

PGW 2200 Softswitch TCAP Release 9.3 and Later

Document ID: 61183

Introduction

Prerequisites

- Requirements
- Components Used
- Conventions

Background Information

TCAP Resolution

- Sniffer the Ethernet Line
- Platform.log TCAP Trace
- MDL Trace Tool

Appendix A: MDL Tags

Appendix B: Log off SS7 Point Codes

Appendix C: SCCP Message Types

- Unitdata (UDT)
- Unitdata service (UDTS)
- UDTS Return Causes

Appendix D: MDL Interface for TCAP Message

Appendix E: Internal MDL Interface

Related Information

Introduction

Transaction Capabilities Applications Part (TCAP) provides support for interactive applications in a distributed environment. TCAP defines an end-to-end protocol between its users. This may be located in an SS7 network or another network that supports TCAP (IP).

Prerequisites

Requirements

Readers of this document should have knowledge of:

- Cisco Media Gateway Controller Release 9

Components Used

The information in this document is based on the Cisco PGW 2200 Softswitch.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

For more information on document conventions, refer to the Cisco Technical Tips Conventions.

Background Information

The TCAP protocol consists of two sub-layers:

- Component sub-layer
- Transaction sub-layer

The component sub-layer interfaces with the conversion engine. The conversion engine is the equivalent of a service user or subsystem number (SSN). The component sub-layer supports these services:

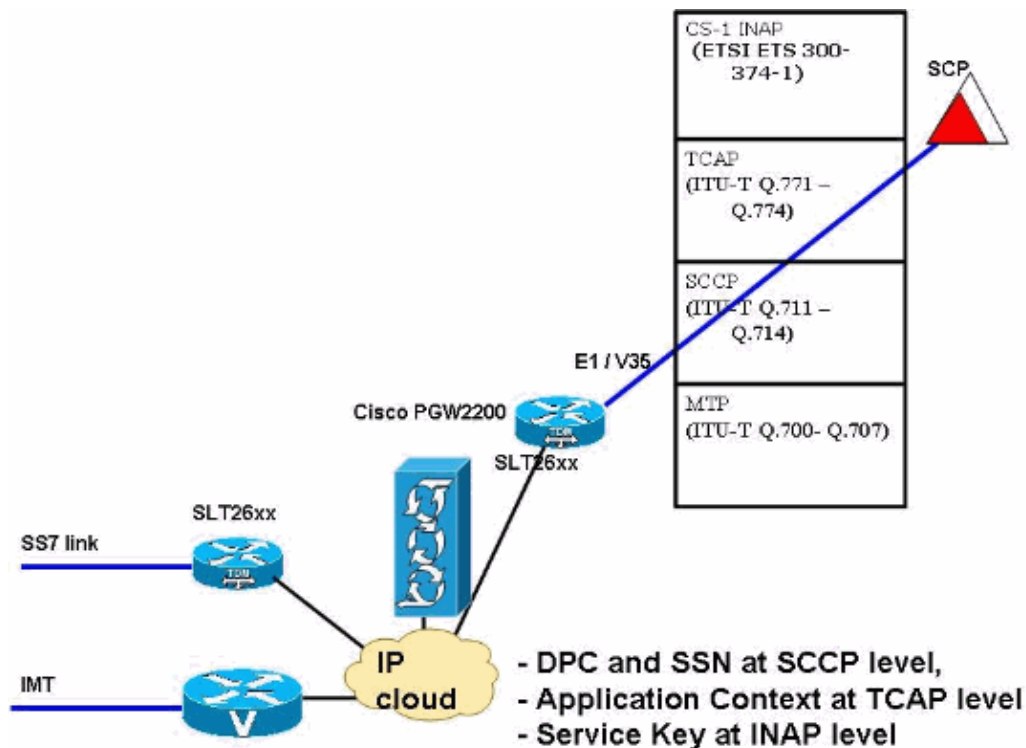
- Association of operations and replies.
- Abnormal situation handling.

The transaction sub-layer interfaces with Signalling Connection Control Part (SCCP). TCAP only supports a connectionless network service. The transaction sub-layer communicates with SCCP through the connectionless interface.

TCAP software uses the services of SCCP software to route the messages to the TCAP user in the destination node. The interface between the TCAP and the SCCP software is tightly coupled. Each TCAP request from the engine contains a global title and destination subsystem number. TCAP provides the subsystem number to SCCP for the Signal Transfer Points (STP) code look-up. If the SS7 addresses and routes are configured correctly and fully operational, troubleshoot the SCCP and TCAP information passed and received between the Cisco PGW 2200 and a remote SCCP or TCAP peer.

The Cisco PGW 2200 uses the SCCP to encapsulate TCAP queries for transport over Message Transfer Part (MTP). This SCCP communication between peers is sent without a connection over MTP. The Cisco PGW 2200 uses the SCCP Unidata (UDT) to send data to the remote SCCP node for connectionless communication. The PGW 2200 receives a valid response when the SCCP UDT message is delivered successfully. This is typically in the form of a UDT message. The exchange of these UDT messages facilitates the connectionless communication between the PGW 2200 and the remote SCCP peer (such as Service Control Point [SCP] for TCAP data base lookups). The PGW 2200 defines an optional field in the UDT that states the SCCP peer should "return on error" the contents of any message it sends to the remote node if the UDT message is undeliverable. The Unidata service (UDTS) message is used to facilitate this error response. The UDTS message indicates to the PGW 2200 that a UDT message received at the remote node (such as STP or SCP) cannot be delivered to the destination.

Cisco PGW 2200 Concept Setup



TCAP Resolution

The SCCP messaging (UDT/UDTS) discussed in the Background Information section is critical when you troubleshoot TCAP services and functionality. Resolve any problems at the SCCP layer before you troubleshoot TCAP data sent or received. The format of the UDT and the UDTS message is shown in Appendix C.

Use these Cisco PGW 2200 tools to debug calls that require the TCAP (TCAP / SCCP) services:

- Sniffer the Ethernet line with tools such as Ethereal, UNIX snoop, and Snooper.
- Platform.log TCAP trace on the PGW 2200.
- MDL Trace Tool for call processing at the protocol level.

Sniffer the Ethernet Line

The Cisco PGW 2200 uses Reliable UDP (RUDP) to send MTP3 and upper layer SS7 messages between the local MTP1 and MTP2 devices (such as a Signaling Link Terminal [SLT]). This communication is typically done over port 7000 on the Cisco PGW 2200 local Ethernet interface. This is configurable. Refer to the configuration guide for details on configuring the PGW "stPort" ports in XECfgParm.dat.

You can use any Ethernet sniffer to view the packets sent between the Cisco PGW 2200 and its local MTP2 control device. However, not all of them support the MTP and SCCP protocol used to display a decoded message. If an Ethernet sniffer is not available to the customer, use the UNIX **snoop** command to troubleshoot. The output of the **snoop** command is not user friendly, but is helpful in a worst case scenario.

An Ethernet sniffer that supports the SS7 protocol stack is preferred. It allows you to decode packets seen on the Cisco PGW 2200 Ethernet interface. An open source sniffer such as Ethereal can also be used and is available online.

If no commercial sniffer utility is available, issue the **snoop** command on the target Cisco PGW 2200 to see the hex data output of the messages sent to and from the Cisco PGW 2200. With root permission on the Cisco

PGW 2200, issue this command to see the hex data sent out of the configured 'stPort.' For additional information on the **snoop** command, refer to the 'snoop man pages' or the SUN Administrative guides.

```
#snoop -d <ethernet device name> -x 42 port <stPort>
```

Issue this command to snoop the packets sent out the Ethernet device, hmeX, on port 7000.

```
#snoop -d hmeX -x 42 port 7000
```

This is example output of captured SS7 packets with the **snoop** command.

```
#snoop -d hme0 -x 42 port 7000
```

```
PGW2200 -> C2600.cisco.com UDP D=7000 S=7000 LEN=96
```

```
0: 4004 dcb5 0000 8000 0001 0000 0010 0000 @.....
```

```
16: 0000 0044 8321 4802 3209 8003 0d11 0a8b ...D.H.2..... ← UDT (09) to SLT from PGW
```

```
32: 2108 3000 1838 3344 4404 c309 0865 2962 !0..83DD...e)b
```

```
48: 2748 0102 6c22 a120 0201 0102 0100 3018 'H..1" .....0.
```

```
64: 8004 0000 0001 8207 0110 1838 3344 4483 .....83DD.
```

```
80: 0701 1107 1311 0010 .....
```

```
PGW2200 -> C2600.cisco.com UDP D=7000 S=7000 LEN=32
```

```
0: 4004 ddb5 0000 8000 0001 0000 0044 0000 @.....D..
```

```
16: 0000 0004 0000 0001 .....
```

```
C2600.cisco.com -> PGW2200 UDP D=7000 S=7000 LEN=144
```

```
0: 4004 b6dd 0000 8000 0001 0001 0045 0000 @.....E..
```

```
16: 0000 0074 0000 001e 0000 0000 0000 0000 ...t.....
```

```
32: 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

```
48: 0000 0000 0000 0000 0000 0000 0000 0000 .....
```

```
0: 4004 b6dd 0000 8000 0001 0001 0045 0000 @.....E..
```

```
16: 0000 0074 0000 001e 0000 0000 0000 0000 ...t.....
```

```

16: 0000 0074 0000 001e 0000 0000 0000 0000  ...t.....
32: 0000 0000 0000 0000 0000 0000 0000 0000  .....
48: 0000 0000 0000 0000 0000 0000 0000 0000  .....
64: 0000 0000 0000 0003 0000 0000 0000 8571  .....q
80: 0000 0000 0000 0002 0000 0000 0000 000a  .....
96: 684f 3338 0000 0000 22b3 e70f0003 598a  hO38....".....Y.
112: 0000 0001 0000 0000 0000 0000 0000 0000  .....
128: 0000 0000 0000 0005  .....

PGW2200 -> C2600.cisco.com UDP D=7000 S=7000 LEN=12

0: 4004 deb6  @...

C2600.cisco.com -> PGW2200  UDP D=7000 S=7000 LEN=96

0: 4004 b7dd 0000 8000 0001 0000 0011 0000  @.....
16: 0000 0044 8309 4808 a20a 0103 0d11 04c3  ...D.H..... ← UDTS (0A) from SLT to PGW
32: 0908 650a 8b21 0830 0018 3833 4444 2962  ...e..!0.83DD)b
48: 2748 0102 6c22 a120 0201 0102 0100 3018  'H..I'.....0.
64: 8004 0000 0001 8207 0110 1838 3344 4483  .....83DD.
80: 0701 1107 1311 0010  .....

```

Cisco's Snooper can also be used (if available) to show the hex dump of the SCCP message. The SCCP message header is decoded but the display of the output is dependant on the version of Snooper chosen. The important point is that the message type is visible and gives an indication as to where to start to troubleshoot the call flow. The hex dump shows that message type 09 is a UDT message and message type 0a is the UDTS service message that indicates an error. The direction of the message flow is also useful since the SS7 PCs are shown. If the rest of the hex dump is shown (depends on the snooper version) it can be used to further decode the SCCP and TCAP portions of message. This is based on the industry standards for SCCP and TCAP.

This is the Snooper output of the UDT SCCP message with TCAP data (to PSTN).

```

15:23:03.847052 1-001-1[02057] 1-004-1[02081] ITU SCCP.-> UDT (09) CGPA=0103TCAPMsgType= Pr:0 Ni:NTL
09 80 03 07 0b 04 c3 21 08 0c 04 c3 09 08 67 52 .....!.....gR
62 50 48 01 1f6b 22 28 20 06 07 00 11 86 05 01 bPH.k"( .....
01 01 a0 15 60 13 80 02 07 80 a1 0d06 0b 2a 81 ...`.....*.
76 82 15 01 01 01 01 00 01 6c 27 a1 25 02 01 01 v.....f.%...
02 01 00 30 1d80 04 00 01 5f91 82 08 83 10 65 ...0....._.....e
27 32 54 76 0f83 07 03 11 03 23 22 11 11 9a 02 '2T∇.....#* ....
20 00

```

If there is an undeliverable SCCP UDT message sent from the Cisco PGW 2200 and / or an SCCP (on the remote node) has problems with the message, the Cisco PGW 2200 receives a UDTS response message. This message indicates a 'return cause' which is very useful in troubleshooting. The UDTS is message type 10 (or 0a hex).

This is an example of a UDTS SCCP message with TCAP data (from PSTN).

Note: This message is an example only and may not reflect an actual query response combination / sequence. The format and amount of information displayed varies depending on the Snooper version.

```

15:23:04.952706 1-004-1[02081] 1-001-1[02057] ITU SCCP.-> UDTS (0a) CGPA=0012TCAPMsgType=0a
Pr:0 Ni:NTL
0a 01 03 0d 11 04 c3 09 08 65 0a 8b 21 08 30 00 .....g.!..v
18 38 33 44 44 29 62 27 48 01 03 6c 22 a1 20 02 etH.P...I.k*(((
01 01 02 01 00 30 18 80 04 00 00 00 01 82 07 01 .....a.....
10 18 38 33 44 55 83 07 01 11 07 13 11 00 10 *.v.....

```

This Snooper output displays the IAM, UDT, UDTS, and REL sequence.

Note: This message is an example only and may not reflect an actual query response combination / sequence. The format and amount of information displayed varies depending on the Snooper version.

```

10:49:37.940189 1-022-1[02225] 1-001-1[02057] ITU ISUP.-> IAM (01) CIC=00010 CDPN=8183334444 CGPN=7031110001
SLS=00 Pr:0 Ni:NTL
10:49:37.962583 1-001-1[02057] 1-004-1[02081] ITU SCCP.-> UDT (09) CGPA=0101TCAPMsgType=
Pr:0 Ni:NTL
10:49:38.034121 1-004-1[02081] 1-001-1[02057] ITU SCCP.-> UDTS (0a) CGPA=0068TCAPMsgType=
Pr:0 Ni:NTL
10:49:38.052539 1-001-1[02057] 1-022-1[02225] ITU ISUP.-> REL (0c) CIC=00010 Cause 31 = Normal, Unspecified
SLS=00 Pr:0 Ni:NTL

```

This is an SS7 sniffer trace that includes SS7 SCCP and TCAP information.

```

-----
SCP(IN)- 19/03/04 18:01:54:223      SCCP      SCP(IN)  UDT      SCP(IN)  BGN  INVK  IDP
-----
Octet001  ITU-T SS7                      Time=19/03/02 18:01:54:223
-----
11010011  BIB/BSN                          1/83
10010110  FIB/FSN                          1/22
..111111  SU type/length                    MSU63
00.....  Spare                             0
-----
Octet004  Service information octet
-----
....0011  Service indicator                  SCCP Signalling Connection Control Part
..00....  Message priority                   0
10.....  Network indicator                 N  National network
-----
Octet005  Routing label
-----
.....    DPC                            10337 SCP(IN)
.....    OPC                            10321
0001....  SLS                               1
-----
Octet009  Message type
-----
00001001  Message type                       UDT  Unitdata
-----
Octet010  SCCP Protocol Class parameter
-----
....0001  Protocol class                     Class 1
0000....  Message handling                   No special options
00000011  Ptr -> Called number               3
00000111  Ptr -> Calling #                   7
00001011  Pointer -> Data                    11
-----
Octet014  SCCP Called Party Address parameter
-----
00000100  Parameter length                   4
.....1   Sgnl pt code bit                   SPC present
.....1   Subsystem # bit                    SSN present
..0000..  Global title ind                   No global title included
.1.....  Routing bit                         DPC and SSN based routing
0.....  Reserved natl use                   0
.....  Point code                         10337 SCP(IN)
00.....  Spare                               0
11111100  Subsystem number                   INAP      IN-CS1+
-----
Octet019  SCCP Calling Party Address parameter
-----
00000100  Parameter length                   4
.....1   Sgnl pt code bit                   SPC present
.....1   Subsystem # bit                    SSN present
..0000..  Global title ind                   No global title included
.1.....  Routing bit                         DPC and SSN based routing
0.....  Reserved natl use                   0
.....  Point code                         10321
00.....  Spare                               0
11111100  Subsystem number                   INAP      IN-CS1+
-----
Octet024  SCCP Data parameter
-----
01100001  Parameter length                   97
01100010  Tag                               BGN Begin, constructor, application-wide
01011111  Length                             95
-----

```

Octet027 Originating Transaction ID

...01000	Tag	Originating Transaction ID
010.....	Class and form	Application-wide, primitive
00000011	Length	3
.....	Originating ID	F30051

Octet032 TCAP Dialogue Portion

...01011	Tag	TCAP Dialogue Portion
011.....	Class and form	Application-wide, constructor
00100011	Length	35

Octet034 TCAP External

...01000	Tag	TCAP External
001.....	Class and form	Universal, constructor
00100001	Length	33

Octet036 Object identifier

...00110	Tag	Object identifier
000.....	Class and form	Universal, primitive
00000111	Length	7
00000000	Organization	itu-t recommendation
00010001	q	Q
.....	773 (X'305)	773
00000001	as(1)	1
00000001	Protocol data unit	dialogue PDU(1)
00000001	version(1)	1
10100000	Single-ASN.1-typeTag	Parameter
00010110	Length	22

Octet047 Dialogue request

...00000	Tag	Dialogue request
011.....	Class and form	Application-wide, constructor
00010100	Length	20

Octet049 Protocol-version

...00000	Tag	Protocol-version
100.....	Class and form	Context-specific, primitive
00000010	Length	2
00000111	Unused Bit	07
.0000000	Unused Bit	00
1.....	Protocol Version	Version 1

Octet053 Application-context-name

...00001	Tag	Application-context-name
101.....	Class and form	Context-specific, constructor
00001110	Length	14

Octet055 Object Identifier

...00110	Tag	Object identifier
000.....	Class and form	Universal, primitive
00001100	Length	12
00101010	Protocol	ccitt identified-organization
10000110	SubProtocol	etsi
00111010	Domain	inDomain
00000000	Network	in-Network
10001001	AC Name	ac (application context)
01100001	Service	cs1-ssp-to-scp(0)
00110011	Version	Reserved

```

..... Contents                01 00 01 00 01
-----
Octet069 TCAP Component Portion
-----
...01100 Tag                    TCAP Component Portion
011..... Class and form        Application-wide, constructor
10000000 Length                128
-----
Octet071 Invoke component
-----
...00001 Tag                    Invoke component
101..... Class and form        Context-specific, constructor
00101111 Length                47
-----
Octet073 Invoke ID
-----
...00010 Tag                    Invoke ID
000..... Class and form        Universal, primitive
00000001 Length                1
00000001 Invoke ID            01
-----
Octet076 Operation Code
-----
...00010 Tag                    Local
000..... Class and form        Universal, primitive
00000001 Length                1
..... Operation Code            IDP InitialDP
-----
Octet079 Parameter Sequence
-----
...10000 Tag                    Parameter Sequence
001..... Class and form        Universal, constructor
00100111 Length                39
-----
Octet081 ServiceKey
-----
...00000 Tag                    ServiceKey
100..... Class and form        Context-specific, primitive
00000001 Length                1
..... Service key              94
-----
Octet084 CalledPartyNumber
-----
...00010 Tag                    CalledPartyNumber
100..... Class and form        Context-specific, primitive
00000111 Length                7
.0000011 Nature of address      National (significant) number( national use )
1..... Odd/even                Odd number of address signals
...0000 Spare                    00
.001.... Numbering plan          ISDN (Telephony) numbering plan (Rec. E.164)
1..... Internal network #        Routing to internal network number not allowed
..... Address signals            999956738
0000.... Filler                  0
-----
Octet093 CallingPartyNumber
-----
...00011 Tag                    CallingPartyNumber
100..... Class and form        Context-specific, primitive
00000111 Length                7
.0000011 Nature of address      National (significant) number( national use )
1..... Odd/even                Odd number of address signals
.....01 Screening Indicator      User provided, verified and passed
....00.. Presentation?          Presentation allowed
.001.... Numbering plan          ISDN (Telephony) numbering plan (Rec. E.164)
0..... Number Incomplete?      Complete
..... Address signals            2199997137

```

0000....	Filler	0

Octet102	CallingPartysCategory	

...00101	Tag	CallingPartysCategory
100.....	Class and form	Context-specific, primitive
00000001	Length	1
00001010	CallngPartyCategory	Ordinary calling subscriber

Octet105	ForwardCallIndicators	

...11010	Tag	ForwardCallIndicators
100.....	Class and form	Context-specific, primitive
00000010	Length	2
.....0	Nat'l/International	Call to be treated as a national call
....00.	End-to-end method	No end-to-end method available
...1...	Interworking	Interworking encountered
...0...	End-to-end info	No end-to-end information available
..1.....	ISUP indicator	ISDN user part used all the way
01.....	ISUP preference	ISDN user part not required all the way
.....1	Orig ISDN access	Originating access ISDN
....00.	SCCP method	No indication
....0...	Spare	0
0000....	ReservedForNat'lUse	0

Octet109	BearerCapability	

...11011	Tag	BearerCapability
101.....	Class and form	Context-specific, constructor
00000101	Length	5

Octet111	Bearer Cap	

...00000	Tag	Bearer Cap
100.....	Class and form	Context-specific, primitive

Octet112	User service information parameter	

00000011	Parameter length	3

Octet113	User service info octet 3	

...00000	Transfer capability	Speech
.00.....	Coding standard	CCITT standardized coding
1.....	Extension bit	1

Octet114	User service info octet 4	

...10000	Transfer rate	64 kbit/s
.00.....	Transfer mode	circuit mode
1.....	Extension bit	1

Octet115	User service info octet 5	

...00011	Layer 1 protocol	Recommendation G.711 A-law
.01.....	Layer 1 Identifier	User information layer 1 protocol
1.....	Extension bit	1

Octet116	CalledPartyNumber	

...00010	Tag	CalledPartyNumber
110.....	Class and form	Private use, primitive
00000010	Length	2
.0000000	Nature of address	Spare
0.....	Odd/even	Even Number of Address signals
...1010	Spare	0A

```

.000.... Numbering plan      Spare (no interpretation)
0..... Internal network #    Routing to internal network number allowed
-----
Octet120 End-of-contents
-----
00000000 Tag                  00
00000000 Length              00
-----
Checksum CRC16..... 0001011001110111 hex=1677
-----
-----
-----

```

SCP(IN)- 19/03/04 18:01:54:269 SCCP SCP(IN) UDT SCP(IN) CON INVK CUE

```

-----
Octet001 ITU-T SS7           Time=19/03/02 18:01:54:269
-----
10000001 BIB/BSN            1/1
10110010 FIB/FSN            1/50
..111111 SU type/length     MSU63
00..... Spare              0
-----
Octet004 Service information octet
-----
....0011 Service indicator   SCCP Signalling Connection Control Part
..00.... Message priority    0
10..... Network indicator    N National network
-----
Octet005 Routing label
-----
..... DPC                    10321
..... OPC                    10337 SCP(IN)
1010.... SLS                 10
-----
Octet009 Message type
-----
00001001 Message type        UDT Unitdata
-----
Octet010 SCCP Protocol Class parameter
-----
....0001 Protocol class      Class 1
0000.... Message handling     No special options
00000011 Ptr -> Called number 3
00000111 Ptr -> Calling #    7
00001011 Pointer -> Data     11
-----
Octet014 SCCP Called Party Address parameter
-----
00000100 Parameter length      4
.....1 Sgnl pt code bit       SPC present
.....1 Subsystem # bit        SSN present
..0000.. Global title ind     No global title included
.1..... Routing bit          DPC and SSN based routing
0..... Reserved natl use      0
..... Point code              10321 Matinha
00..... Spare                 0
11111100 Subsystem number     INAP IN-CS1+
-----
Octet019 SCCP Calling Party Address parameter
-----
00000100 Parameter length      4
.....1 Sgnl pt code bit       SPC present
.....1 Subsystem # bit        SSN present
..0000.. Global title ind     No global title included

```

.1.....	Routing bit	DPC and SSN based routing
0.....	Reserved natl use	0
.....	Point code	10337 SCP(IN)
00.....	Spare	0
11111100	Subsystem number	INAP IN-CS1+

Octet024	SCCP Data parameter	

01001001	Parameter length	73
01100101	Tag	CON Continue, constructor, application-wide
01000111	Length	71

Octet027	Originating Transaction ID	

...01000	Tag	Originating Transaction ID
010.....	Class and form	Application-wide, primitive
00000011	Length	3
.....	Originating ID	7A01B4

Octet032	Destination Transaction ID	

...01001	Tag	Destination Transaction ID
010.....	Class and form	Application-wide, primitive
00000011	Length	3
.....	Destination ID	F30051

Octet037	TCAP Dialogue Portion	

...01011	Tag	TCAP Dialogue Portion
011.....	Class and form	Application-wide, constructor
00101111	Length	47

Octet039	TCAP External	

...01000	Tag	TCAP External
001.....	Class and form	Universal, constructor
00101101	Length	45

Octet041	Object identifier	

...00110	Tag	Object identifier
000.....	Class and form	Universal, primitive
00000111	Length	7
00000000	Organization	itu-t recommendation
00010001	q	Q
.....	773 (X'305)	773
00000001	as(1)	1
00000001	Protocol data unit	dialogue PDU(1)
00000001	version(1)	1
10100000	Single-ASN.1-typeTag	Parameter
00100010	Length	34

Octet052	Dialogue response	

...00001	Tag	Dialogue response
011.....	Class and form	Application-wide, constructor
00100000	Length	32

Octet054	Protocol-version	

...00000	Tag	Protocol-version
100.....	Class and form	Context-specific, primitive
00000010	Length	2
00000111	Unused Bit	07
.00000000	Unused Bit	00
1.....	Protocol Version	Version 1

Octet058	Application-context-name	
...00001	Tag	Application-context-name
101.....	Class and form	Context-specific, constructor
00001110	Length	14
Octet060	Object Identifier	
...00110	Tag	Object identifier
000.....	Class and form	Universal, primitive
00001100	Length	12
00101010	Protocol	ccitt identified-organization
10000110	SubProtocol	etsi
00111010	Domain	inDomain
00000000	Network	in-Network
10001001	AC Name	ac (application context)
01100001	Service	cs1-ssp-to-scp(0)
00110011	Version	Reserved
.....	Contents	01 00 01 00 01
Octet074	Result	
...00010	Tag	Result
101.....	Class and form	Context-specific, constructor
00000011	Length	3
Octet076	Integer	
...00010	Tag	Integer
000.....	Class and form	Universal, primitive
00000001	Length	1
.....	Value	accepted
Octet079	Result-source-diagnostic	
...00011	Tag	Result-source-diagnostic
101.....	Class and form	Context-specific, constructor
00000101	Length	5
Octet081	Dialogue service user	
...00001	Tag	Dialogue service user
101.....	Class and form	Context-specific, constructor
00000011	Length	3
Octet083	Integer	
...00010	Tag	Integer
000.....	Class and form	Universal, primitive
00000001	Length	1
.....	Value	Null
Octet086	TCAP Component Portion	
...01100	Tag	TCAP Component Portion
011.....	Class and form	Application-wide, constructor
10000000	Length	128
Octet088	Invoke component	
...00001	Tag	Invoke component
101.....	Class and form	Context-specific, constructor
00000110	Length	6
Octet090	Invoke ID	

```

-----
...00010  Tag                Invoke ID
000..... Class and form    Universal, primitive
00000001 Length            1
00000001 Invoke ID        01
-----
Octet093  Operation Code
-----
...00010  Tag                Local
000..... Class and form    Universal, primitive
00000001 Length            1
.....    Operation Code    CUE  Continue
-----
Octet096  End-of-contents
-----
00000000 Tag                00
00000000 Length            00
-----
Checksum  CRC16.....          0011010011100010 hex=34E2
-----
-----
-----

```

Troubleshoot Tip: UDTS Return Cause

For a UDTS message, the 'return cause' is the first byte after the message type 0a. This value helps determine why the STP / SCP sends a UDTS error response. If this information is not visible in the sniffer, proceed to the Platform.log TCAP Trace section in order to enable TCAP traces in the Cisco PGW 2200 log.

Platform.log TCAP Trace

MML allows a user to start a TCAP trace that dumps <Trace> messages for the TCAP channel controller into /opt/CiscoMGC/var/log/platform.log. A TCAP trace allows the user to see the TCAP / SCCP messages sent to the SS7 channel controller to route out to the SS7 switch over MTP3. See Appendix E for the message flow of a TCAP query through the PGW 2200 software.

TCAP tracing is started via mml with the **sta-tcap-trc** command. In order to capture the relevant information, enable debug logging for the TCAP and SS7 channel controller.

This is an example of how to enable a TCAP trace:

```

mml> set-log:TCAP-01:debug,confirm

MGC-01 - Media Gateway Controller 2004-03-26 11:17:31.503 EST

M  COMPLD

"TCAP-01"

;

mml> set-log:ss7-i-1:debug,confirm

MGC-01 - Media Gateway Controller 2004-03-26 11:17:40.715 EST

M  COMPLD

"ss7-i-1"

;

mml> sta-tcap-trc

```

MGC-01 - Media Gateway Controller 2004-03-26 11:05:27.040 EST

```
M RTRV

SROF

"TCAP-01"

/* Component already started */

;
```

Note: Debug logging can have an effect on system performance and should not be used in a production environment under high call volume. Please plan your maintenance window accordingly.

TCAP Messages Sent by the Cisco PGW 2200

Once an IN_TRIGGER is sent to the engine, the engine begins to send the message out of the PGW 2200. Information passed down from the protocol level is relayed to the TCAP channel controller. The TCAP portion is sent down to the SCCP channel controller. Also, a log is created in platform.log to indicate a TCAP message was 'transmitted'. From the previous UDT message (shown in the sniffer portion of this document) you can see how the PGW 2200 logs information related to this same message in the platform.log. This platform log matches the data content shown in the Sample SCCP Message Breakdown: Unitdata / Unitdata Service table in Appendix C. From this table, the first value is the data length value (52 hex = 82 decimal). The actual TCAP data portion follows the message length. In the event that sniffer or snoopers is not available, this platform.log can be used to view / debug TCAP and SCCP transactions.

Troubleshoot Tip: If the TCAP message is not sent down to SCCP, there is a problem at the MDL or Engine level. Troubleshoot the MDL trace and look at the Ltrigger and LTriggerRelease signal.

This output shows the PGW 2200 log sending TCAP down stack to SCCP/MTP.

```
Thu Dec 4 15:23:03:837 2003 EST | TCAP (PID 9513) <Trace>
PROT_TRACE_TCAP_PDU_TX: Hex dump of TCAP message transmitted, SSN=103,
LEN=82,
62 50 48 1f 6b 22 28 20 6 7 0 11 86 5 1 1 1 a0 15 60 13 80 2 7 80 a1 d 6 b 2a 81 76 82 15
1 1 1 1 0 1 6c 27 a1 25 2 1 1 2 1 0 30 1d 80 4 0 1 5f 91 82 8 83 10 65 27 32 54 76 f83 7 3
11 3 23 22 11 11 9a 2 20 0
```

After TCAP sends the message to SCCP, the SS7 channel controller plays RECEIVED MSG FROM SCCP and logs the hex representation of the message to indicate receipt of the message. This hex dump includes the SCCP and TCAP portions as shown in this output.

```
Thu Dec 4 15:23:03:846 2003 EST | ss7-i-1 (PID 9518) <Debug>
RECEIVED MSG FROM SCCP ← INDICATES MESSAGE WAS FROM SCCP (TCAP)
```

```
Thu Dec 4 15:23:03:846 2003 EST | ss7-i-1 (PID 9518) <Debug>
<<<< To: 821 from 809 (bytes 98) prior 0 sio 83 sls 8: ← DPC 1-004-1, OPC 1-001-1
```

```
Thu Dec 4 15:23:03:846 2003 EST | ss7-i-1 (PID 9518) <Trace>
PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages 1e0002 1 09 80 03 07 0b 04 c3 21 08 0c 04 c3 09 08
67
52 62 50 48 01 1f 6b 22 28 20 06 07 00 11 86 05 01 01 01 a0 15 60 13 80 02 07 80 a1 0d 06 0b 2a 81 76 82 15 01 01 01 01 00
01 6c 27 a1 25 02 01 01 02 01 00 30 1d 80 04 0 0 01 5f 91 82 08 83 10 65 27 32 54 76 0f 83 07 03 11 03 23 22 11 11 9a 02 20
00
```

Troubleshoot Tips:

- Use the SCCP message format shown in Appendix C to decode the message type, SCCP header information (shown in the output in yellow) and the beginning of the TCAP data (shown in the output in blue). The 1e0002 in the output represents the destination point code from dpc.dat and the SCCP message dump begins immediately after type "1" (beginning with SCCP message type).
- The PGW 2200 logs counter and Alarms for SCCP, TCAP and SS7 events. If measurements are enabled, check the counters for the TCAP message. Also check the SCCP, UDT, and UDTs received and transmitted. Refer to these documents for MGC operational procedures.

- ◆ Managing System Measurements
- ◆ Cisco MGC Measurements
- ◆ Retrieving TCAP Transactions

- If the SS7 channel controller does not receive the message sent out of the PGW 2200, verify that TCAP transmitted a message down to SCCP. If the TCAP layer transmits the message down, it can be because the SCCP does not have enough information to build the proper SCCP message. This may also be an indication that the SS7 subsystem is not provisioned properly or is not available. Check this list to verify:

- ◆ SS7 Point code configuration and status
- ◆ SS7 Subsystem configuration
- ◆ SS7 Subsystem routing configuration
- ◆ Local and Remote SSN status
- ◆ IN Service configuration (trigger.dat)

System Verification

```
mml>rtrv-spc:all
```

```
MGC-01 - Media Gateway Controller 2004-03-26 13:22:05.492 EST
```

```
M RTRV
```

```
"ss7svc1:DPC=001.022.001,DNW=2:OPC=001.001.001:IS"
```

```
"ss7svc2:DPC=001.022.002,DNW=2:OPC=001.001.001:IS"
```

```
"itussn1:DPC=001.004.001,DNW=2:OPC=001.001.001:IS"
```

```
"itussn2:DPC=001.003.001,DNW=2:OPC=001.001.001:IS"
```

```
"itussn3:DPC=001.004.001,DNW=2:OPC=001.001.001:IS"
```

```
;
```

```
mml> prov-rtrv:ss7subsys:NAME="itussn1"
```

```
MGC-01 - Media Gateway Controller 2004-03-26 11:48:26.321 EST
```

```
M RTRV
```

```
"session=fix551tgp:ss7subsys"
```

```
/*
```

```
NAME = itussn1
```

```
DESC = pc_ssn rte-ssn 48
```

```
SVC = scp1
```

```
PRI = 1
```

```

MATEDAPC =

LOCALSSN = 101

PROTO = SS7-ITU

STPSCPIND = 1

TRANSPROTO = SCCP

OPC = opc1

SUAKEY =

REMOTESSN = 48

    */

    ;

mm1> rtrv-lssn:all

    MGC-01 - Media Gateway Controller 2004-03-26 11:49:01.985 EST

M RTRV

    "TCAP-01:SSN=12,PST=IS"

    "TCAP-01:SSN=101,PST=IS"

    "TCAP-01:SSN=102,PST=IS"

    ;

mm1> rtrv-rssn:all

    MGC-01 - Media Gateway Controller 2004-03-26 11:49:04.695 EST

M RTRV

    "scpl:PC=001.004.001,SSN=12,PST=IS"

    "scpl:PC=001.004.001,SSN=48,PST=IS"

    ;

mm1> prov-rtrv:inservice:name="finap-initdp"

    MGC-01 - Media Gateway Controller 2004-03-29 14:45:25.738 EST

M RTRV

    "session=fix55ltgp:inservice"

    /* NAME = finap-initdp

SKORTCV = 90001

GTORSSN = ROUTEBYSSN

GTFORMAT = NOGT

MSNAME = finap-initdp

    */

```

```

;

mm1> prov-rtrv:SS7ROUTE:NAME="route4"

MGC-01 - Media Gateway Controller 2004-03-30 11:53:08.493 EST

M RTRV

"session=fix551tgp:SS7ROUTE"

/*

NAME = route4

DESC = rte to 1.4.1 scp1

OPC = opcl

DPC = scp1

LNKSET = ls3

PRI = 1

*/

;

```

- If all of this information appears to be correct (as shown in the output displayed above) verify the tagged values sent down from the TCAP protocol level such as the SSN, SCCPCalledParty address and / or SCCPCallingParty address.

TCAP Messages that enter the Cisco PGW 2200

The reverse logic can be used to trace an SS7 message that comes into the Cisco PGW 2200 that is destined to the TCAP / SCCP user layer of the SS7 stack. The PGW 2200 logs show the SS7 message that comes into the SS7 channel controller (from the SS7 line) and is sent to TCAP for processing. The message is broken down at each layer of the SS7 stack. Also, note the OPC/DPC, Service Indicator (SIO) and signaling link selection (SLS). The OPC and DPC is represented in ITU format (in this example only).

Troubleshoot Tip: Verify the message type received from the SS7 line. If a UDTS message is receive check the 'return cause'.

This output shows the PGW 2200 log when it receives SCCP messages from the SS7 line:

Thu Dec 4 15:23:04:953 2003 EST | ss7-i-1 (PID 9518) <Debug>
CP Received PDU from ssetId 3, chan 0

Thu Dec 4 15:23:04:953 2003 EST | ss7-i-1 (PID 9518) <Trace>
PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP
messages 1d0005 0 CP DATA IND len: 139 data: 83 09 48 08 02 09 ←msgtype 09= UDT

Thu Dec 4 15:23:04:953 2003 EST | ss7-i-1 (PID 9518) <Debug>
>>>> from: 821 to opc 809 (bytes 134) sio 83 sls 0: ← OPC 1-004-1, DPC 1-001-1

Thu Dec 4 15:23:04:953 2003 EST | ss7-i-1 (PID 9518) <Trace>
PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages
1e0002 0 09 ffff80 03 07 0b 04 ffff83 09 08 67 04 ffff83 21 08 0c 7...<continues>

Thu Dec 4 15:23:04:953 2003 EST | ss7-i-1 (PID 9518) <Debug>
RECEIVED SCCP STACK MSG
<lines omitted>

Thu Dec 4 15:23:04:954 2003 EST | TCAP (PID 9513) <Trace>
PROT_TRACE_TCAP_PDU_RX: Hex dump of TCAP message received, SSN=103, LEN=118,
65 74 48 4 50 0 0 0 49 1 1f6b 2a 28 28 6 7 0 11 86 5 1 1 1 a0 1d61 1b a1 d 6 b
2a 81 76 82 15 1 1 1 1 0 1 a2 3 2 1 0 a3 5 a1 3 2 1 1 6c 3d a1 17 2 1 4 2 1 17 30 fa0 d
30 b 80 1 a 81 1 0 a2 3 80 1 1 a1 22 2 1 5 2 1 23 30 1a 80 10 30 e a0 c a0 a a1 5 a0 3 81
1 6 82 1 a 81 1 1 a2 3 80 1 1

Troubleshoot Tip: Use the SCCP message format shown in Appendix C to decode the message type, SCCP header information (shown in the output in yellow) and the start of the TCAP data. The 1e0002 in the output above represents the calling address (OPC) for the message received at the PGW as represented in dpc.dat. The SCCP message dump begins immediately after the "0" (beginning with SCCP message type).

This output is from the PGW 2200 log when it receives UDTs TCAP over SCCP/MTP:

Thu Mar 25 18:35:35:385 2004 EST | ss7-i-1 (PID 27288) <Debug>
CP Received PDU from ssetId 3, chan 0

Thu Mar 25 18:35:35:385 2004 EST | ss7-i-1 (PID 27288) <Trace>

PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages 1d0005 0
CP DATA IND len: 68 data: 83 09 48 08 a2 0a

Thu Mar 25 18:35:35:385 2004 EST | ss7-i-1 (PID 27288) <Debug>

>>>> from: 821 to opc 809 (bytes 63) sio 83 sls a:

Thu Mar 25 18:35:35:385 2004 EST | ss7-i-1 (PID 27288) <Trace>

PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages 1e0002 0 0a 01 03 0d
11 04 ffffffff83 09 08 65 0a ffffffff8b 21 08 30 00 18 38 33 44 44 29 62
27 48 01 02 6c 22 ffffffff81 20 02 01 01 02 01 00 30 18 ffffffff80 04 00 00
00 01 ffffffff82 07 01 10 18 38 33 44 44 ffffffff83 07 01 11 07 13 11 00 10

Thu Mar 25 18:35:35:385 2004 EST | TCAP (PID 27283) <Debug>

Got 91 bytes from fifo /tmp/sccp_input (fd=16)

Thu Mar 25 18:35:35:385 2004 EST | ss7-i-1 (PID 27288) <Debug>

RECEIVED SCCP STACK MSG

!--- Indicates message is from MTP(SS7 stack).

!--- Lines omitted.

Thu Mar 25 18:35:35:385 2004 EST | TCAP (PID 27283) <Debug>

```
00 01 00 01 1E 00 15 00 00 00 1A 00 00 02 00 00 00 00 00 08 21 00 00
08 09 FFF0A 0A 01 03 0D 11 04 FFF09 08 65 0A FFF21 08 30 00 18 38 33 44
44 29 62 27 48 01 02 6C 22 FFF20 02 01 01 02 01 00 30 18 FFF04 00 00 00
01 FFF07 01 10 18 38 33 44 44 FFF07 01 11 07 13 11 00 10
```

Thu Mar 25 18:35:35:386 2004 EST | TCAP (PID 27283) <Debug>

ioTcSuIntfc::handleNotInd: **Cause =1**

Thu Mar 25 18:35:35:386 2004 EST | TCAP (PID 27283) <Debug>

Calling StUiStuDatReq(), spId = 1

Thu Mar 25 18:35:35:386 2004 EST | TCAP (PID 27283) <Debug>

Deleted spDlgEntry 2-69

Thu Mar 25 18:35:35:386 2004 EST | TCAP (PID 27283) <Debug>

Sending **msgType 15 to Engine**

!--- TCAP sends response to Engine which is translated into L.

This output is from the PGW 2200 log when it receives an invalid TCAP message over SCCP / MTP:

```
Tue Mar 23 16:24:51:565 2004 EST | ss7-i-1 (PID 22997) <Trace>
PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages
1d0005 0 CP DATA IND len: 12 data: 83 09 48 08 02 0a ←msgtype 10= UDTS
```

```
Tue Mar 23 16:24:51:565 2004 EST | ss7-i-1 (PID 22997) <Debug>
>>> from: 821 to opc 809 (bytes 7) sio 83 sls 0:
```

```
Tue Mar 23 16:24:51:565 2004 EST | ss7-i-1 (PID 22997) <Trace>
PROT_TRACE_MTP3_PDU: Hex dump of MTP3 and UP messages
1e0002 0 0a 03 00 00 00 00 ←Msg Type 10 (UDTS), Return cause = 03 =
<lines omitted>
```

```
Tue Mar 23 16:24:51:565 2004 EST | ss7-i-1 (PID 22997) <Debug>
RECEIVED SCCP STACK MSG
<lines omitted>
```

```
Tue Mar 23 16:24:51:566 2004 EST | TCAP (PID 22992) <Debug>
00 01 00 01 1E 00 15 00 00 00 1A 00 00 02 00 00 00 00 00 00 08 21 00 00 08
09 FFF00 0A 03 00 00 00 00 ← OA= dec (10) = UDTS message is
incorrect format missing parameters
```

```
Tue Mar 23 16:24:51:566 2004 EST | TCAP (PID 22992) <Error>
TIOS_ERR_SCCP_SYNTAX_ERR: Syntax error in SCCP switch 1 suId = 0
```

MDL Trace Tool

The Cisco PGW 2200 uses triggers to initiate a TCAP transaction. TCAP protocol transactions use the IN_TRIGGER method to send and receive messages to and from the TCAP control layer. When call analysis hits result type 22, the IN_TRIGGER TCAP protocol is initialized. TCAP information / messages are exchanged between the TCAP protocol layer (for example, triggers written in MDL language) and the Cisco PGW 2200 engine process using a tag, length, and value or TLV syntax. The engine then forwards the information to the TCAP channel controller for further processing.

Use the Cisco PGW 2200 MDL trace to see the data that is sent to and from the TCAP protocol layer to the TCAP controller (via the engine). The TCAP channel controller does the necessary processing on MDL messages received and forwards them to the appropriate IOCC (either TALI-IOCC, IP-IOCC or SS7-IOCC). The engine also converts TCAP message information received from the TCAP channel controller (via SCCP / MTP3) into a TLV format that can be passed to the TCAP protocol layer, also known as IN_TRIGGER. To trace a TCAP call at the protocol level, complete these steps:

1. Start an MDL trace.

```
mml> sta-sc-trc:ss7svc1:log="udts",confirm
```

2. Make a call that triggers a TCAP service (hits analysis result type IN_TRIGGER).
3. Stop the MDL trace.

```
mml> stp-sc-trc:all
```

```
MGC-01 - Media Gateway Controller 2004-03-24 17:41:04.702 EST
```

```
M COMPLD
```

```
"ALL:Trace stopped for the following files:
```

```
../var/trace/udts_ss7svc2_20040324174103.btr
```

4. Run `get_trc` to view the captured MDL trace.

```
get_trc.sh udts_ss7svc2_20040324174103.btr
```

5. Run option **S** to see a 'sim print' of the call that shows the message flow between internal PGW 2200 processes.
6. Run option **D** to see the actual trace of the call through the PGW 2200 code.

Note: The content shown by options **D** and **S** in `get_trc.sh` may not be obvious to understand as the data is shown with internal data types and variable names. However a description of what to look for to debug TCAP transactions is shown in the **MDL Trace Analysis for TCAP** section.

MDL Trace Analysis for TCAP

Use 'sim print' (option **S** of `get_trc.sh`) to view the overall call flow at the Cisco PGW 2200 protocol level. The sim print resembles the one shown in Appendix D. If it does not, try to make a note of where the derived call flow diverges and begin to troubleshoot with that event. For TCAP troubleshooting, focus your attention on one of these events.

- LTrigger
- LTriggerInformation
- LTriggerNext
- LtriggerRelease

These are the internal events that drive the IN_TRIGGER state machine.

Use the Cisco PGW 2200 MDL trace to see the actual code flow for each of these events. LTrigger results in an OUTPUT IN_TRIGGER, and the other three are sent received by IN_TRIGGER by an INPUT IN_TRIGGER message from the engine.

Outgoing TCAP Messages

To identify messages that come in and out of MDL for TCAP, search for IN_TRIGGER in the MDL trace. The Sample IN_TRIGGER Syntax from MDL Trace graphic shows a message sent out and one received into MDL to and from the engine. The OUTPUT indicates that IN_TRIGGER has sent a request for the Engine to forward a TCAP message.

Troubleshoot Tips

- Use the MDL trace to verify that the TRIGGER message was sent to the engine if IN_TRIGGER or OUTPUT was not sent.
- Check the dialplan for the IN_TRIGGER result configuration.
- Check the inservice and / or trigger.dat configuration.
- Verify that the message was sent out of the SS7 channel controller. If the message never made it out of the SS7 channel controller, it is a result of the SCCP channel controller not having enough information to route the call or build a valid message.
- Check the SCCP configuration and SS7_SUBSYSTEM configuration.
- Check the SSN status.
- Check the PC status.

If the output of the IN_TRIGGER is successful, the Cisco PGW 2200 MDL trace displays the response to that message as an INPUT into the IN_TRIGGER.

Sample IN_TRIGGER Syntax from MDL Trace

```

OUTPUT *IN_TRIGGER*: 00 00 00 0e 00 00 00 69 00 01 06 00 01 00 01 01 00 02 00 01 01 00 03 00 07 01 00 00 00 00 00 00 0e 00 01 03 00 0f
00 01 01 00 13 00 0d 02 00 2a b1 76 82 15 01 01 01 01 00 01 00 05 00 01 01 00 06 00 03 01 02 00 00 07 00 01 01 00 09 00 1d 80 04 00 01 5f 91
82 08 83 10 65 27 32 34 76 0f 83 07 03 11 03 23 22 11 11 9a 02 20 00 00 0a 00 00

```

```

INPUT *IN_TRIGGER*: 00 00 00 02 00 00 00 69 00 02 0d 00 12 00 04 00 00 08 21 00 11 00 04 00 00 00 02 00 10 00 12 00 00 00 08 21 0e 01 67
02 04 50 00 00 00 00 00 08 09 00 13 00 0d 03 00 2a 81 76 82 15 01 01 01 01 00 01 00 05 00 01 01 00 06 00 03 01 00 06 00 03 01 00 07 00 01 05 00 09 00 1a 80 10 30 0e a0 0c a0 0a a1 05 a0
0f a0 0d 30 06 80 01 0a 81 01 00 a2 03 80 01 01 00 05 00 01 01 00 06 00 03 01 00 23 00 07 00 01 05 00 09 00 1a 80 10 30 0e a0 0c a0 0a a1 05 a0
03 81 01 06 82 01 0a 81 01 01 a2 03 80 01 01 00 0a 00 00

```

The INPUT message is the response from the engine in reference to the request (or OUTPUT message) sent from the TCAP protocol. The engine can respond on its own behalf or on behalf of the TCAP layer.

The IN_TRIGGER message indicates that MDL sends TCAP / SCCP information down to the engine and channel controllers to be used to construct a UDT message that is sent out on the LINE to the SCP. Information sent down to the engine is derived from the trigger.dat file and it shows directly above the output of this message. To see the content of this message as MDL built it, scroll up from the text IN_TRIGGER. The start of the message building procedure is indicated by SendMessage () & , as shown here.

```

FUNCTION SendMessage() BEGIN

    <messageData>.tagCount = bit(card(<messageData>.DATA), 8) -> '00001011'B

    <messageData>.processId = bit(self, 32) -> '000000000000000000000000000000001101001'B

    <messageData>.callRef = bit(CC.db.essentialData.releaseData.DATA.globalCallRefElem.DATA, 32)
-> '000000000000000000000000

000000000000101'B

    VAR inTable := GetTT(<trigger>, 2) -> 24    ← TRIGGER TABLE in trigger.dat (FINAP Initial DP)

    VAR msTable := GetIN(inTable, 1) -> 24    ← IN Service Index (see figure 9)

    SELECT GetMS(msTable, 3) -> 1            ← Msg type 1 = ITU BEGIN

    OUTPUT Begin TO LINE AS <messageData> -> ELEMENT

    SET TcapTimer := <defaultTimer> -> 5000

... <omitted lines>

    NEXTSTATE <state> -> STATE_WaitResponse

END INPUT

END STATE

ok

```

```

writing message Begin          ← TCAP MESSAGE TYPE

writing element _Begin

  writing field callRef        ← Identifies Call reference for MDL/engine Xaction
    '0000 0000 0000 0000 0000 0000 0000 0010'B

  ok

  writing field processed      ← Identifies process ID for MDL/engine Xaction
    '0000 0000 0000 0000 0000 0000 0110 1001'B

  ok

  writing field msgType       ← Identifies Msg Type for MDL/engine Xaction
    '0000 0000 0000 0001'B    ← Msg type 1 = ITU BEGIN

  ok

  writing field tagCount      ← Identifies number of tags included in this msg
    '0000 1011'B 11 0x0b

  ok

writing field DATA           ← beginning of tags

  writing element TcapTypeElem ← Tag element #1

    writing field DATA       ← Tag element #1 data portion begins

      writing field octet1    ← Tag element #1 field begins

        writing field tcapType ← Tag element #1 field, variable name
          '0000 0001'B 1 0x01    ← Tag #1 VALUE; tcapType = 01

        ok

      ok

    ok

    writing field ielD        ← Tag element #1 TAGID
      '0000 0000 0000 0001'B

    ok

    writing field ieLength    ← Tag element #1 TAG LENGTH
      '0000 0000 0000 0001'B

    ok

  ok

  writing element TcapSystemDestElem ← Tag element #2

```

...

Troubleshoot Tips

- If a TCAP query is sent out of the Cisco PGW 2200 with incorrect data, the MDL trace can be used to see exactly where the Cisco PGW 2200 derived its information. Most of the information comes from the trigger.dat file. To see where the Cisco PGW 2200 derived its information for the outgoing message, search up (from IN_TRIGGER) for the TCAP element in question. For example, if the TCAP type is incorrectly encoded, search for the string `tcapType` in the MDL trace (around the writing field `tcapType`).
- To see where the Cisco PGW 2200 reads trigger.dat to encode TCAP content, search for the strings shown in this table. These strings represent the procedure calls used to retrieve the trigger.dat information. These procedure calls should occur between the INPUT LTrigger event and the OUTPUT IN_TRIGGER message in question.

Name	Description	MDL Search String
TT	Trigger Table Record	GetTT
MA	Message Action Record	GetMA
MS	Message Sending Record	GetMS
OS	Operation Sending	GetOS
PS	Parameter Sending Record	GetPS
RR	Received Response Record	GetRR
MR	Message Receiving Record	GetMR
OR	Operation Receiving	GetOR
PR	Parameter Receiving Record	GetPR
RA	Response Action Record	GetRA
AD	Action Data	GetAD

Incoming TCAP Messages

The INPUT message is the response from the engine in reference to the request. The engine can respond on its own behalf or on behalf of the TCAP layer. The incoming message is identified by the INPUT IN_TRIGGER message string in the Cisco PGW 2200 MDL trace as shown in this example output. This example also shows the message that is decoded. This is helpful if you need to identify any problems that may exist with the TCAP response.

To decode the Engine message received by Cisco PGW 2200 MDL, use the same TLV format described earlier in this document. These message are decoded immediately after the text, INPUT IN_TRIGGER.

```
INPUT  "IN_TRIGGER": 00 00 00 02 00 00 00 69 00 02 0d 00 12 00 04
00 00 08 21 00 11 00 04 00 00 00 02 00 10 00 12 00 00 00 08 21 0c 01 67 02
04 50 00 00 00 00 00

    08 09 00 13 00 0d 03 00 2a 81 76 82 15 01 01 01 01 00 01 00 05 00 01 01
00 06 00 03 01 00 17 00 07 00 01 04 00 09 00 0f a0 0d 30 0b 80 01 0a 81
01 00 a2 03 80 0

1 01 00 05 00 01 01 00 06 00 03 01 00 23 00 07 00 01 05 00 09 00 1a 80
10 30 0e a0 0c a0 0a a1 05 a0 03 81 01 06 82 01 0a 81 01 01 a2 03 80 01
01 00 0a 00 00
```

```
reading element header: TcapMessageStyle
    reading field callRef
!--- Identifies call reference for MDL / engine Xaction.

        '0000 0000 0000 0000 0000 0000 0000 0010'B
    ok
    reading field processed
!--- Identifies process ID for MDL/engine Xaction.

        '0000 0000 0000 0000 0000 0000 0110 1001'B
    ok
    reading field msgType
!--- Identifies message type for MDL/engine Xaction.

        '0000 0000 0000 0010'B
!--- Message type 2 = ITU CONTINUE.

    ok
    reading field tagCount
!--- Identifies the number of tags included in this message.

        '0000 1101'B 13 0x0d
    ok
ok
reading element _Continue
!--- TCAP message type.

    reading field RAW
        1136 bits read
    ok
    reading field DATA
        reading element header: TcapElementStyle
!--- Tag element #1.

            reading field ieId
!--- Tag element #1 TAG ID.
```

```

        '0000 0000 0001 0010'B

ok

        reading field ieLength

!--- Tag element #1 Tag Length.

        '0000 0000 0000 0100'B

!--- 4 bytes.

ok

ok

        reading element TcapDatabaseIdElem

        reading field RAW

        32 bits read

ok

        reading field DATA

!--- Tag element #1 data portion begins.

        '0000 0000'B 0 0x00

!--- Byte 1.

        '0000 0000'B 0 0x00

!--- Byte 1.

        '0000 1000'B 8 0x08

!--- Byte 1.

        '0010 0001'B 33 0x21 "!"

!--- Byte 1.

        ''B

ok

ok

        reading element header: TcapElementStyle

!--- Tag element #2.

        reading field ieId

```

This is sample output of an incoming response to a UDTS message:

```
INPUT "IN_TRIGGER": 00 00 00 02 00 00 00 69 00 0f 02 00 0b
00 01 01 00 0a 00 00
```

```
reading element header: TcapMessageStyle
```

```
reading field callRef
```

```
'0000 0000 0000 0000 0000 0000 0000 0010'B
```

```
ok
```

```
reading field processId
```

```
'0000 0000 0000 0000 0000 0000 0110 1001'B
```

```
ok
```

```
reading field msgType
```

```
!--- Message type - Information message.
```

```
'0000 0000 0000 1111'B
```

```
ok
```

```
reading field tagCount
```

```
'0000 0010'B 2 0x02
```

```
ok
```

```
ok
```

```
reading element _Information
```

```
reading field RAW
```

```
72 bits read
```

```
ok
```

```
reading field DATA
```

```
reading element header: TcapElementStyle
```

```
reading field ieId
```

```
'0000 0000 0000 1011'B
```

```
ok
```

```
reading field ieLength
```

```
'0000 0000 0000 0001'B
```

```
ok
```

```
ok
```

```
reading element TcapErrorElem
```

```
!--- TCAP error element.
```

```

reading field RAW
    8 bits read
ok
reading field DATA
    reading field octet1
        reading field error
            '0000 0001'B 1 0x01
!--- TCAP error element = 01 > TCAP_ERROR_SSN_OOS.

ok
ok
ok
ok
ok
ok
Continuing State Machine: IN_TRIGGER (105)
STATE *
INPUT Information AS <messageData>
    CC.db.nonEssentialData.TCAPTransactionUnixEndTimeElem.DATA
:= MGetTime(CC.db.nonEssentialData.TCAPTransactionMsecEndTimeElem.DATA)
-> 1080257735

```

Another valuable piece of information you can obtain from the Cisco PGW 2200 MDL trace (for TCAP calls) is the LTriggerRelease cause value. The INErrorElem encoded in the LTriggerRelease also provides insight into why a call or TCAP transaction does not work as expected. See this Cisco PGW 2200 MDL graphic that shows a LTriggerRelease that is sent out in response to the initial LTrigger event received by IN_TRIGGER. See Appendix E for details about IN_TRIGGER events and INErrorElem values.

```

OD

END FUNCTION

VAR iNErrElem := NULL

iNErrElem.DATA.error := 42      → TRIG_ERROR_UNKNOWN

INSERT iNErrElem INTO <signalData>

IF (<signalData>::INActionElem = NULL) -> FALSE

FI

OUTPUT LTriggerRelease TO <callingProcess> -> 3 AS <signalData> -> ELEMLIST

NEXTSTATE <state> -> STATE_WaitResponse

END INPUT

END STATE

```

Appendix A: MDL Tags

The Cisco PGW 2200 MDL tags are exchanged between the Cisco PGW 2200 MDL and the engine. This Appendix describes the order, content, and format of all tags used in TCAP transactions. The information used to populate these tag values is obtained from call context and values populated in the trigger.dat file. The trigger file is also used to indicate what should be sent to / from the engine for TCAP message building and what should be received from the engine for TCAP message processing when a response is received.

These tags are used for TCAP call processing:

- **TAG ID 1 TCAP Type**

Description: Indication of the type of TCAP MDL

Data Length: fixed(1)

Data Format:

```

1 = ETSI 300 374-1

2 = Bell Core   GR-1298-CORE
                  TR-NWT-001284
                  TR-NWT-001285

3 = Bell Core   Pre AIN
                  GR-1428-CORE

```

- **TAG ID 2 System Destination**

Description: Internal Destination of event

Data Length: fixed(1)

Data Format: Octet

Contents: 0 = Internal SCP, 1 = Trillium TCAP

• **TAG ID 3 SCCP Called Address**

Description: SCCP data required by trillium

Data Length: Variable

Data Format:

Octet 1 Routing Indicators

Bit A 0 - Route by GT, 1 - Route by SSN

Bit B DPC is present (Octets 2 to 4 have valid data)

Bit C SSN is present (Octet 5 has valid data)

Octet 2 DPC Network

Octet 3 DPC Cluster

Octet 4 DPC Member

Octet 5 Called SSN

Octet 6 GTFormat

0 - No global Title Included

1 - Global Title includes nature of address indicator only (ITU)

- Global title includes translation type,
numbering plan and encoding scheme.(ANSI)

2 - Global Title Includes translation type only.(ITU/ANSI)

3 - Global title includes translation type,
numbering plan and encoding scheme.

(ITU). - not used in ANSI.

4 - Global Title includes translation type, numbering plan,
encoding scheme and nature of address digits.

(ITU). - Not used in ANSI.

Octet 7 Translation Type Value

Octet 8 Numbering Plan

0 - Unknown

1 - ISDN Telephony

2 - Telephony

3 - Data

4 - Telex

5 - Maritime Mobile

6 - Land Mobile

7 - ISDN Mobile

Octet 9 Nature Of Number

1 - Subscriber Number

2 - National Number

3 - International Number

Octet 10 Number Of Digits in octets 11 to 43

Octet 11 to 43

Digits in IA5 format

• **TAG ID 4 SCCP Calling Address**

Description: SCCP data required by trillium

Data Length: Variable

Data Format:

Octet 1 Routing Indicators

Bit A 0 - Route by GT, 1 - Route by SSN
Bit B DPC is present (Octets 2 to 4 have valid data)
Bit C SSN is present (Octet 5 has valid data)

Octet 2 DPC Network

Octet 3 DPC Cluster

Octet 4 DPC Member

Octet 5 Calling SSN

• **TAG ID 5 TCAP Component Type**

Description: Type of TCAP component

Data Length: fixed(1)

Data Format:

Octet
0 = Unknown
1 = Invoke
2 = Return Result Last
3 = Return Error
4 = Reject
5 = Return Result Not Last
6 = Invoke Last
7 = Invoke Not Last

• **TAG ID 6 TCAP Operation Code**

Description: TCAP message operation code

Data Length: Variable (Always 4 for ANSI)

Data Format:

Octet 1 Flag
0 = None
1 = Local
2 = Global
3 = National
4 = Private

Octet 2 Operation Class

Octet 3 Op Code Highest byte (ITU) Family (ANSI)

Octet 4 Op Code Next byte (ITU) Specifier (ANSI)

Octet n Op Code Least byte (ITU)

• **TAG ID 7 TCAP Invoke ID**

Description: ID of the component

Data Length: fixed(1)

Data Format: Octet

• **TAG ID 8 TCAP Correlation ID**

Description: ID of the component that this component correlates to

Data Length: fixed(1)

Data Format: Octet

• **TAG ID 9 TCAP Dialogue Component ANSI**

Description: Body of a TCAP message from first parameter onwards

Data Length: Variable

Data Format: Octet

• **TAG ID 10 TCAP Dialogue End Marker**

Description: Body of a TCAP message from first parameter onwards (SEQUENCE)

Data Length: fixed(0)

Data Format: None

• **TAG ID 11 Error**

Description: Error data

Data Length: fixed(1)

Data Format: Octet

Contents:

- 1 = TCAP_ERROR_SSN_OOS
- 2 = TCAP_ERROR_PC_UNAVAILABLE
- 3 = TCAP_ERROR_SERVICE_NOT_RESPONDING
- 4 = TCAP_TRIGGER_TIMEOUT

• **TAG ID 12 STP–SCP group index**

Description: STP–SCP group index, data passed from analysis.

Data Length: fixed(1)

Data Format: Octet

Contents: STP–SCP group index value.

• **TAG ID 13 TCAP Transport Protocol**

Description: Type of transport protocol

Data Length: fixed(1)

Data Format: Octet

Contents:

- 1 = TCAP_TRANSPORT_SCCP
- 2 = TCAP_TRANSPORT_TCP_IP

• **TAG ID 14 TCAP External Error / Problem**

Description: Error or Problem value received or sent in Error & Result components

Data Length: Variable

Data Format: Octet

• **TAG ID 15 TCAP Body Type**

Description: Type of body of component

Data Length: fixed(1)

Data Format: Octet

Contents:

1 = TCAP_BODY_SEQUENCE

2 = TCAP_BODY_SET

• **TAG ID 16 TCAP Dialog info**

Description: Trillium TCAP includes this TAG in all the messages sent to MDL. MDL should store this information and send it to the Trillium TCAP in all subsequent messages for the dialog or unidirectional messages related to the call.

Data Length: Variable

Data Format: Octet

• **TAG ID 17 TCAP Transaction Id**

Description: Trillium TCAP includes this TAG in all the messages sent to MDL. MDL should store this information for sending to CDB.

Data Length: Variable

Data Format: Octet

• **TAG ID 18 TCAP Database Id**

Description: Trillium TCAP will include this TAG in all the messages sent to MDL. MDL should store this information for sending to CDB.

Data Length: Variable

Data Format: Octet

Appendix B: Log off SS7 Point Codes

ETSI PC 1-1-1 (padded to 16 bits) =
00001000 00001001 = 08 09 = 809 (shown in log)

ETSI PC 1-4-1 (padded to 16 bits) = 00001000 00100001 =
08 21 = 821 (shown in log)

ETSI PC 3-3-3 (padded to 16 bits) 00011000 00011011 =
18 1B = 181b (another ex.)

	Cluster	Network	Member	Point Code
ESTI (14 bits)	3 bits	8 bits	3 bits	14 bits
ANSI (24 bits)	8 bits	8 bits	8 bits	24 bits
PC 1-1-1 (no padding, 14 bit only)	001	000 00001	001	001000 = 8 00000001 = 01
PC 1-4-1 (no padding, 14 bit only)	001	00000100	001	001000 = 8 00100001 =
PC 3-3-3	011	00000011	011	011000 = 18 00011011 = 1B

Appendix C: SCCP Message Types

Message Type	Message Type Code
CR Connection request	0000 0001
CC Connection confirm	0000 0010
CREF Connection refused	0000 0011
RLSD Released	0000 0100
RLC Release complete	0000 0101
DT1 Data form 1	0000 0110
DT2 Data form 2	0000 0111
AK Data acknowledgement	0000 1000
UDT Unitdata	0000 1001
UDTS Unitdata service	0000 1010
ED Expedited data	0000 1011
EA Expedited data acknowledgement	0000 1100
RSR Reset request	0000 1101
RSC Reset confirmation	0000 1110
ERR Protocol data unit error	0000 1111
IT Inactivity test	0001 0000
XUDT Extended unitdata	0001 0001
XUDTS Extended unitdata service	0001 0010
LUDT Long unitdata	0001 0011
LUDTS Long unitdata service	0001 0100

Unitdata (UDT)

The UDT message contains:

- Three pointers
- The parameters indicated in this table.

Parameter	Q.713 reference	Type (F V O)	Length (octets)
Message type	2.1	F	1
Protocol class	3.6	F	1
Called party address	3.4	V	3 minimum
Calling party address	3.5	V	3 minimum
Data	3.16	V	2 - X (Note 1)

Note: Due to the ongoing studies on the SCCP called and calling party address, the maximum length of this parameter needs further study. It is also noted that the transfer of up to 255 octets of user data is allowed when the SCCP called and calling party address do not include global title.

Unitdata service (UDTS)

The UDTS message contains:

- Three pointers.
- The parameters indicated in this table.

Parameter	Q.713 reference	Type (F V O)	Length (octets)
Message type	2.1	F	1
Return cause	3.12	F	1
Called party address	3.4	V	3 minimum
Calling party address	3.5	V	3 minimum
Data	3.16	V	2 - X (Note)

Note: Due to the ongoing studies on the SCCP called and calling party address, the maximum length of this parameter needs further study. It is also noted that the transfer of up to 255 octets of user data is allowed when the SCCP called and calling party address do not include global title.

This table shows a sample SCCP message breakdown for the Unitdata / Unitdata service:

Parameter	Type (F V O)	Length (octets)	Correlation outgoing message	Correlation incoming message

Message type	F	1	09	0a
Protocol class	F	1	80	01
Called party address pointer	F	1	03	03
Calling party address pointer	F	1	07	0d
Data Pointer	F	1	0b	11
Called party address	V	3 minimum	04 c3 21 08	04 c3 & 30
Calling party address	V	3 minimum	04 c3 09 08	18 38 33 44
Data (TCAP DATA)	V	04 c3 09 08 67 18 38 33 44 44 Data (TCAP DATA) V	67 52 62 & 20 00	44 29 62 & 00 10

Note: These messages are examples only and may not reflect an actual query response combination / sequence.

UDTS Return Causes

In the Unitdata service, Extended Unitdata service, or Long Unitdata service message, the "return cause" parameter field is a one octet field that contains the reason for a message return. Bits 1 through 8 are coded as shown here:

Value	Bits	
0	0 0 0 0 0 0 0 0	no translation for an address of such nature
1	0 0 0 0 0 0 0 1	no translation for this specific address
2	0 0 0 0 0 0 1 0	subsystem congestion
3	0 0 0 0 0 0 1 1	subsystem failure
4	0 0 0 0 0 1 0 0	unequipped user
5	0 0 0 0 0 1 0 1	MTP failure
6	0 0 0 0 0 1 1 0	network congestion
7	0 0 0 0 0 1 1 1	unqualified
8	0 0 0 0 1 0 0 0	error in message transport (Note)
9	0 0 0 0 1 0 0 1	error in local processing (Note)
10	0 0 0 0 1 0 1 0	destination cannot perform reassembly (Note)
11	0 0 0 0 1 0 1 1	SCCP failure
12	0 0 0 0 1 1 0 0	hop counter violation
13	0 0 0 0 1 1 0 1	segmentation not supported
14	0 0 0 0 1 1 1 0	segmentation failure
15	0 0 0 0 1 1 1 1	

to

228 1 1 1 0 0 1 0 0 Reserved for International Use
 229 1 1 1 0 0 1 0 1

to

254 1 1 1 1 1 1 1 0 Reserved for National Networks
 255 1 1 1 1 1 1 1 1 Reserved

Appendix D: MDL Interface for TCAP Message

All messages adhere to a common TLV format:

- **Call Instance and ProcessId** – 8 bytes long and should be received by the Engine and returned in the response message from the Engine unaltered.
- **Message ID** – Identifies the message that is sent or received by TCAP protocol layer (values shown in this table).
- **Tagged Id** – Number of tags and tag data (tag ID, data length and data) dictate what is sent out in the TCAP message to the remote destination. All field sizes are fixed except for the data field of a tag item whose length is variable and is defined (in octets) by the data length.

Octet 1–8	Octet 9–10	Octet 11	Octet 12–13	Octet 14–15	Octet 16 (15+n)	Octet (17+n)–(16+n)	Octet (18+n)–(17+n)	...
Call Instance and Process ID	Message ID	Number of tags	Tag ID X	Data length (n)	Data	Tag ID y	Data length (m)	&

Each of the fields Total Length, Call Instance and Process Id, Message Id, Tag Id and Data Length is transmitted by the most significant byte first.

1	ITU Begin
2	ITU Continue
3	ITU End
4	ITU Abort
6	ANSI QueryWithPermission
8	ANSI Response
9	ANSI ConversationWithPermission
99	ANSI ConversationWithOutPermission
17	ANSI Abort
12	ANSI Protocol Abort
11	ANSI User Abort
5	Unidirectional
15	Information
16	Release

Appendix E: Internal MDL Interface

Internally, communication with TCAP State Machine Objects (SMOs) is through signals with data. Any MDL data type can be sent with the signal. The names and meanings of the signals and data are listed here.

- **LTrigger**

Description: This is the first signal that LCM sends to TCAP to start the dialogue. In Elan, INTriggerElem also contains the stpScpGroupIndex.

MSG_ACTION_COPY_STP_SCP_INDEX_FROM_SIGNAL_DATA must be set in the MA table for this to be used.

Components: INTriggerElem, BNumberElem, BNumberDataElem

- **LTriggerInformation**

Description: This signal is sent from TCAP to LCM in response to LTrigger, when the dialogue continues.

Components: INTriggerElem, BNumberElem, BNumberDataElem

- **LTriggerNext**

Description: This signal is sent from LCM to TCAP as a subsequent trigger request in an existing dialogue.

Components: INTriggerElem, BNumberElem, BNumberDataElem

- **LTriggerRelease**

Description: This signal is the last to be sent from either LCM or TCAP and can be sent from TCAP in response to LTrigger after a response has been received from the SCP.

Components: INErrorElem, BNumberElem, BNumberDataElem

INErrorElem has these values :

- 1 TRIG_ERROR_NONE,
- 2 TRIG_EXIT_UNABLE_TO_COMPLETE_MA_IS_LNP_M_BIT_CLEAR,
- 3 TRIG_ERROR_NULL_TRIGGER,
- 4 TRIG_ERROR_TRIGGER_TABLE_NOT_FOUND,
- 5 TRIG_ERROR_UNKNOWN_MESSAGE_ACTION,
- 6 TRIG_ERROR_UNKNOWN_RESPONSE_ACTION,
- 7 TRIG_ERROR_UNKNOWN_PARAMETER_ACTION,
- 8 TRIG_ERROR_MESSAGE_ACTION_FAILED,
- 9 TRIG_ERROR_UNABLE_TO_LOAD_DIALOGUE_COMPONENT,
- 10 TRIG_ERROR_UNABLE_TO_LOAD_TAG,
- 11 TRIG_ERROR_READING_TT,
- 12 TRIG_ERROR_READING_MA,

13 TRIG_ERROR_READING_PS ,
14 TRIG_ERROR_READING_RR ,
15 TRIG_ERROR_READING_PR ,
16 TRIG_ERROR_READING_RA ,
17 TRIG_ERROR_ACTION_NOT_COMPATIBLE_IN_PR ,
18 TRIG_ERROR_NO_ACTION_DATA_FOR_ACTION_RE_TRIGGER ,
19 TRIG_ERROR_NO_ACTION_DATA_FOR_ACTION_SEND_ACTION_TO_LCM ,
20 TRIG_ERROR_UNKNOWN_MESSAGE_IN_MS ,
21 TRIG_ERROR_UNKNOWN_PR_ACTION ,
22 TRIG_ERROR_UNABLE_TO_COMPLETE_MA_COPY_SCCP_GT_FROM_BNUMBER ,
23 TRIG_ERROR_UNABLE_TO_COMPLETE_MA_COPY_STP_SCP_INDEX_FROM_SIGNAL_DATA ,
24 TRIG_ERROR_UNKNOWN_DIALOGUE_COMPONENT ,
25 TRIG_ERROR_SIGNAL_IN_WRONG_STATE ,
26 TRIG_ERROR_SCCP_TIMEOUT ,
27 TRIG_ERROR_IN_RESPONSE_OPERATION_CODE_MISSING ,
28 TRIG_ERROR_IN_RESPONSE_INVOKE_ID_IN_USE ,
29 TRIG_ERROR_IN_RESPONSE_INVOKE_ID_NOT_FOUND ,
30 TRIG_ERROR_IN_RESPONSE_CORROLATION_ID_NOT_FOUND ,
31 TRIG_ERROR_IN_RESPONSE_UNEXPECTED_CORROLATION_ID ,
32 TRIG_ERROR_IN_RESPONSE_NO_COMPONENT_CONTENTS ,
33 TRIG_ERROR_IN_RESPONSE_INVALLID_COMPONENT_CONTENTS ,
34 TRIG_ERROR_IN_RESPONSE_UNEXPECTED_INVOKE_ID ,
35 TRIG_ERROR_IN_RESPONSE_EXTERNAL_ERROR_NOT_FOUND ,
36 TRIG_ERROR_ABORT ,
37 TRIG_ERROR_USER_ABORT ,
38 TRIG_ERROR_PROTOCOL_ABORT ,
39 TRIG_ERROR_UNKNOWN

Related Information

- [Cisco PGW 2200 Softswitch Tech Notes](#)
 - [Voice Technology Support](#)
 - [Voice and Unified Communications Product Support](#)
 - [Recommended Reading: Troubleshooting Cisco IP Telephony](#)
 - [Technical Support & Documentation – Cisco Systems](#)
-

[Contacts & Feedback](#) | [Help](#) | [Site Map](#)

© 2008 – 2009 Cisco Systems, Inc. All rights reserved. [Terms & Conditions](#) | [Privacy Statement](#) | [Cookie Policy](#) | [Trademarks of Cisco Systems, Inc.](#)

Updated: Jul 23, 2008

Document ID: 61183
