indicator
MCIDREQ	MCID Request Indicator
MCIDRSP	MCID Response Indicator
PROPDLTCTR	Propagation Delay Counter
TRANNETSEL	Transit Network Selection

Parameter Field Values

In addition to the codes defined in ITU-T Q.763 for *Calling Party Category*, the Taiwanese version of ISUP services defines code `00000110` as **Language Chinese (Mandarin)** and code `00001001` as **National Operator**. Although these codes are currently defined as `CAT_ADMIN1 and CAT_NATOPR`, respectively in `issl_sit.h`, the `teCagPrtyCatEnums7` table lists the Taiwanese-specific parameter values. These values are listed in Table N-5.

Table N-5. Mandatory Parameter Values

Message Parameter	Value	Description
CAT_UNKNOWN	0x00	Unknown at this time.
CAT_OPLANGFR	0x01	French
CAT_OPLANGENG	0x02	English
CAT_OPLANGGER	0x03	German
CAT_OPLANGRUS	0x04	Russian
CAT_OPLANGSP	0x05	Spanish
CAT_ADMIN1	0x06	Chinese (Mandarin)
CAT_ADMIN2	0x07	Admin
CAT_ADMIN3	0x08	Admin
CAT_NATOPR	0x09	National Operator
CAT_ORD	0x0a	Ordinary Calling Subscriber
CAT_PRIOR	0x0b	Calling Subscriber with Priority
CAT_DATA	0x0c	Data Call
CAT_TEST	0x0d	Test Call
CAT_PAYPHONE	0x0f	Payphone

Differences in Protocol Procedures

There are no differences in the procedures for handling unrecognized messages and parameters between the Taiwanese and CCITT versions of ISUP services.

Disabled Options in Taiwan ISUP Services

In the Taiwanese version of ISUP services, only the following protocol option, which is included in the CCITT version of ISUP services, is disabled:

- Automatic transmission of an Unequipped Circuit Identification Code (UCIC) message in reply to a message received for an unequipped circuit. The SINAP/SS7 system discards messages it receives for unequipped CICs.

ISUP Services Timers

The timers used in the Taiwanese version of ISUP services are described in the ME 0001-3 specification. The default values for these timers are not automatically initialized by the SINAP program `gen_stat`. Instead, the System Administrator is required to manually set the correct default timer values according to the information provided in the following tables **prior** to the SINAP node being started.

The timers listed in Table N-6 can be viewed and modified using the MML commands `DISPLAY-SYSTAB` and `CHANGE-SYSTAB` or by using the Terminal Handler (`sysopr`) menu options (Network Commands, Change Commands, and Change System Table).

Table N-6. Timers Accessible via `sysopr`

Symbol	Default Value	Acceptable Range
`T1`	15 seconds	15 to 60 seconds
`T2`	3 minutes	3 minutes
`T5`	5 minutes	5 to 15 minutes
`T6`	1 minute	1 to 2 minutes
`T7`	20 seconds	20 to 30 seconds
`T8`	10 seconds	10 to 15 seconds
`T9`	3 minutes	1.5 to 3 minutes
`T12`	15 seconds	15 to 60 seconds
`T13`	5 minutes	5 to 15 minutes
`T14`	15 seconds	15 to 60 seconds
`T15`	5 minutes	5 to 15 minutes
`T16 (T1)`	15 seconds	15 to 60 seconds
`T17 (T5)`	5 minutes	5 to 15 minutes
`T18`	15 seconds	15 to 60 seconds
`T19`	5 minutes	5 to 15 minutes
`T20`	15 seconds	15 to 60 seconds
`T21`	5 minutes	5 to 15 minutes
`T22`	15 seconds	15 to 60 seconds
`T23`	5 minutes	5 to 15 minutes
`T27`	4 minutes	4 minutes

Table N-6. Timers Accessible via `sysopr` *(Continued)*

Symbol	Default Value	Acceptable Range
`T33`	12 seconds	12 to 15 seconds
`T34`	2 seconds	2 to 4 seconds
`T36`	10 seconds	10 to 15 seconds

The timers listed in Table N-7 are required but are **not** supported by `sysopr`.

Table N-7. Timers Not Accessible via `sysopr`

Symbol	Default Value	Acceptable Range	Initiated When...
`T10`	4 seconds	4 to 6 seconds	The last digit has been received.
`T11`	20 seconds	15 to 20 seconds	The last address message has been received.
`T24`	1.9 seconds	< 2 seconds	The check tone has been sent.
`T25`	5 seconds	1 to 10 seconds	The initial continuity check fails.
`T26`	2 minutes	1 to 3 minutes	The second or any subsequent continuity checks fail.
`T28`	10 seconds	10 seconds	The Circuit Group Query (CQM) message has been sent.
`T29`	300 milliseconds	300 to 600 milliseconds	A congestion indication is received.
`T30`	5 seconds	5 to 10 seconds	A congestion indication is received and timer T29 is not running.
`T35`	15 seconds	1 to 20 seconds	The latest digit (<>ST) has been received and before either the minimum or fixed number of digits have been received.
`T38`	3 minutes	2 to 3 minutes	The incoming international exchange sends to the preceding exchange a Suspend (SUS) message.

Appendix O
Q.767 Version of ISUP Services

The Q.767 version of ISUP services adheres to the following standards:

- CCITT Recommendation Q.767, “Specifications of Signalling System No. 7, Application of the ISDN User Part of CCITT for International ISDN Interconnections”, February 1991
- CCITT Recommendations Q.761 through Q.764, “Specifications of Signaling System No. 7. ISDN User Part, Volume VI, Fascicle VI.8”, November 1988
- ITU-T Recommendations Q.761 through Q.764, “Specifications of Signaling System No. 7. ISDN User Part, Volume VI, Fascicle VI.8”, March 1993

Implementing the Q.767 Version of ISUP Services

To implement the Q.767 version of ISUP services, uncomment the environment variable `ISUP_FEATURE` in the SINAP environment file (`$SINAP_HOME/Bin/sinap_env.[csh` or `sh]`) and set the variable to `Q767`.

```
ISUP_FEATURE=Q767
```

NOTE

The CCITT network variant must be configured on the SINAP node in order to activate the Q.767 version of ISUP services on that node. You define the network variant to be configured on the node during installation of the SINAP/SS7 software. See the *SINAP/SS7 Installation Guide* (R8060).

Differences Between the Q.767 and CCITT/ITU Versions of ISUP Services

The Q.767 version of ISUP services differs from the CCITT/ITU-T version of ISUP services that the SINAP/SS7 system currently supports, which is based on the 1993 edition of ITU-T Recommendations for ISUP services.

The following sections describe the differences in the message types, message parameters, and parameter field values between these two ISUP versions.

For more detailed information on the differences contained in the Q.767 version of ISUP services, see the documents referenced at the beginning of this appendix.

Message Types

The following sections describe the messages that are and are not supported in the Q.767 version of ISUP services.

Supported Message Types

The Q.767 version of ISUP services supports the message types listed in Table O-1.

Table O-1. Supported Message Types

Message Type	Description
ACM	Address Complete
ANM	Answer
BLA	Blocking Acknowledgment
BLO	Blocking
CCR	Continuity Check Request
CGB	Circuit Group Blocking
CGBA	Circuit Group Blocking Acknowledgment
CGU	Circuit Group Unblocking
CGUA	Circuit Group Unblocking Acknowledgment
CON	Connect
COT	Continuity
CPG	Call Progress
FOT	Forward Transfer
GRA	Circuit Group Reset Acknowledgment
GRS	Circuit Group Reset
IAM	Initial Address
REL	Release
RES	Resume

Table O-1. Supported Message Types *(Continued)*

Message Type	Description
RLC	Release Complete
RSC	Reset Circuit
SAM	Subsequent Address
SUS	Suspend
UBA	Unblocking Acknowledgment
UBL	Unblocking

Unsupported Message Types

The Q.767 version of ISUP services does not support the message types in Table O-2. The SINAP node treats these messages as unrecognized and ignores them.

Table O-2. Unsupported Message Types

Message Type	Description
CFN	Confusion
CMC	Call Modification Completed
CMR	Call Modification Request
CMRJ	Call Modification Reject
CQM	Circuit Group Query
CQR	Circuit Group Query Response
FAA	Facility Accepted
FAC	Facility
FAR	Facility Request
FRJ	Facility Rejected
IDR	Identification Request
INF	Information
INR	Information Request
IRS	Identification Response
LPA	Loopback Acknowledgment

Table O-2. Unsupported Message Types *(Continued)*

Message Type	Description
`NRM`	Network Resource Management
`OLM`	Overload
`PAM`	Pass-Along
`SGM`	Segmentation
`UCIC`	Unequipped Circuit Identification Code
`UPA`	User Part Available
`UPT`	User Part Test
`USR`	User-to-User Information

Message Parameters

This section describes the messages parameters that are and are not supported in the Q.767 version of ISUP services.

Supported Message Parameters

The Q.767 version of ISUP recognizes the following message parameters.

For more detailed information on these parameters, see the documents referenced at the beginning of this appendix.

- Access Transport
- Automatic Congestion Level
- Backward Call Indicators
- Called Party Number
- Calling Party Number
- Calling Party’s Category
- Cause Indicators
- Circuit Group Supervision Message Type Indicator
- Closed User Group Interlock Code
- Connected Number
- Continuity Indicators
- End of Optional Parameters
- Event Information

- Forward Call Indicators
- Nature of Connection Indicators
- Optional Backward Call Indicators
- Optional Forward Call Indicators
- Range and Status
- Subsequent Number
- Suspend/Resume Indicators
- Transmission Medium Requirement
- User Service Information
- User-to-User Indicators
- User-to-User Information

Table O-3 describes the fields and field values which are used by each of the supported message parameters.

Table O-3. Allowable Parameter Values

Message Parameter	Where Used	Field	Allowable Value(s)
Access Transport	ACM[1], ANM[1], CON[1], CPG[1], IAM[1]	[only one]	arbitrary series of bytes
Automatic Congestion Level	REL[1]	[only one]	`ACLVL_LVL1` (=1) `ACLVL_LVL2` (=2)
Backward Call Indicators	ANM[1], CPG[1] ACM[2], CON[2]	Charge Indicator (bits BA)	`CHRG_NOIND` (=0) `CHRG_NOCHRG` (=1) `CHRG_CHRG` (=2)
		Called Party's Status Indicator (bits DC)	`CADSTAT_NOIND` (=0) `CADSTAT_SUBFREE` (=1)
		Called Party's Category Indicator (bits FE)	`CADCAT_NOIND` (=0) `CADCAT_ORDSUBS` (=1) `CADCAT_PAYPHONE` (=2)
		End-to-End Method Indicator (bits HG)	`E2EMTH_NOMETH` (=0)
		Interworking Indicator (bit I)	`INTIND_NOINTW` (=0) `INTIND_INTW` (=1)
		End-to-End Information Indicator (bit J)	`E2EINF_NOINF` (=0)

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Backward Call Indicators (cont.)	ANM[1], CPG[1] ACM[2], CON[2] (cont.)	ISDN User Part Indicator (bit K)	ISUP_NOTUSED (=0) ISUP_USED (=1)
		Holding Indicator (bit L)	HOLD_NOTREQ (=0)
		ISDN Access Indicator (bit M)	ISDNNACC_NOISDN (=0) ISDNACC_ISDN (=1)
		Echo Control Device Indicator (bit N)	ECHODEV_NOTINCL (=0) ECHODEC_INCL (=1)
		SCCP Method Indicator (bits PO)	SCCPMTH_NOIND (=0)
Called Party Number	IAM[2]	Odd/Even Indicator	NMB_EVEN (=0) NMB_ODD (=1)
		Nature of Address Indicator	NATNUM (=3) INTNATNUM (=4)
		Internal Network Number Indicator	INN_ALLOW (=0) INN_NOTALLOW (=1)
		Numbering Plan Indicator	ISDNNUM (=1)
		Address Signal	arbitrary series of bytes (up to 8)
Calling Party Number	IAM[1]	Odd/Even Indicator	NMB_EVEN (=0) NMB_ODD (=1)
		Nature of Address Indicator	INTNATNUM (=4)
		Number Incomplete Indicator	NMB_COMPLTE (=0)
		Numbering Plan Indicator	ISDNNUM (=1)
		Address Presentation Restricted Indicator	PREALLOW (=0) PRESREST (=1)
		Screening Indicator	USRPROV (=1) NETPROV (=3)
		Address Signal	arbitrary series of bytes (up to 8)

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)

Calling Party's Category	IAM[2]	[only one]	`CAT_OPLANGFR (=1)`
			`CAT_OPLANGENG (=2)`
			`CAT_OPLANGGER (=3)`
			`CAT_OPLANGRUS (=4)`
			`CAT_OPLANGSP (=5)`
			`CAT_ADMIN1 (=6)`
			`CAT_ADMIN2 (=7)`
			`CAT_ADMIN3 (=8)`
			`CAT_ORD (=10)`
			`CAT_PRIOR (=11)`
			`CAT_DATA (=12)`
			`CAT_TEST (=13)`
			`CAT_PAYPHONE (=15)`

Cause Indicators	ACM[1], REL[2]	Coding Standard	`CSTD_CCITT (=0)`

		Location	`ILOC_USER (=0)`
			`ILOC_TRANNET (=3)`
			`ILOC_PUBNETRUS (=4)`
			`ILOC_PRIVNETRUS (=5)`
			`ILOC_INTNET (=7)`
			`ILOC_NETINTER (=10)`

		Cause Value	`CCUNALLOC (=1)`
			`CCNORTTODEST (=3)`
			`CCSENDSPCLTONE (=4)`
			`CCCALLCLR (=16)`
			`CCUSRBSY (=17)`
			`CCNOUSRRSP (=18)`
			`CCNOANSWR (=19)`
			`CCCALLRJT (=21)`
			`CCNMBECHG (=22)`
			`CCDESTOUTORD (=27)`
			`CCADDRINCOMP (=28)`
			`CCFACREJ (=29)`
			`CCNORMUNSPEC (=31)`
			`CCNOCIRCUIT (=34)`
			`CCNETAOL (=38)`
			`CCTMPFAIL (=41)`
			`CCSWTCHCONG (=42)`
			`CCREQUNAVAIL (=44)`
			`CCRESCUNAVAIL (=47)`
			`CCINCBARRDCUG (=55)`
			`CCNOTAUTHBCAP (=57)`
			`CCBCAPUNAVAIL (=58)`
			`CCSERVUNAVAIL (=63)`

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)

Cause Indicators (cont.)	ACM[1], REL[2] (cont.)	Cause Value (cont.)	`CCBCAPNOTIMP (=65)`
			`CCSERVNOTIMP (=79)`
			`CCUNOTMEMBR (=87)`
			`CCINCOMPDEST (=88)`
			`CCINVMSG (=95)`
			`CCTMRRECV (=102)`
			`CCPROTERR (=111)`
			`CCINTWRK (=127)`

Circuit Group Supervision Message Type Indicator	CGB[2], CGBA[2], CGU[2], CGUA[2]	Type Indicator (bits BA)	`MAINT (=0)` `HARDFAIL (=1)`
Closed User Group Interlock Code	IAM[1]	Network Identity (2 bytes, each with 2 digits)	`0` to `9` for each digit
		Binary Code	arbitrary 2 bytes
Connected Number	ANM[1], CON[1]	Odd/Even Indicator	`NMB_EVEN (=0)` `NMB_ODD (=1)`
		Nature of Address Indicator	`ADDR_NOTPRSNT (=0)` `INTNATNUM (=4)`
		Numbering Plan Indicator	`NP_UNK (=0)` `ISDNNUM (=1)`

		Address Presentation Restricted Indicator	`PREASALLOWQ (=0)`
			`PRESREST (=1)`
			`ADDRNOAVAIL (=2)`

		Screening Indicator	`USRPROV (=1)` `NETPROV (=3)`
		Address Signal	arbitrary series of bytes (up to 8)
Continuity Indicators	COT[2]	Continuity Indicator	`CONT_CHKFAIL (=0)` `CONT_CHKSUCC (=1)`

Event Information	CPG[2]	Event Indicator	`EV_ALERT (=1)`
			`EV_PROGRESS (=2)`
			`EV_INBAND (=3)`

		Event Presentation Restricted Indicator	`EVPR_NOIND (=0)`

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
Forward Call Indicators	IAM[2]	National / International Call Indicator (bit A)	`CALL_NAT (=0)` `CALL_INTERNAT (=1)`
		End-to-End Method Indicator (bits CB)	`E2EMTH_NOMETH (=0)`
		Interworking Indicator (bit D)	`INTIND_NOINTW (=0)` `INTIND_INTW (=1)`
		End-to-End Information Indicator (bit E)	`E2EINF_NOINF (=0)`
		ISDN User Part Indicator (bit F)	`ISUP_NOTUSED (=0)` `ISUP_USED (=1)`

		ISDN User Part Preference Indicator (bits HG)	`PREF_PREFAW (=0)`
			`PREF_NOTREQAW (=1)`
			`PREF_REQAW (=2)`

		ISDN Access Indicator (bit I)	`ISDNACC_NONISDN (=0)` `ISDNACC_ISDN (=1)`
		SCCP Method Indicator (bits KJ)	`SCCPMTH_NOIND (=0)`

Nature of Connection Indicators	IAM[2]	Satellite Indicator (bits BA)	`SAT_NONE (=0)`
			`SAT_ONE (=1)`
			`SAT_TWO (=2)`

		Continuity Check Indicators (bits DC)	`CONTCHK_NOTREQ (=0)`
			`CONTCHK_REQ (=1)`
			`CONTCHK (=2)`

		Echo Control Device Indicator (bit E)	`ECHODEV_NOTINCL (=0)` `ECHODEV_INCL (=1)`
Optional Backward Call Indicators	ACM[1], CPG[1]	In-Band Information Indicator (bit A)	`INBND_NOIND (=0)` `INBND_AVAIL (=1)`
		Call Forwarding May Occur Indicator (bit B)	`CAFWD_NOIND (=0)`

Optional Forward Call Indicators	IAM[2]	Closed User Group Call Indicator (bits BA)	`CUG_NONCUG (=0)`
			`CUG_ACCALLOW (=2)`
			`CUG_ACCNOTALLOW (=3)`

		Connected Line Identity Request Indicator (bit H)	`0` or `1`

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)

Range and Status	CGB[2], CGBA[2], CGU[2], CGUA[2], GRS[2], GRA[2]	Range
		
		(in CGB, CGBA, CGU, CGUA, GRA)
		
		(in GRS, GRA)
			
			`1` through `255`
			
			`1` through `31`

		Status	arbitrary series of bits
Subsequent Number	SAM[2]	Odd/Even Indicator	`NMB_EVEN (=0)` `NMB_ODD (=1)`
		Address Signal	arbitrary series of bytes (up to 8)
Suspend/Resume Indicators	SUS[2], RES[2]	Suspend/Resume Indicator (bit A)	`SR_ISDNSUBINIT (=0)` `SR_NETINIT (=1)`

Transmission Medium Requirement	IAM[2]	[only one]	`TMR_SPEECH (=0)`
			`TMR_64KBITS (=2)`
			`TMR_31KHZ (=3)`

User Service Information	IAM[1]	Coding Standard	`CSTD_CCITT (=0)`
			`CSTD_INT (=1)`
			`CSTD_NAT (=2)`
			`CSTD_NET (=3)`

		Information Transfer Capability	`ITC_SPEECH (=0)`
			`ITC_UNRDIG (=8)`
			`ITC_RESDIG (=9)`
			`ITC_A31KHZ (=16)`
			`ITC_A7KHZ (=17)`
			`ITC_VIDEO (=24)`

		Transfer Mode	`TM_CIRCUIT (=0)` `TM_PACKET (=2)`

		Information Transfer Rate	`ITR_PKT (=0)`
			`ITR_64KBIT (=16)`
			`ITR_2X64KBIT (=17)`
			`ITR_384KBIT (=19)`
			`ITR_1536KBIT (=21)`
			`ITR_1920KBIT (=23)`

		Note 3	
User-to-User Indicators	ACM[1], CON[1]	Type (bit A)	`TYPE_UUSRP (=1)`

Table O-3. Allowable Parameter Values ***(Continued)***

Message Parameter	Where Used	Field	Allowable Value(s)
User-to-User Indicators (con.t)	ACM[1], CON[1] (cont.)	Service 1 (bits CB)	`0` for `no information`
		Service 2 (bits ED)	`0` for `no information`
		Service 3 (bits GF)	`0` for `no information`
		Network Discard Indicator (bit H)	`NETDCRD_UUIDCRD (=1)`
User-to-User Information	ACM[1], ANM[1], CON[1], CPG[1], IAM[1], REL[1]	[only one]	arbitrary series of bytes (up to 8)

NOTES

1. Parameter is optional in this message.
2. Parameter is mandatory in this message.
3. Several additional fields are defined in optional extension bytes per Q.931.

Unsupported Message Parameters

The Q.767 version of ISUP services does not recognize the following list of message parameters:

- Access Delivery Information
- Call Diversion Information
- Call History Information
- Call Modification Indicators
- Call Reference
- Circuit State Indicator
- Connection Request
- Echo Control Information
- Facility Indicator
- Generic Digits
- Generic Notification
- Generic Number

- Information Indicators
- Information Request Indicators
- Location Number
- MCID Request Indicator
- MCID Response Indicator
- Message Compatibility Information
- MLPP Precedence
- Network Specific Facilities
- Original Called Number
- Origination ISC Point Code
- Parameter Compatibility Information
- Pass-Along Information
- Propagation Delay Counter
- Redirecting Number
- Redirection Information
- Redirection Number
- Redirection Number Restriction
- Remote Operations
- Service Activation
- Signalling Point Code
- Transit Network Selection
- Transmission Medium Requirement Prime
- Transmission Medium Used
- User Service Information Prime
- User Teleservice Information

Message Parameter Values

There are a number of cases in the Q.767 version of ISUP services where messages parameters are only recognized in certain message types and not in others. These cases are contained in Table O-4.

Table O-4. Parameter Differences

Message	Parameters
Address Complete (ACM)	Backward Call Indicators[1] Optional Backward Call Indicators Cause Indicators User-to-User Indicators User-to-User Information Access Transport End of Optional Parameters[2]
Answer (ANM)	Backward Call Indicators User-to-User Information Connected Number Access Transport End of Optional Parameters[2]
Call Progress (CPG)	Event Information[1] Backward Call Indicators Optional Backward Call Indicators Access Transport User-to-User Information End of Optional Parameters[2]
Connect (CON)	Backward Call Indicators[1] Connected Number User-to-User Indicators User-to-User Information Access Transport End of Optional Parameters[2]
Forward Transfer (FOT)	End of Optional Parameters[2]
Initial Address (IAM)	Nature of Connection Indicators[1] Forward Call Indicators[1] Calling Party's Category[1] Transmission Medium Requirement[1] Called Party Number[1] Calling Party Number Optional Forward Call Indicators Closed User Group Interlock Code User-to-User Information Access Transport

Table O-4. Parameter Differences*(Continued)*

Message	Parameters
Initial Address (IAM) (cont.)	User Service Information End of Optional Parameters[2]
Release (REL)	Cause Indicators[1] User-to-User Information Automatic Congestion Level End of Optional Parameters[2]
Release Complete (RLC)	End of Optional Parameters[2]
Resume (RES)	Suspend/Resume Indicators[1] End of Optional Parameters[2]
Suspend (SUS)	Suspend/Resume Indicators[1] End of Optional Parameters[2]

NOTES

1. These message parameters are mandatory.
2. "End of Optional Parameters" implies that the corresponding message type must contain a pointer to the start of the optional part. If no optional part exists, the pointer is zero.

Differences in Message Parameter Field Values

The parameter field values contained in Table O-5 are specified for the Q.767 version of ISUP services, and are different than those defined in the 1988 edition of ITU-T Recommendation Q.763 for ISUP services.

Table O-5. Differences in Parameter Field Values Between Q.767 and Q.763

Parameter	Field	Allowable Value(s)
Backward Call Indicators	Called Party's Status Indicator	`CADSTAT_NOIND (=0)` `CADSTAT_SUBFREE (=1)`
	End-to-End Information Indicator	`E2EINF_NOINF (=0)`
	End-to-End Method Indicator	`E2EMTH_NOMETH (=0)`
	Holding Indicator	`HOLD_NOTREQD (=0)`
	SCCP Method Indicator	`SCCPMTH_NOIND (=0)`

Table O-5. Differences in Parameter Field Values Between Q.767 and Q.763*(Continued)*

Parameter	Field	Allowable Value(s)
Called Party Number	Nature of Address Indicator	NATNUM (=3) INTNATNUM (=4)
	Numbering Plan Indicator	ISDNNUM (=0)
Calling Party Number	Nature of Address Indicator	INTNATNUM (=4)
	Number Incomplete Indicator	NMB_COMPLTE (=0)
	Numbering Plan Indicator	ISDNNUM (=1)
	Address Presentation Restricted Indicator	PRESALLOW (=0) PRESREST (=1)
	Screening Indicator	USRPROV (=1) NETPROV (=3)
Calling Party's Category	Calling Party's Category Indicator	CAT_OPLANGFR (=1) CAT_OPLANGENG (=2) CAT_OPLANGGER (=3) CAT_OPLANGRUS (=4) CAT_OPLANGSP (=5) CAT_ADMIN1 (=6) CAT_ADMIN2 (=7) CAT_ADMIN3 (=8) CAT_ORD (=10) CAT_PRIOR (=11) CAT_DATA (=12) CAT_TEST (=13) CAT_PAYPHONE (=15)
Cause Indicators	Coding Standard	CSTD_CCITT (=0)
	Location	ILOC_USER (=0) ILOC_TRANNET (=3) ILOC_PUBNETRUS (=4) ILOC_PRIVNETRUS (=5) ILOC_INTNET (=7) ILOC_NETINTER (=10)
	Cause Value	CCUNALLOC (=1) CCNORTTODEST (=3) CCSENDSPCLTONE (=4) CCCALLCLR (=16) CCUSRBSY (=17) CCNOUSRRSP (=18) CCNOANSWR (=19) CCCALLRJT (=21)

Table O-5. Differences in Parameter Field Values Between Q.767 and Q.763*(Continued)*

Parameter	Field	Allowable Value(s)

Cause Indicators (cont.)	Cause Value (cont.)	`CCNOANSWR (=19)`
		`CCCALLRJT (=21)`
		`CCNMBRCNG (=22)`
		`CCDESTOUTORD (=27)`
		`CCADDRINCOMP (=28)`
		`CCFACREJ (=29)`
		`CCNORMUNSPEC (=31)`
		`CCNOCIRCUIT (=34)`
		`CCNETAOL (=38)`
		`CCTMPFAIL (=41)`
		`CCSWTCHCONG (=42)`
		`CCREQUNAVAIL (=44)`
		`CCRESCUNAVAIL (=47)`
		`CCINCBARRDCUG (=55)`
		`CCNOTAUTHBCAP (=57)`
		`CCBCAPUNAVAIL (=58)`
		`CCSERVUNAVAIL (=63)`
		`CCBCAPNOTIMP (=65)`
		`CCSERVNOTIMP (=79)`
		`CCUNOTMEMBR (=87)`
		`CCINCOMPDEST (=88)`
		`CCINVMSG (=95)`
		`CCTMRRECOV (=102)`
		`CCPROTERR (=111)`
		`CCINTRWRK (=127)`

	Diagnostic(s)	See Note.
Connected Number	Nature of Address Indicator	`ADR_NOTPRESNT (=0)` `INTNATNUM (=4)`
	Numbering Plan Indicator	`NP_UNK (=0)` `ISDNNUM (=1)`
	Screening Indicator	`USRPROV (=1)`

Event Information	Event Indicator	`EV_ALERT (=1)`
		`EV_PROGRESS (=2)`
		`EV_INBAND (=3)`

	Event Presentation Restricted Indicator	`EVPR_NOIND (=0)`
Forward Call Indicators	End-to-End Method Indicator	`E2EMTH_NOMETH (=0)`
	End-to-End Information Indicator	`E2EINF_NOINF (=0)`
	SCCP Method Indicator	`SCCPMTH_NOIND (=0)`

Table O-5. Differences in Parameter Field Values Between Q.767 and Q.763*(Continued)*

Parameter	Field	Allowable Value(s)
Optional Backward Call Indicators	Call Forwarding May Occur Indicator	`CAFWD_NOIND (=0)`

Transmission Medium Requirement	Transmission Medium Requirement	`TMR_SPEECH (=0)`
		`TMR_64BITS (=2)`
		`TMR_31KHZ (=3)`

User-to-User Indicators	Type	`TYPE_UURSP (=1)`
	Service 1, Service 2, and Service 3	`No information (=0)`
	Network Discard Indicator	`NETDCRD_UUIDCRD (=1)`

NOTE

Although not supported in the Q.767 version of ISUP, these fields are defined in ITU-T 1988 recommendations for ISUP services.

In the Address Complete (ACM) message (wherein the Cause Indicators parameter is optional), the `MsgElmtDef` structure defines the maximum parameter data length as 2, which results in the parameter being discarded if it contains these unsupported fields.

In the Release (REL) message (wherein the Cause Indicators parameter is mandatory), this parameter is accepted along with the unsupported fields so as not to disrupt normal call handling processes. However, these fields are ignored and not sent as output to the destination system.

Differences in Protocol Procedures

This section describes the differences in protocol procedures between the Q.767 and ITU-T/CCITT (1993) versions of ISUP services currently implemented on the SINAP/SS7 system.

Unrecognized Messages Types and Messages Parameters

The procedures for handling unrecognized message types and message parameters in the Q.767 version of ISUP services are much simpler than those found in the CCITT version of ISUP services.

In the Q.767 version of ISUP services, the Message Compatibility Information or Parameter Compatibility Information message parameters are not supported, nor is the use of the

Confusion (CFN) message type. Therefore the Q.767 version of ISUP services simply discards and ignores unrecognized message types and message parameters.

Default Message Parameter Field Values

The message parameter fields for which a default value will be passed up to the user if the input is not one of the allowable valu8es are listed in Table O-6.

Table O-6. Message Parameter Fields Passing Default Values to User

Parameter	Field
Backward Call Indicators	Charge Indicator Called Party's Status Indicator Called Party's Category Indicator End-to-End Method Indicator* End-to-End Information Indicator* Holding Indicator* SCCP Method Indicator*
Calling Party Number	Address Presentation Restricted Indicator
Calling Party's Category	Calling Party's Category Indicator
Cause Indicators	Coding Standard Location Cause Value
Connected Number	Address Presentation Restricted Indicator
Forward Call Indicators	End-to-End Method Indicator* End-to-End Information Indicator* SCCP Method Indicator*
Nature of Connection Indicators	Satellite Indicator Continuity Check Indicators
Optional Backward Call Indicators	Call Forwarding May Occur Indicator*
Optional Forward Call Indicators	Closed User Group Call Indicator
User-to-User Indicators	Type* Service 1 Service 2 Service 3

NOTE

Default interpretation of input is not performed for the message parameter fields marked with an asterisk (*) because all possible values are allowed.

Capability-Based Protocol Options

The following paragraphs describe the differences in protocol options between the Q.767 and ITU-T/CCITT (1993) versions of ISUP services.

Disabled Protocol Options

The Q.767 version of ISUP services does not support three significant protocol features which are supported in the ITU-T/CCITT (1993) version of ISUP services. These are:

- The automatic sending of an Unequipped Circuit Identification Code (UCIC) message in reply to a message that is received for an unequipped circuit.
- The ability to perform segmentation of overlength messages for certain message types
- Use of the Confusion (CFN) message to reply to unrecognized message types or message parameters

To disable these protocol options, the following capability-based protocol option bits are automatically set whenever the Q.767 version of ISUP services implemented (`ISUP_FEATURE=Q767`).

```
ISUP_OPT_NO_UCIC_REPLIES
ISUP_OPT_NO_SEGEMENTATION
ISUP_OPT_NO_CFN_REPLIES
```

New Protocol Options

The Q.767 version of ISUP services supports a protocol feature which is not available in the ITU-T/CCITT (1993) version of ISUP services. This feature prevents the Recommendation and Diagnostic fields from appearing in the Cause Indicators message parameter on output of a Release (REL) message.

The following protocol option bit is automatically set whenever the Q.767 version of ISUP services implemented (`ISUP_FEATURE=Q767`) to enable this feature.

```
ISUP_OPT_SHORT_CAUSE_IND
```

ISUP Services Timers

The ISUP services timers supported in the Q.767 version of ISUP services are listed in Table O-7.

Table O-7. Supported ISUP Timers

Timer	Description
T1[1]	Release (REL) message sent
T2	Suspend (SUS) (user) message received

Table O-7. Supported ISUP Timers*(Continued)*

Timer	Description
T5[2]	Initial Release (REL) message sent
T6	Suspend (SUS) (network) message received
T7	Latest address message sent
T8	Receipt of Initial Address (IAM) message requiring continuity check
T9	Receipt of Address Complete (ACM) message
T12	Blocking (BLO) message sent
T13	Initial Blocking (BLO) message sent
T14	Unblocking (UBL) message sent
T15	Initial Unblocking (UBL) message sent
T16[1]	Reset Circuit (RSC) message (not due to T5 expiry) sent
T17[2]	Initial Reset Circuit (RSC) message sent
T18	Circuit Group Blocking (CGB) message sent
T19	Initial Circuit Group Blocking (CGB) message sent
T20	Circuit Group Unblocking (CGU) message sent
T21	Initial Circuit Group Unblocking (CGU) message sent
T22	Circuit Group Reset (GRS) message sent
T23	Initial Circuit Group Reset (GRS) sent
T27	Receipt of continuity check failure indication
T36	Continuity recheck timer

NOTES

1. The IST1 timer in the System Table is used for these timers.
2. The IST5 timer in the System Table is used for these timers.

Index

C

D

E

F

G

H

N

O

P

S

T

U