

Critical Release Notice

Publication number: 297-2621-380
Publication release: Standard 04.04

The content of this customer NTP supports the
SN06 (DMS) software release.

Bookmarks used in this NTP highlight the changes between the baseline NTP and the current release. The bookmarks provided are color-coded to identify release-specific content changes. NTP volumes that do not contain bookmarks indicate that the baseline NTP remains unchanged and is valid for the current release.

Bookmark Color Legend

Black: Applies to new or modified content for the baseline NTP that is valid through the current release.

Red: Applies to new or modified content for NA017 that is valid through the current release.

Blue: Applies to new or modified content for NA018 (SN05 DMS) that is valid through the current release.

Green: Applies to new or modified content for SN06 (DMS) that is valid through the current release.

Attention!

Adobe® Acrobat® Reader™ 5.0 is required to view bookmarks in color.

Publication History

March 2004

Standard release 04.04 for software release SN06 (DMS).

Change of phone number from 1-800-684-2273 to 1-877-662-5669, Option 4 + 1.

297-2621-380

Digital Switching Systems

UCS DMS-250

Programmable Service Node (PSN)

Application Guide

UCS09 Standard 04.03 August 1999

NORTEL
NETWORKS™

How the world shares ideas.

Digital Switching Systems

UCS DMS-250

Programmable Service Node (PSN) Application Guide

Publication number: 297-2621-380
Product release: UCS09
Document release: Standard 04.03
Date: August 1999

Copyright © 1996, 1997-1999 Northern Telecom,
All Rights Reserved

Printed in the United States of America

NORTEL NETWORKS CONFIDENTIAL: The information contained herein is the property of Nortel Networks and is strictly confidential. Except as expressly authorized in writing by Nortel Networks, the holder shall keep all information contained herein confidential, shall disclose the information only to its employees with a need to know, and shall protect the information, in whole or in part, from disclosure and dissemination to third parties with the same degree of care it uses to protect its own confidential information, but with no less than reasonable care. Except as expressly authorized in writing by Nortel Networks, the holder is granted no rights to use the information contained herein.

Information is subject to change without notice. Nortel Networks reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

DMS, DMS-250, MAP, NORTEL, NORTEL NETWORKS, NORTHERN TELECOM, NT, and SUPERNODE are trademarks of Nortel Networks.

Publication history

August 1999

Standard release 04.03 for UCS09 software release.

Changed title of Chapter 10 from “PSN human machine interface” to “PSN office parameters.”

Added the “What is in this document” section to “About this document.”

November 1998

Standard release 04.02 for software release UCS09.

October 1998

Preliminary release 04.01 for software release UCS09.

August 1998

Standard release 03.02 for software release UCS08.

May 1998

Preliminary release 03.01 for software release UCS08.

September 1997

Standard release 02.02 for software release UCS07 (CSP07).

August 1997

Preliminary release 02.01 for software release UCS07 (CSP07).

October 1996

Preliminary release 01.01 for software release UCS06 (CSP05).

Contents

About this document	xvii
When to use this document	xvii
Intended audience	xviii
What is in this document	xviii
How to check the version and issue of this document	xx
References in this document	xxi
What precautionary messages mean	xxi
How commands, parameters, and responses are represented	xxii
Input prompt (>)	xxiii
Commands and fixed parameters	xxiii
Variables	xxiii
Responses	xxiii
PSN introduction	1-1
Scope of this document	1-1
Background	1-1
Functional overview	1-2
Networking overview	1-2
Identifying a service call	1-3
PSN data communications	1-3
PSN peer-to-peer application protocol - SPI	1-4
SPI parameters	1-4
PSN finite state machine	1-4
PSN admin process	1-4
PSN audit process	1-4
PSN flow control	1-4
PSN office parameters	1-5
PSN logs and OMs	1-5
PSN billing	1-5
PSN data communications	2-1
Functional overview	2-1
Networking overview	2-1
Data communications specification	2-2
Feature activation	2-2
PSN – SCU physical interface	2-3
PSN – SCU protocol stack	2-5
Address resolution	2-10
PSN Restarts	2-14
SCU resets	2-15

- EIU maintenance and link level external node maintenance 2-15
- I/O interface specifications 2-16
- Switch/network upgrade requirements 2-16
- Error handling specifications – message errors 2-16
- Logs – Datafill Problem 2-17
- Logs – SCU Arbitrator Address 2-18
- Engineering hardware information 2-18
 - Ethernet Interface Units 2-18
 - PSN datacom server's UDP Port Number 2-18
 - Multi-Application certification For LPP and FLIS 2-18
- Restrictions and limitations 2-18

PSN messages 3-1

- PSN peer-to-peer application protocol 3-1
 - PSN messaging 3-1
 - Call_Control primitives 3-4
 - Macros 3-7
 - Event notifications 3-8
 - PSN message classifications 3-13
 - PSN message flow 3-15

PSN finite state machine 4-1

- Agent state 4-1
- Connection state 4-2
- The Finite State Machine 4-4
- Call walk through 4-11
- General rules and restrictions for using primitives 4-15
- Feature interactions 4-34
 - Terminating agent bearer capability (BC) screening 4-34
 - New call rejected 4-35
 - New call timeout 4-35
- Error handling 4-38
 - Fatal versus non-fatal errors 4-38
 - Non-context specific protocol errors 4-39
 - Context specific application errors 4-40
- Restrictions and limitations 4-46
 - Warm restarts 4-47
 - No-Restarts SWACTS 4-47
 - Hardware 4-47

PSN flow control 5-1

- Overview 5-1
- Flow control application process 5-1
- Flow control initiation 5-2
 - SCU as a source of flow control 5-2
 - PSN as a source of flow control 5-2
- Flow control behavior at restarts 5-3

Flow control behavior at a heartbeat failure	5-3	
PSN admin process		6-1
<hr/>		
PSN audit process		7-1
Audit process	7-1	
Audit office parameters	7-1	
Audit logs and OMs	7-1	
Audit description	7-2	
<hr/>		
PSN logs		8-1
PSN log summary	8-1	
PSN log description	8-3	
PSN 100	8-3	
PSN 101	8-4	
PSN 102	8-5	
PSN 103	8-6	
PSN 104	8-7	
PSN 105	8-8	
PSN 106	8-9	
PSN 200	8-11	
PSN 201	8-14	
PSN 202	8-15	
PSN 203	8-17	
PSN 204	8-19	
PSN 205	8-20	
PSN 206	8-21	
PSN 207	8-22	
PSN 208	8-24	
PSN 209	8-26	
PSN 212	8-27	
PSN 400	8-28	
PSN 401	8-30	
AUD 599	8-31	
TRKT 214	8-31	
<hr/>		
PSN operational measurements		9-1
OMs	9-1	
OM group summary	9-1	
OM register summary	9-2	
PSN OM and log associations	9-9	
<hr/>		
PSN office parameters		10-1
Office parameters	10-1	
PSN_AUDIT_INTERVAL_TIME	10-3	
PSN_AUDIT_MAX_RETRY	10-3	
PSN_AUDIT_DROP_AGENTS	10-3	
PSN_CALLS_ALLOWED	10-3	
PSN_DROP_AGENTS_SCU_SHELF_RESET	10-3	
PSN_DROP_AGENTS_SCU_SRVC_RESET	10-3	
PSN_DROP_AGENTS_SCU_SYS_RESET	10-4	
PSN_INIT_SCU_POLLING	10-4	

PSN_EVENT_TIMER	10-4
PSN_FLOW_CTRL_MESSAGING	10-4
PSN_INTER_POLL_TIME	10-4
PSN_HEARTBEAT_WAIT_TIME	10-4
PSN_MAX_MEMBER_ADVANCE	10-5
PSN_PERFORM_NEWCALL_DIGCOL	10-5
PSN_PRIMITIVE_NUM_EXT_BLOCKS	10-5
PSN_SCRATCHPAD_NUM_EXT_BLOCKS	10-5
PSN_SPI_LOGS_ON	10-5
Use of existing extension blocks for PSN	10-5
Feature extension blocks	10-6
Tables	10-6
PSNMSGIX	10-7
PSNROUTE	10-8
SCUADDR	10-9
Datafill changes to existing tables for PSN	10-10
PSN billing	11-1
Billing summary	11-1
PSN billing details	11-1
UCS SPI introduction	12-1
Feature synopsis	12-1
Networking overview	12-1
Custom layer protocol	12-2
SPI version 1 for UCS	12-4
SPI version 2 for UCS	12-4
SPI version 3 for UCS	12-6
SPI version 4 for UCS	12-10
UCS PSN parameters version 1	13-1
PSN peer-to-peer application protocol	13-1
PSN parameter definitions	13-1
AccessType	13-13
BearerCapability	13-14
BillingInformation	13-16
BillingInfo Parameter	13-16
CallReferenceID (CRID)	13-18
Calltype	13-19
Controlinfo	13-20
CRID Parameter	13-21
Destinationtrunkgroup	13-22
DigitsCollection	13-23
DigitsCollected	13-24
DigitsOutpulsed	13-28
DigitstoOutpulse	13-29
ErrorCause	13-31
FlowControlInfo	13-32
FlowControlEncountered parameter	13-34
InstructionID	13-35
Instructiontag	13-36

MessageInfo	13-36
MonitorMask	13-38
ParameterID	13-39
PointInCall	13-41
PortInfo	13-43
PortServiceInfo	13-43
PortStatus	13-47
ResetReason	13-49
ServingTranslationScheme	13-50
SessionID	13-51
SignalingInfo	13-51
SigInfo_Mask	13-61
Switch_ID	13-64
Time_of_day	13-65
Tone_Detected	13-66

UCS PSN LOPER messages and parameters version 1 **14-1**

LOPER format	14-1
Messages sent from the SCU to the PSN	14-1
Messages sent from the PSN to the SCU	14-4
Parameters	14-6
PSN LOPER mandatory parameters for primitives	14-8
Bridge mandatory parameters	14-9
Collect Digits mandatory parameters	14-10
Connect mandatory parameters	14-11
Disconnect mandatory parameters	14-11
Error Detected mandatory parameters	14-12
Flow Control mandatory parameters	14-13
Heartbeat mandatory parameters	14-13
Hold mandatory parameters	14-13
Monitor mandatory parameters	14-14
Mute mandatory parameters	14-15
New call accepted mandatory parameters	14-15
New call rejected mandatory parameters	14-16
Play message mandatory parameters	14-17
InstructionTag	14-17
Play prompt collect digits mandatory parameters	14-18
Port status mandatory parameters	14-19
Query port mandatory parameters	14-19
Query time of day (TOD) mandatory parameters	14-20
Reconnect mandatory parameters	14-20
Reset switch mandatory parameters	14-22
Set billing record mandatory parameters	14-23
Set IP address mandatory parameters	14-24
Stop message mandatory parameters	14-24
Transmit signinfo mandatory parameters	14-25
PSN LOPER mandatory parameters for events	14-26
Current time of day (TOD) mandatory parameters	14-26
Digits collected mandatory parameters	14-26
Error detected mandatory parameters	14-28
In service mandatory parameters	14-28

- Instruction completed mandatory parameters 14-29
- Message played mandatory parameters 14-30
- New call mandatory parameters 14-31
- Off-hook mandatory parameters 14-32
- On-hook mandatory parameters 14-33
- Port status mandatory parameters 14-34
- Query port mandatory parameters 14-34
- Route not available mandatory parameters 14-35
- Route selected mandatory parameters 14-36
- Signaling event mandatory parameters 14-37
- Tone detected mandatory parameters 14-38

PRI messages **15-1**

- Alerting message 15-1
 - Alert mandatory parameters 15-1
 - Alert optional parameters (Octet 3–293) 15-2
- Call proceeding 15-5
 - Call proceeding mandatory parameters 15-5
 - Call proceeding optional parameters 15-6
- Connect message 15-7
 - Connect mandatory message parameters 15-7
 - Connect optional parameters 15-11
- Disconnect message 15-13
 - Disconnect mandatory message 15-13
 - Disconnect optional parameters 15-16
- Facility (call associated) message (decoded) 15-21
 - Facility (call associated) mandatory parameters 15-21
 - Facility (call associated) optional parameters 15-21
- Facility (call associated) message (encoded) 15-23
 - Facility (call associated) mandatory parameters 15-24
 - Facility (call associated) optional parameters 15-25
- Progress message 15-29
 - Progress mandatory parameters 15-29
 - Progress optional parameters 15-33
- Release message 15-38
 - Release mandatory message 15-38
 - Release optional parameters 15-41
- Setup message 15-46
 - SETUP mandatory parameters 15-47
 - Setup optional parameters 15-49

UCS PSN parameters version 2 **16-1**

- PSN peer-to-peer application protocol 16-1
 - PSN parameter definitions 16-1
 - Access type 16-14
 - Agent data info 16-15
 - Agent type parameter 16-16
 - Bearer capability 16-17
 - Billing info 16-18
 - Call reference ID (CRID) 16-20
 - Call type 16-21

Control info parameter	16-21
COT required parameter	16-23
CRID Parameter	16-24
Destination trunk group	16-25
Digit collection parameter	16-26
Digits collected	16-28
Digits outpulsed	16-31
Digits to outpulse	16-32
Error cause	16-34
Flow control info parameter	16-35
Flow control encountered parameter	16-37
Info change reason parameter	16-38
Instruction ID	16-38
Instruction tag	16-39
ISUP index	16-40
Message info	16-41
Monitor mask	16-43
Parameter ID	16-44
Point in call	16-46
Port info	16-48
Port service info	16-49
Port status	16-52
Reset reason	16-54
Serving translation scheme	16-55
Session ID	16-56
Signaling info	16-57
Signaling type	16-66
SigInfo mask	16-67
Switch ID	16-70
Time of day	16-71
Tone detected	16-72

UCS PSN LOPER messages and parameters version 2 **17-1**

LOPER format	17-1
Messages sent from the SCU to the PSN	17-1
Messages sent from the PSN to the SCU	17-3
Parameters	17-6
PSN LOPER mandatory parameters for primitives	17-8
Bridge mandatory parameters	17-9
Collect digits mandatory parameters	17-10
Connect mandatory parameters	17-10
Disconnect mandatory parameters	17-11
Error detected mandatory parameters	17-12
Flow control mandatory parameters	17-13
Heartbeat mandatory parameters	17-13
Hold mandatory parameters	17-13
Monitor mandatory parameters	17-14
Mute mandatory parameters	17-15
New call accepted mandatory parameters	17-16
New call rejected mandatory parameters	17-17
Play message mandatory parameters	17-17

- Play prompt collect digits mandatory parameters 17-18
- Port status mandatory parameters 17-19
- Query port mandatory parameters 17-19
- Query time of day (TOD) mandatory parameters 17-20
- Reconnect mandatory parameters 17-20
- Reset switch mandatory parameters 17-21
- Set billing record mandatory parameters 17-22
- Set IP address mandatory parameters 17-23
- Stop message mandatory parameters 17-24
- Transmit signinfo mandatory parameters 17-25
- PSN LOPER mandatory parameters for events 17-26
 - Agent data mandatory parameters 17-26
 - Current time of day (TOD) mandatory parameters 17-27
 - Digits collected mandatory parameters 17-28
 - Error detected mandatory parameters 17-29
 - In service mandatory parameters 17-29
 - Instruction completed mandatory parameters 17-30
 - Message played mandatory parameters 17-31
 - Off-hook mandatory parameters 17-33
 - On-hook mandatory parameters 17-34
 - Port status mandatory parameters 17-35
 - Query port mandatory parameters 17-35
 - Route not available mandatory parameters 17-36
 - Route selected mandatory parameters 17-37
 - Signaling event mandatory parameters 17-38
 - Stop heartbeat mandatory parameters 17-39
 - Tone detected mandatory parameters 17-39

UCS PSN parameters version 3

18-1

- PSN peer-to-peer application protocol 18-1
 - PSN parameter definitions 18-1
 - Access type 18-14
 - Agent data info 18-15
 - Agent type parameter 18-16
 - Bearer capability 18-17
 - Billing info 18-18
 - Call reference ID (CRID) 18-19
 - Call type 18-20
 - Control info parameter 18-22
 - COT required parameter 18-24
 - CRID Parameter 18-25
 - Destination trunk group 18-26
 - Digit collection parameter 18-27
 - Digits collected 18-29
 - Digits outputted 18-32
 - Digits to output 18-33
 - Error cause 18-35
 - Flow control info parameter 18-37
 - Flow control encountered parameter 18-38
 - Info change reason parameter 18-39
 - Instruction ID 18-40

Instruction tag	18-41
ISUP index	18-42
Message info	18-42
Monitor mask	18-44
Parameter ID	18-45
Point in call	18-47
Port info	18-49
Port service info	18-50
Port status	18-52
Reset reason	18-54
Serving translation scheme	18-54
Session ID	18-55
Signaling info	18-56
Signaling type	18-66
SigInfo mask	18-67
Switch ID	18-70
Time of day	18-71
Tone detected	18-72

UCS PSN LOPER messages and parameters version 3 **19-1**

LOPER format	19-1
Messages sent from the SCU to the PSN	19-1
Messages sent from the PSN to the SCU	19-4
Parameters	19-6
PSN LOPER mandatory parameters for primitives	19-8
Bridge mandatory parameters	19-9
Collect digits mandatory parameters	19-10
Connect mandatory parameters	19-10
Disconnect mandatory parameters	19-11
Error detected mandatory parameters	19-12
Flow control mandatory parameters	19-13
Heartbeat mandatory parameters	19-13
Hold mandatory parameters	19-13
Monitor mandatory parameters	19-14
Mute mandatory parameters	19-15
New call accepted mandatory parameters	19-16
New call rejected mandatory parameters	19-17
Play message mandatory parameters	19-17
Play prompt collect digits mandatory parameters	19-18
Port status mandatory parameters	19-19
Query port mandatory parameters	19-19
Query time of day (TOD) mandatory parameters	19-20
Reconnect mandatory parameters	19-20
Reset switch mandatory parameters	19-21
Set billing record mandatory parameters	19-22
Set IP address mandatory parameters	19-23
Stop message mandatory parameters	19-24
Transmit signinfo mandatory parameters	19-25
PSN LOPER mandatory parameters for events	19-26
Agent data mandatory parameters	19-26
Current time of day (TOD) mandatory parameters	19-27

- Digits collected mandatory parameters 19-28
- Error detected mandatory parameters 19-29
- In service mandatory parameters 19-29
- Instruction completed mandatory parameters 19-30
- Message played mandatory parameters 19-31
- Off-hook mandatory parameters 19-33
- On-hook mandatory parameters 19-34
- Port status mandatory parameters 19-35
- Query port mandatory parameters 19-35
- Route not available mandatory parameters 19-36
- Route selected mandatory parameters 19-37
- Signaling event mandatory parameters 19-38
- Stop heartbeat mandatory parameters 19-39
- Tone detected mandatory parameters 19-39

UCS PSN parameters version 4 **20-1**

- PSN Peer To Peer Application Protocol 20-1
- PSN Parameter Definitions 20-1
- Agent Data Info 20-14
- Agent Type Parameter 20-15
- Bearer Capability parameter 20-16
- Billing Digits Parameter 20-17
- Billing Info parameter 20-19
- Call Type parameter 20-20
- Control Info Parameter 20-22
- COT Required Parameter 20-24
- Destination Trunk Group parameter 20-25
- Digit Collection Parameter 20-25
- Digits Collected parameter 20-27
- Digits Outpulsed parameter 20-30
- Digits To Outpulse parameter 20-31
- Error Cause parameter 20-33
- Flow Control Info Parameter 20-34
- Flow Control Encountered Parameter 20-36
- Info Change Reason Parameter 20-37
- Instruction ID parameter 20-37
- Instruction Tag parameter 20-38
- ISUP Index parameter 20-39
- Message Info parameter 20-40
- Monitor Mask parameter 20-42
- Parameter ID parameter 20-43
- Point In Call parameter 20-45
- Port Info parameter 20-47
- Port Service Info parameter 20-47
- Port Status parameter 20-50
- Reset Reason parameter 20-51
- Serving Translation Scheme parameter 20-52
- Session ID parameter 20-53
- Signalling Info parameter 20-54
- Signalling Type parameter 20-66
- Switch ID parameter 20-67

Time of Day parameter 20-68
Tone Detected parameter 20-69

UCS PSN LOPER messages and parameters version 4 **21-1**

LOPER Format	21-1
Messages sent from the SCU to the PSN	21-2
Messages sent from the PSN to the SCU	21-4
Parameters	21-6
PSN LOPER Mandatory Parameters for Primitives	21-8
Bridge Mandatory Parameters	21-8
Collect Digits Mandatory Parameters	21-10
Connect Mandatory Parameters	21-10
Disconnect Mandatory Parameters	21-11
Error Detected Mandatory Parameters	21-12
Flow Control Mandatory Parameters	21-12
Heartbeat Mandatory Parameters	21-13
Hold Mandatory Parameters	21-13
Monitor Mandatory Parameters	21-14
Mute Mandatory Parameters	21-14
New Call Accepted Mandatory Parameters	21-15
New Call Rejected Mandatory Parameters	21-16
Play Message Mandatory Parameters	21-17
Play Prompt Collect Digits Mandatory Parameters	21-18
Port Status Mandatory Parameters	21-18
Query Port Mandatory Parameters	21-19
Query Time of Day (TOD) Mandatory Parameters	21-20
Reconnect Mandatory Parameters	21-20
Reset Switch Mandatory Parameters	21-21
Set Billing Record Mandatory Parameters	21-21
Set IP Address Mandatory Parameters	21-22
Stop Message Mandatory Parameters	21-23
Transmit Siginfo Mandatory Parameters	21-24
PSN LOPER Mandatory Parameters for Events	21-25
Agent Data Mandatory Parameters	21-25
Current Time of Day (TOD) Mandatory Parameters	21-27
Digits Collected Mandatory Parameters	21-27
Error Detected Mandatory Parameters	21-28
In Service Mandatory Parameters	21-29
Instruction Completed Mandatory Parameters	21-29
Message Played Mandatory Parameters	21-30
New Call Mandatory Parameters	21-31
Off Hook Mandatory Parameters	21-33
On Hook Mandatory Parameters	21-33
Port Status Mandatory Parameters	21-34
Query Port Mandatory Parameters	21-35
Route Not Available Mandatory Parameters	21-35
Route Selected Mandatory Parameters	21-36
Signalling Event Mandatory Parameters	21-37
Stop Heartbeat Mandatory Parameters	21-38

Tone Detected Mandatory Parameters 21-39	
Appendix A PSN for UCS DMS-250 switch	22-1
Entering Server Mode 22-1	
Specifications 22-2	
Signaling protocol 22-3	
PSN Messaging 22-10	
PSN Agencies 22-14	
Feature interactions 22-14	
Terminating Agent Bearer Capability (BC) Screening 22-15	
PSN Optionality 22-16	
PSN Optionality Control Requirements 22-16	
PSN Optionality Control Feature Interaction 22-18	
PSN Billing 22-18	
Default CDR 22-18	
Populated CDR 22-19	
PSN Billing Details 22-20	
Additional information 22-21	
Engineering/Hardware 22-21	
Logs 22-21	
Data schema 22-22	
Commands 22-22	
Operational measurements (OM) 22-23	
Restrictions/Limitations 22-23	
SS7 and PRI Messages and Parameters Supported in SIGINFO Parameter 22-24	
SS7 Messages 22-24	
Q.931 PRI Messages 22-28	
New call information collected 22-31	
Parameters in the NEW CALL Event Notification 22-34	
New Call – BEFORE Querying the AIN SCP 22-35	
New Call – AFTER Querying the AIN SCP 22-68	
<hr/>	
List of terms	23-1
<hr/>	
Ordering information	24-1

About this document

When to use this document

This document provides information about the Programmable Service Node (PSN) for UCS DMS-250 switches and how it operates in conjunction with a Service Control Unit (SCU). This document also describes the messages, events, and primitives used by the UCS DMS-250 PSN and SCU to provide call control.

The PSN is a flexible platform that provides operating companies the ability to rapidly deploy advanced services into their network. This ability is achieved by allowing an external computing platform to control the call processing on the switch using a high speed data link.

A peer-to-peer application protocol (called the SPI or Service Programming Interface) has been created for the PSN platform and is included in this document. This protocol defines the primitives and event notification messages sent between the SCU and the PSN. SPI also defines the parameters that are sent with the primitives and event notifications.

Numerous data tables, office parameters, operational measurements, and log reports support the UCS DMS-250 PSN and the SCU and, therefore, are presented throughout this document. However, the discussion of these items entails only how these items pertain to the UCS DMS-250 PSN and the SCU. This document does not discuss all the applications these items may support. For example, a data field that has many entry options show only those options that pertain to the UCS DMS-250 PSN and the SCU.

For information on how the data tables, office parameters, operational measurements, and log reports support other applications, refer to the appropriate document listed in “References in this document.”

Intended audience

This document is intended for use by operating company personnel who have responsibility for implementing or deploying advanced services into their network. This document is also intended for use by operating company personnel who have received Nortel Networks approved training for the table editor, table datafill, translations, and maintenance of the PSN as well as the UCS DMS-250 switch.

What is in this document

The chapters in this document provide the following information:

Chapter 1, “PSN introduction”

Chapter 1 provides introductory information about the PSN.

Chapter 2, “PSN data communication”

Chapter 2 provides information about the software and hardware elements involved in data communication between the PSN and other network elements.

Chapter 3, “PSN messages”

Chapter 3 provides information about the messages sent between the PSN and other network elements.

Chapter 4, “PSN finite state machine”

Chapter 4 provides information about the finite state machine for the PSN.

Chapter 5, “PSN flow control”

Chapter 5 provides information about how the PSN controls the flow of traffic between itself and other network elements.

Chapter 6, “PSN admin process”

Chapter 6 provides administrative information for the PSN.

Chapter 7, “PSN audit process”

Chapter 7 describes the PSN audit process.

Chapter 8, “PSN logs”

Chapter 8 describes the logs associated with the PSN.

Chapter 9, “PSN operational measurements”

Chapter 9 describes the operational measurements (OMs) associated with the PSN.

Chapter 10, “PSN office parameters”

Chapter 10 describes the office parameters associated with the PSN.

Chapter 11, “PSN billing”

Chapter 11 provides a brief description about the PSN handles billing data.

Chapter 12, “UCS SPI introduction”

Chapter 12 provides information about the Service Programmable Interface (SPI).

Chapter 13, “UCS PSN parameters version 1”

Chapter 13 describes the software parameters involved in PSN call processing control (version 1).

Chapter 14, “UCS PSN LOPER messages and parameters version 1”

Chapter 14 describes the Low Overhead Protocol Encoding Rule (LOPER) messages and parameters for the PSN call processing software (version 1).

Chapter 15, “PRI messages”

Chapter 15 describes the Primary Rate Interface (PRI) messages associated with the PSN.

Chapter 16, “UCS PSN parameters version 2”

Chapter 16 describes the software parameters involved in PSN call processing control (version 2).

Chapter 17, “UCS PSN LOPER messages and parameters version 2”

Chapter 17 describes the LOPER messages and parameters and messages for the PSN call processing software (version 2).

Chapter 18, “UCS PSN parameters version 3”

Chapter 18 describes the software parameters involved in PSN call processing control (version 3).

Chapter 19, “UCS PSN LOPER messages and parameters version 3”

Chapter 19 describes the LOPER messages and parameters for the PSN call processing software (version 3).

Chapter 20, “UCS PSN parameters version 4”

Chapter 20 describes the software parameters involved in PSN call processing control (version 4).

Chapter 21, “UCS PSN LOPER messages and parameters version 4”

Chapter 21 describes the LOPER messages and parameters for the PSN call processing software (version 4).

Chapter 22, “Appendix A PSN for UCS DMS-250 switch

Chapter 22, provides an appendix which contains additional information about how PSN applies to the UCS DMS-250 switch.

How to check the version and issue of this document

The version and issue of the document are indicated by numbers, for example, 01.01.

The first two digits indicate the version. The version number increases each time the document is updated to support a new software release. For example, the first release of a document is 01.01. In the *next* software release cycle, the first release of the same document is 02.01.

The second two digits indicate the issue. The issue number increases each time the document is revised but rereleased in the *same* software release cycle. For example, the second release of a document in the same software release cycle is 01.02.

This document is written for all UCS DMS-250 offices. More than one version of this document may exist. To determine whether you have the latest version of this document and how documentation for your product is organized, check the release information in the *UCS DMS-250 Master Index of Publications*.

References in this document

The following documents are referred to in this document:

- *UCS DMS-250 Alarm Clearing and Performance Monitoring Procedures*, 297-2621-543
- *UCS DMS-250 Billing Records Application Guide*, 297-2621-395
- *UCS DMS-250 Commands Reference Manual*, 297-2621-819
- *UCS DMS-250 Data Schema Reference Manual*, 297-2621-851
- *UCS DMS-250 Logs Reference Manual*, 297-2621-840
- *UCS DMS-250 Master Index of Publications*, 297-2621-001
- *UCS DMS-250 NetworkBuilder Application Guide*, 297-2621-370
- *UCS DMS-250 Office Parameters Reference Manual*, 297-2621-855
- *UCS DMS-250 Operational Measurements Reference Manual*, 297-2621-814
- *UCS DMS-250 Service Operation Support Manual*, 297-2621-011
- *UCS DMS-250 Software Optionality Control (SOC) User's Manual*, 297-2621-301
- *UCS DMS-250 SuperNode OM System Reference Manual*, 297-2621-322
- *Peripheral Modules Maintenance Guide*, 297-1001-592
- *TR-NWT-000317 Switching Systems Requirements for Call Control Using ISDNUP*
- *TR-NWT-000394 Switching Systems Requirements for IEC Interconnection using ISDNUP*
- *TR-NWT-000444 Switching Systems Requirements Supporting ISDN Access Using the ISDNUP*

What precautionary messages mean

The types of precautionary messages used in Nortel Networks documents include attention boxes and danger, warning, and caution messages.

An attention box identifies information that is necessary for the proper performance of a procedure or task or the correct interpretation of information or data. Danger, warning, and caution messages indicate possible risks.

Examples of the precautionary messages follow.

ATTENTION Information needed to perform a task

ATTENTION

If the unused DS-3 ports are not deprovisioned before a DS-1/VT Mapper is installed, the DS-1 traffic will not be carried through the DS-1/VT Mapper, even though the DS-1/VT Mapper is properly provisioned.

DANGER Possibility of personal injury



DANGER

Risk of electrocution

Do not open the front panel of the inverter unless fuses F1, F2, and F3 have been removed. The inverter contains high-voltage lines. Until the fuses are removed, the high-voltage lines are active, and you risk being electrocuted.

WARNING Possibility of equipment damage



WARNING

Damage to the backplane connector pins

Align the card before seating it, to avoid bending the backplane connector pins. Use light thumb pressure to align the card with the connectors. Next, use the levers on the card to seat the card into the connectors.

CAUTION Possibility of service interruption or degradation



CAUTION

Possible loss of service

Before continuing, confirm that you are removing the card from the inactive unit of the peripheral module. Subscriber service will be lost if you remove a card from the active unit.

How commands, parameters, and responses are represented

Commands, parameters, and responses in this document conform to the following conventions.

Input prompt (>)

An input prompt (>) indicates that the information that follows is a command:

>BSY

Commands and fixed parameters

Commands and fixed parameters that are entered at a MAP terminal are shown in uppercase letters:

>BSY CTRL

Variables

Variables are shown in lowercase letters:

>BSY CTRL ctrl_no

The letters or numbers that the variable represents must be entered. Each variable is explained in a list that follows the command string.

Responses

Responses correspond to the MAP display and are shown in a different type:

```
FP 3 Busy CTRL 0: Command request has been submitted.  
FP 3 Busy CTRL 0: Command passed.
```

The following excerpt from a procedure shows the command syntax used in this document:

- 1 Manually busy the CTRL on the inactive plane by typing

>BSY CTRL ctrl_no
and pressing the Enter key.

where

ctrl_no is the number of the CTRL (0 or 1)

Example of a MAP response:

```
FP 3 Busy CTRL 0: Command request has been submitted.  
FP 3 Busy CTRL 0: Command passed.
```

PSN introduction

This chapter contains background information on the Programmable Service Node (PSN) Platform. It includes a high level description of how to enter the server mode from the in-switch call processing.

The PSN is a flexible platform that provides operating companies the ability to rapidly deploy advanced services into their network. This ability is achieved by allowing an external computing platform to control the call processing on the switch, using a high speed data link.

The UCS DMS-250 must contain datafill that routes certain calls to the SCU for processing. The existing Carrier Advanced Intelligent Network (CAIN) call processing logic provides the datafill requirements. Refer to the *UCS DMS-250 NetworkBuilder Application Guide* for more details.

Scope of this document

This document has been written with the following types of audiences in mind:

- Customers who want to know the details of the PSN Platform functionality.
- Service developers at the SCU. These designers may want to know the PSN to SCU protocol to create services at the SCU.

Note: Appendix A contains information on PSN that relates to the UCS DMS-250 product specifically.

Background

The operating companies have relied on programmable switching matrices to prototype and deploy services into the public switched network. Examples of such services include voice dialing, televoting, fax server capabilities, and debit card services. These small scale systems have been wired into the existing backbone networks as service overlay networks.

These systems have several disadvantages; for example, they do not offer the reliability or the capacity required for truly robust performance. Likewise, the different computing platforms introduced into the network to

support these overlay systems introduce maintenance and network management complexities, which result in high operations and sustaining costs.

Functional overview

The PSN provides a flexible platform for the switches, providing the operating companies the ability to rapidly deploy services into the network.

The PSN call processing is controllable, using an external control interface through a high speed data link. This control interface is called the Service Control Unit (SCU).

- When a call on the PSN is identified as a service call, the PSN presents the call to the SCU and provides the SCU with the data required to determine which service is to be invoked on this call.

The call is now in a Server Mode. This is a state where the call enters a client/server relationship with the SCU. In other words, the SCU has complete control of the call. The PSN does not make any decisions or take any actions that involve the knowledge of services or predetermined reactions, as is the case with the existing switch call processing.

- In the Server Mode, the SCU provides instructions on how to proceed with the call. These instructions are primitives, or macros.

The SCU has the ability to invoke such functions as digit collection, call terminations, bridging, playing messages, specialized tone monitoring, and outpulsing, over one or more parties involved with a SCU controlled call.

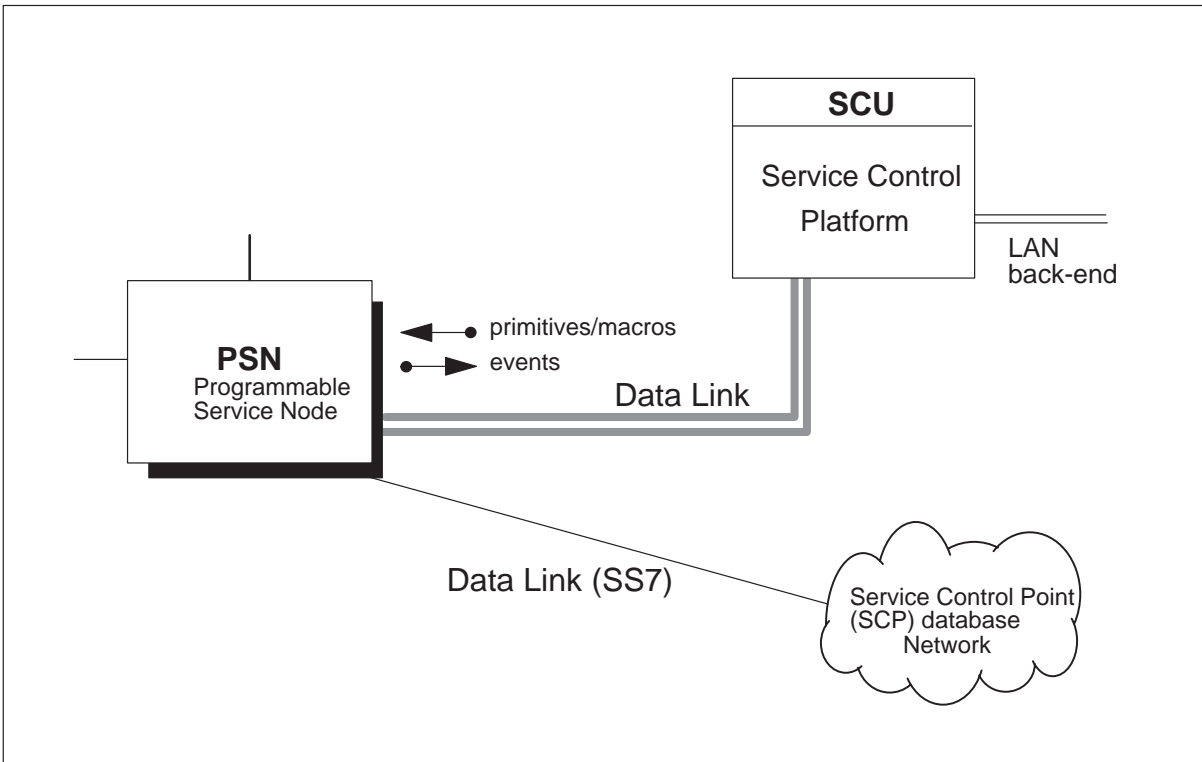
- Occasionally, when external or peripheral events occur at the PSN for any party that is involved in the SCU controlled call, the SCU is notified of these events.

In addition to call control, certain internal resources, such as the universal tone receiver (UTR)-based digit collection, specialized tone receiver (STR)-based tone monitoring, and E-DRAM announcement control are made available to the SCU.

Networking overview

The PSN platform consists of the Programmable Service Node connected to one or more SCUs. Also, a SCU may be connected to more than one PSN. See Figure 1-1 for the PSN switching matrix configuration.

Figure 1-1
PSN Switching Matrix Configuration



The SCU platform provides service control capabilities. It is connected to the PSN, using a high-speed communication interface or a high-speed data link. Some examples of high-speed data link include IEEE 802.3 Ethernet (the phase I offering), fiber distributed data interface (FDDI), and asynchronous transfer mode (ATM).

Identifying a service call

When a certain criteria is met for an in-switch call, the call triggers, it is identified as a service call, and enters the server mode. The criteria that is defined is customer-specific. For example, the call may trigger after all the digits are collected or after the agent goes Off-hook.

PSN data communications

Once the call is identified as a service call, the PSN and the SCU exchange messages which enables the SCU to control call processing on the PSN.

An Ethernet connection handles communication between the SCU and the PSN. The details of this dedicated connection are covered in the Chapter “PSN data communications”.

PSN peer-to-peer application protocol - SPI

A Peer-to-Peer Application protocol (or service programming interface [SPI]) has been created for the PSN platform. The protocol defines the primitives that are provided to allow the SCU to control the calls on the PSN and the event notifications that are reported to the SCU from the PSN. The protocol also defines the parameters that go along with the primitives and event notifications.

The Peer-to-Peer protocol definition utilizes generic functional references to data rather than specific PSN data requirements. The protocol, including the primitive/event notification definitions and message flow (with service implementation examples), is covered in the Chapter “PSN messages”.

SPI parameters

There are two types of SPI parameters: parameters common to all customers, such as the *PortInfo* parameter, and parameters specific to a customer, such as, the *SignalingInfo* parameter.

In addition, multiple versions of SPI are supported per customer for proper network upgrades.

PSN finite state machine

The interactions between all PSN primitives and event is controlled by the PSN finite state machine (FSM). This FSM fully defines all possible event sequences and their resulting outputs. The PSN FSM is covered in the Chapter “PSN finite state machine”.

PSN admin process

The PSN Admin process establishes and maintains the communication link between the SCU and the PSN. The Admin process is covered in the Chapter “PSN admin process”.

PSN audit process

The PSN Audit monitors all active PSN agents. The SCU is notified of all potentially hung PSN agents. The actions taken by the PSN Audit are controlled by a set of office parameters. The PSN Audit process is covered in the the Chapter “PSN audit process”.

PSN flow control

The call traffic originating at the PSN triggers data traffic at the PSN-SCU interface. This data traffic, under overload condition, may potentially overflow the Call Processing layer, receive queues at the PSN and the corresponding message facility at the SCU, and result in message loss. The message loss may in turn impact the services on existing calls. In order to

prevent this impact, PSN flow control provides a mechanism to control the flow of data by controlling the new call traffic generated at the PSN.

PSN office parameters

There are many new office parameters, and tables used to control and monitor the PSN. These parameters and tables are covered in the Chapter “PSN office parameters”.

PSN logs and OMs

There are many new PSN Logs and PSN OMs used to control and monitor the PSN. The logs and OMs are covered in the Chapter “PSN logs” and in the Chapter “PSN operational measurements”, respectively.

PSN billing

Strategic billing provided by PSN is quite small. The PSN billing specification is covered in the Chapter “PSN billing”.

PSN data communications

This chapter contains the detailed description of PSN Data Communications. This includes the PSN Server and Client architecture.

Functional overview

This section describes an ethernet data link interface between the PSN and the SCU by implementing a custom layer protocol over user datagram protocol/internet protocol (UDP/IP). The custom layer resides between the application layer and the transport layer. It provides connectionless transport of application layer's data across the network. The application layer's protocol data unit (PDU) is transported transparently by the custom layer at the PSN to the underlying transport layer to be communicated to the peer layer at the SCU.

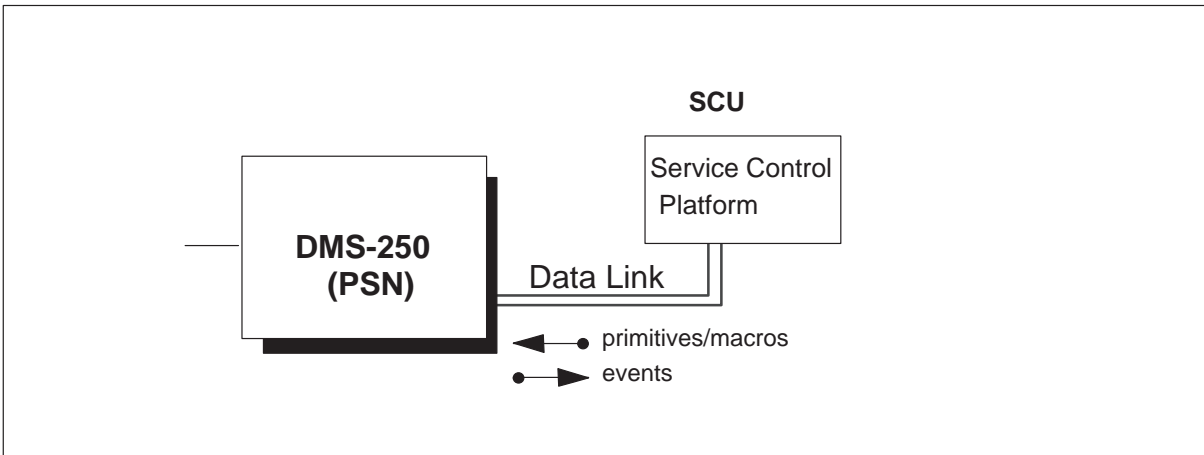
The chapters that follow describe the following functionalities:

- an ethernet data link interface between the PSN and the SCU for data communication
- the support for address resolution for multiple service applications running on the SCU
- the Admin process, which handles administrative and maintenance activities of the PSN features
- the Audit process, which handles periodic audit for active SCU agents

Networking overview

This section covers the networking overview for PSN. Figure 2-1 provides a PSN switching matrix configuration.

Figure 2-1
PSN Switching Matrix Configuration



An enhanced switching matrix is defined at the PSN, and the PSN is connected to the SCU, using an ethernet data link. The data link is used to send call control primitives and macros from the SCU to the PSN to provide SCU services to calls on PSN. Events are sent from the PSN, to the SCU, to notify the SCU of the completion of execution of the primitives and the occurrence of external events that affect the call. Refer to Figure 2-1, “PSN Switching Matrix Configuration.”

Data communications specification

This section covers the data communications specification.

Feature activation

This feature resides on the UDP/IP stack on PSN. At all restarts, the stack requires initialization and configuration information which is datafilled in LIUINV, IPNETWRK, IPROUTER, IPPROTO, and IPTHRON tables.

This feature requires the initialization information datafilled in table SCUADDR. This table stores the address information of the initial point of contacts at the SCU.

Ethernet connectivity for PSN is provided by a pair of EIUs that are configured in a pool. The feature requires that at least one of the two EIUs is in service at the time of initialization.

The office parameter `PSN_INIT_SCU_POLLING` in table OFCVAR is created to control the functionality of the group of PSN related features. If its value is “No”, then the process of establishing the SCU connection is not initiated. If its value is “Yes”, then the process of establishing the SCU connection is initiated.

For more information, refer to the *UCS DMS-250 Data Schema Reference Manual* and *UCS DMS-250 Office Parameters Reference Manual*.

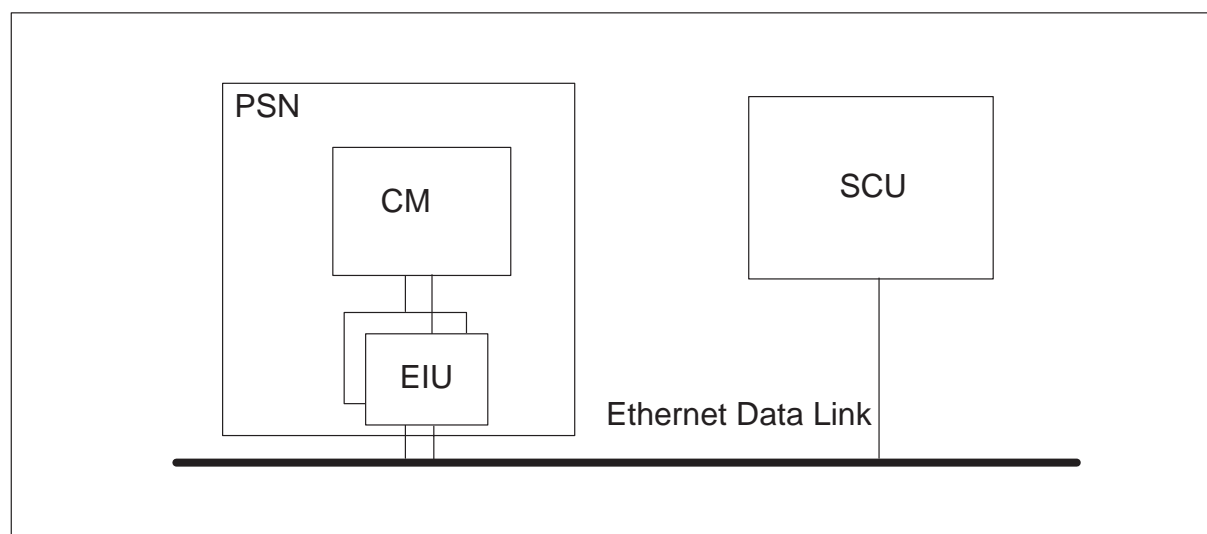
PSN – SCU physical interface

The PSN – SCU interface sends/receives information on an ethernet data link between the PSN and the SCU (see Figure 2-2).

Ethernet connectivity for PSN is provided by a pair of EIUs that are configured in a pool, where both EIUs are in service but the CM traffic is sent through only one EIU at a time. In case of the failure of the first EIU, the CM traffic is automatically directed to the second.

The SCU sends messages to the CM's IP Address and port number. The EIU's IP Address is not used as the destination IP Address. The EIU acts as a router and the EIU routing is transparent to the SCU.

Figure 2-2
PSN – SCU Physical Interface



The two EIUs can reside on a link peripheral processor (LPP), as shown in Figure 2-3, “EIU on the LPP,” or on a fiber link interface shelf (FLIS) housed in an application processor (AP) cabinet, as shown in Figure 2-4, “EIU resident on a FLIS shelf housed in an AP.”

It is required that the two EIUs and the SCU be configured on a subnet dedicated to the PSN application in order to isolate the PSN traffic from the traffic generated by other transmission control protocol (TCP)/IP applications running on the PSN, such as billing.

2-4 PSN data communications

Figure 2-3
EIU on the LPP

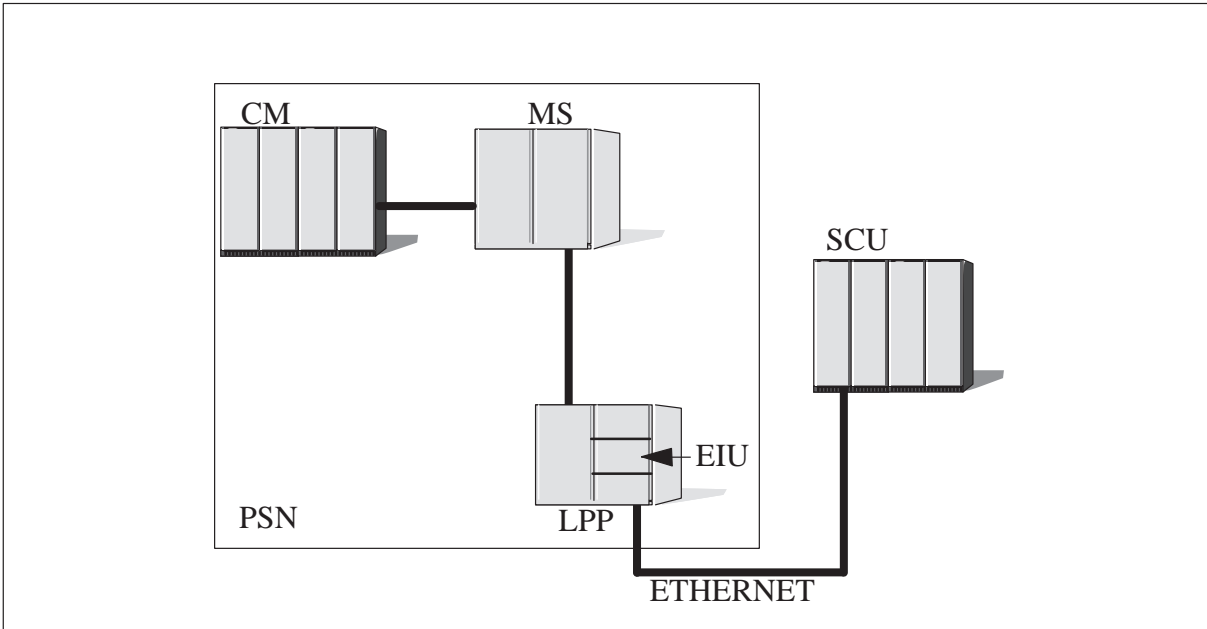
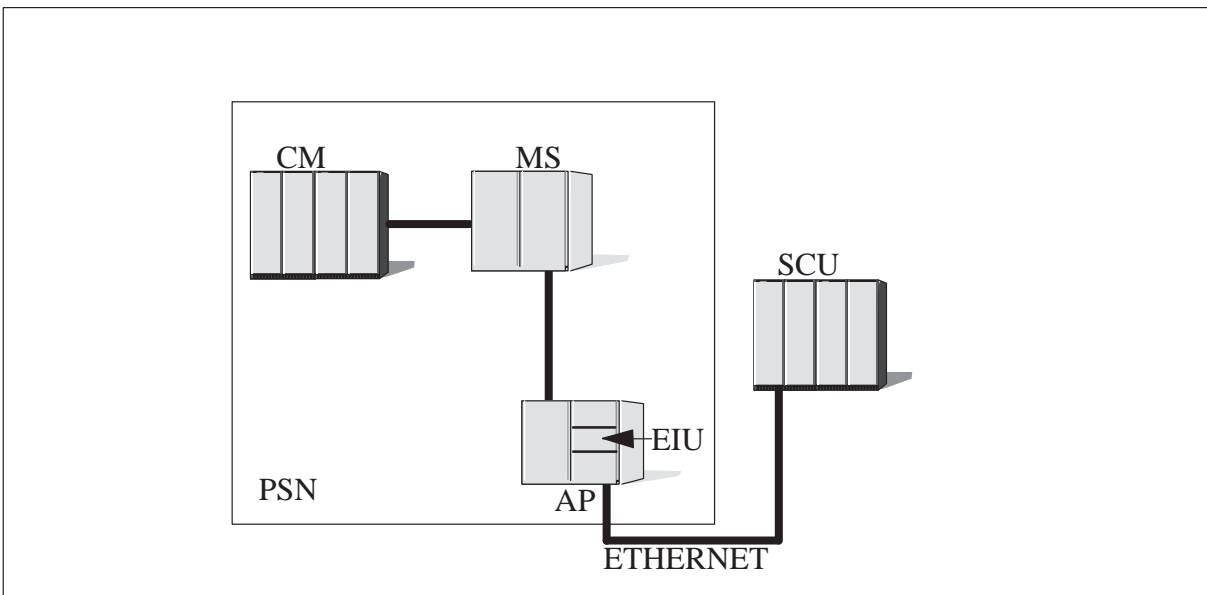


Figure 2-4
EIU resident on a FLIS shelf housed in an AP

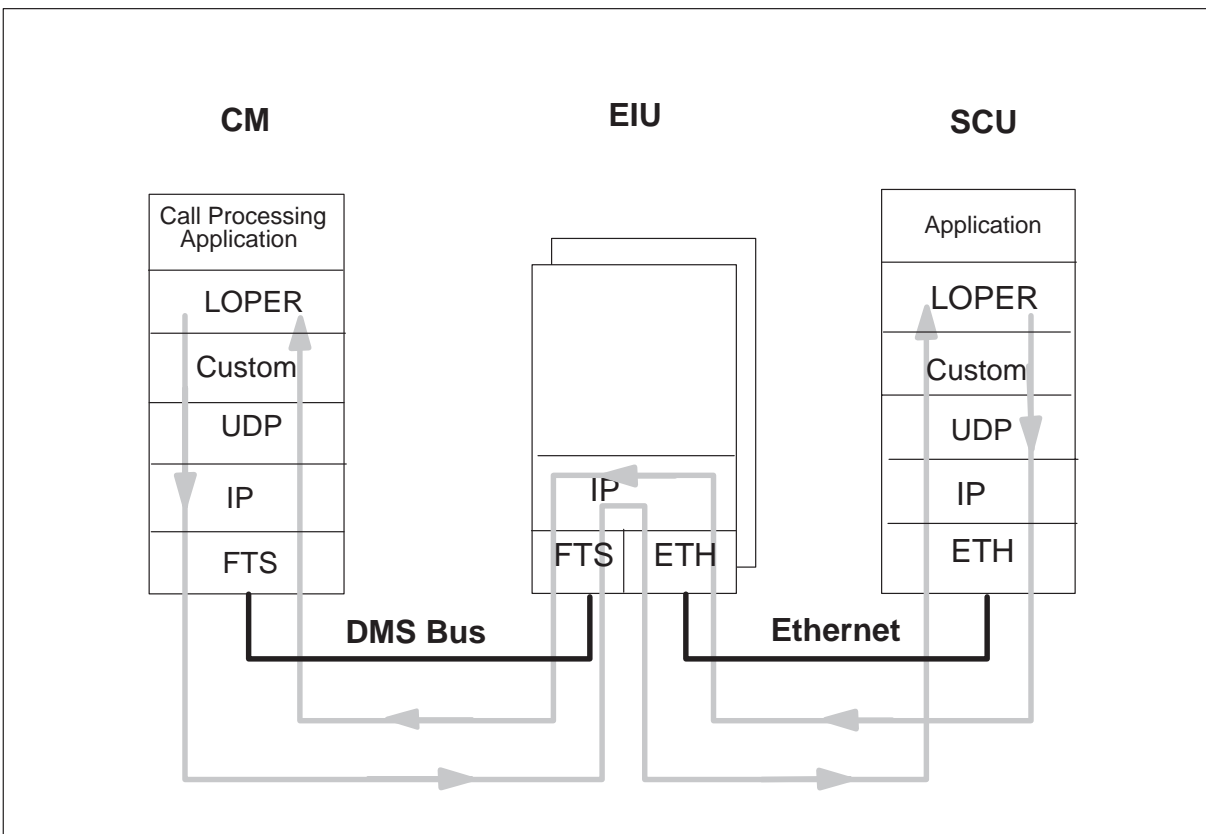


PSN – SCU protocol stack

A client/server model is implemented as an interface between the PSN and the SCU. The client/server model is designed using a custom layer protocol that sits on top of UDP.

Figure 2-5 shows the data path between the PSN and the SCU through the EIU. The EIU acts as a router, providing ethernet connectivity to the PSN.

Figure 2-5
PSN – SCU Data Path



Network & transport layers – IP & UDP

The existing UDP/IP protocol stack on PSN is used. The custom layer protocol is flexible enough to run over UDP or TCP; however, only UDP is supported in this feature.

LOPER

The low overhead protocol encoding rule (LOPER) provides a custom decoding/encoding protocol for PSN. The focus of this custom PSN PER is to mimic the internal data structures found in the PSN call processing system, which provides a LOPER. As a result, this scheme uses the internal

PSN structures, which reduces the amount of time spent on the encoding and decoding of each individual piece of information found in the PSN structures, since related information will be grouped together in parameters. These parameters are directly derived from the PSN internal structures, depending upon the information that is mandatory and optional to the primitive and event, and the proximity of information to each other in the PSN structures. Mandatory parameters are the parameters that are required by the primitive and event in order to be processed. Optional parameters are the parameters that are not required by the primitive and event but add extra functionality for processing. Since certain PSN structures are customer-dependent, the contents of the parameters derived from these structures are also customer-dependent. Encapsulating the internal PSN structures gives the ability of reading directly from, or writing directly to, the structure without an intermediate encoding or decoding step.

Custom layer protocol

The custom layer provides a connectionless transfer of application layer's PDU across the network. Since it is a connectionless protocol, messages are not guaranteed to be delivered, nor are those that are delivered, guaranteed to be passed to the application layer in the order they were sent by the peer layer at the SCU. The application layer must be designed to accommodate and recover from these situations.

The custom layer implements a datacom client and server model to communicate data across the network. The datacom client and server acts as an interface between the application layer and the underlying transport layer. In addition, the datacom client transfers the application layer's PDU to the underlying transport layer and the datacom server receives the application layer's PDU from the transport layer and transfers it to the application layer.

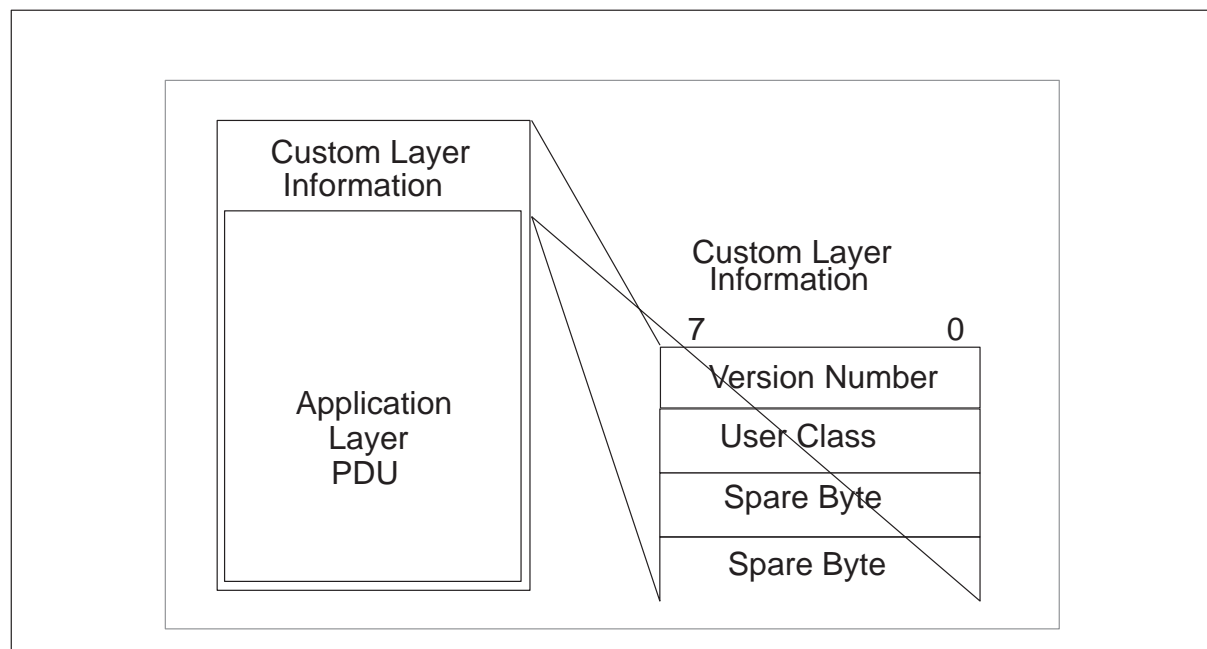
Custom layer message size

The maximum length of a message that can be communicated at the custom layer is restricted to 3000 bytes. This includes the custom layer message header and the application layer's PDU. Any message received at the PSN that is greater than 3000 bytes in length, is dropped. The PSN101 log is generated with the text reason "Message length exceeds maximum size", and the OM register MSGSIZE in the PSN_ERDC OM group is pegged at the PSN. Because the message is dropped prior to decoding, the SCU is not notified of the event.

Custom layer message header

Each application layer's PDU is wrapped in a custom layer's envelope before being sent to the SCU. The PDU is unwrapped by the peer layer at the SCU. Figure 2-6 shows the custom layer header.

Figure 2-6
Custom Layer header



The version number specifies the version of the custom protocol being used.

The user class field is used to differentiate between the different users of messages going across this layer. In this feature, only four types of messages are identified by this layer, which are encoded as follows:

Admin Class	00000001
CallP Class	00000010
Audit Class	00000011
Flow Control Class	00000100

Only the application layer PDU is encoded in LOPER. The headers (custom layer, UDP, and below) are not encoded in LOPER and are sent as raw data.

Datacom client/server model

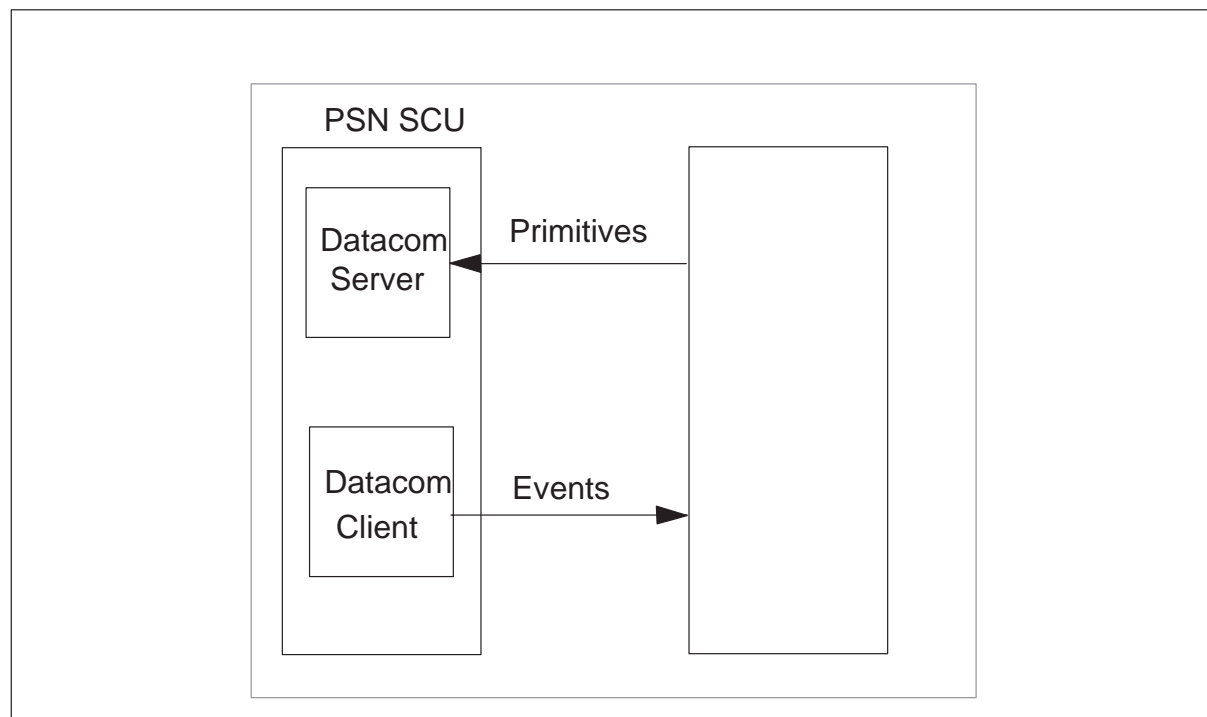
The custom layer implements a datacom client and server architecture to communicate the application layer's information to the SCU. The primitives sent from the SCU are received by the datacom server at the PSN, unwrapped, and passed to the application layer. Events are sent from the application layer to the datacom client, wrapped in the custom layer's envelope at the datacom client and sent to the SCU. Refer to Figure 2-7, "Datacom Client and Server Message Flow."

At the receipt of a message from the SCU, the datacom server performs the following activities:

- checks the length of the message. If the message length is less than the size of the custom header size (four bytes), the datacom server drops the message and generates PSN101 log at the PSN with text reason "Datacom header corrupted". The OM register DCOMHDR in the PSN_ERDC OM group is pegged. If the message length is greater than 3,000 bytes, it drops the message and generates PSN101 log at the PSN with text reason "Message length exceeds maximum size". That exceeded length is displayed in the field LENGTH and the values in the other fields of the log, USERCLASS and SPI VERSION, are ignored. The OM register MSGSIZE in the PSN_ERDC OM group is pegged.
- unwraps the message to extract its header and does necessary header processing. This includes the verification of the custom layer protocol version number and determination of the user class from the user class field. If any error occurs during this process, then the message is dropped and PSN101 log is generated at the PSN with the text reason "Unrecognized Datacom Version" or "Unrecognized Userclass". For the case of unrecognized version, the invalid version is displayed in the field SPI VERSION and the values in the other fields of the log, USERCLASS and LENGTH, are ignored. For the unrecognized userclass case, the invalid userclass value is displayed in the field USERCLASS and the other fields of the log, LENGTH and SPI VERSION, do contain valid values because they were decoded and validated prior to the userclass. The OM register DCOMHDR in the PSN_ERDC OM group is pegged. Because the message is dropped prior to decoding, the SCU is not notified of the event.
 - If the message passes the validity checks that are mentioned above, it is ready to be sent to the appropriate user application. The selection of the user application depends on the user class field in the custom layer header. For example, if the message is an admin user class, then the PDU is sent to the admin process.

- If the datacom server is not able to send the message to the User Application successfully, it drops the message. This happens if the communication path is not available or the communication buffers are not available. This is considered a software error on the PSN and the OM register in the PSN_ERDC OM group is pegged. The SCU, however, is not notified of the message being dropped.

Figure 2-7
Datacom Client and Server Message Flow



At the receipt of an event from the application layer, the datacom client performs the following activities:

- checks whether the user class of the event message is valid. If the user class of the event message is invalid then the message is dropped, PSN102 log is generated, and MSGDROP register in the PSN_ERDC OM group is pegged at the PSN.
- wraps the PDU in the custom layer's envelope.
- sends the message to the transport layer to be shipped out to the SCU. If the datacom client is not able to send the message across the network, it drops the message and pegs the MSGDROP register in the PSN_ERDC OM group. If the datacom client is able to send the message successfully, then the SMSGSENT register in the PSN_USAG OM group is pegged.

Address resolution

The SCU can be equipped with multiple service applications, each with a unique IP/Port address. This feature supports address resolution for multiple service applications residing at the SCU.

Initial contact address resolution

Up to three initial points of contact to the SCU can be datafilled in table SCUADDR. An IP Address, and a corresponding UDP port number, define a point of contact. When the PSN comes into service after a reboot, or after reload or cold restart, the point of contacts are polled periodically until an initial contact has been established between the PSN and any one of the points of contact. The initiation of this polling is controlled by office parameter `PSN_INIT_SCU_POLLING`. Initial contact address polling is also performed in the event of **Heartbeat_Failure**. For more details on **Heartbeat_Failure**, refer to Chapter “PSN Admin”.

PSN sends an **In_Service** event to the first point of contact that is datafilled in the table SCUADDR and waits for the **Set_IP_Address** primitive for a time specified by `RETRY_TIME`, datafilled in table SCUADDR, for that point of contact. If the **Set_IP_Address** primitive is not received within `RETRY_TIME`, and if the number of retries is less than `MAX_RETRY` datafilled in table SCUADDR for that point of contact, then another event retry is sent to the same point of contact. If the number of retries exceeds the `MAX_RETRY` and `RETRY_TIME` has expired, then the next point of contact is queried by sending an **In_Service** message. The second **In_Service** event sent to the same point of contact is regarded as the retry number 1 and so forth.

For example, if `RETRY_TIME=20` and `MAX_RETRY=1`, then the following is the logical flow for reestablishing SCU communication. PSN sends the first **In_Service** event to the point of contact and waits for the response. If SCU responds with the **Set_IP_Address** primitive within 20 seconds, then the contact has been reestablished. Otherwise, PSN checks the number of retries (which is currently 0) against the `MAX_RETRY`. Since the number of retries is less than `MAX_RETRY`, the PSN sends the **In_Service** event (retry number 1) to the same point of contact. If no **Set_IP_Address** primitive is received within 20 seconds, the PSN sends the **In_Service** event to the next point of contact, because `MAX_RETRY` retries have been performed for the first point of contact.

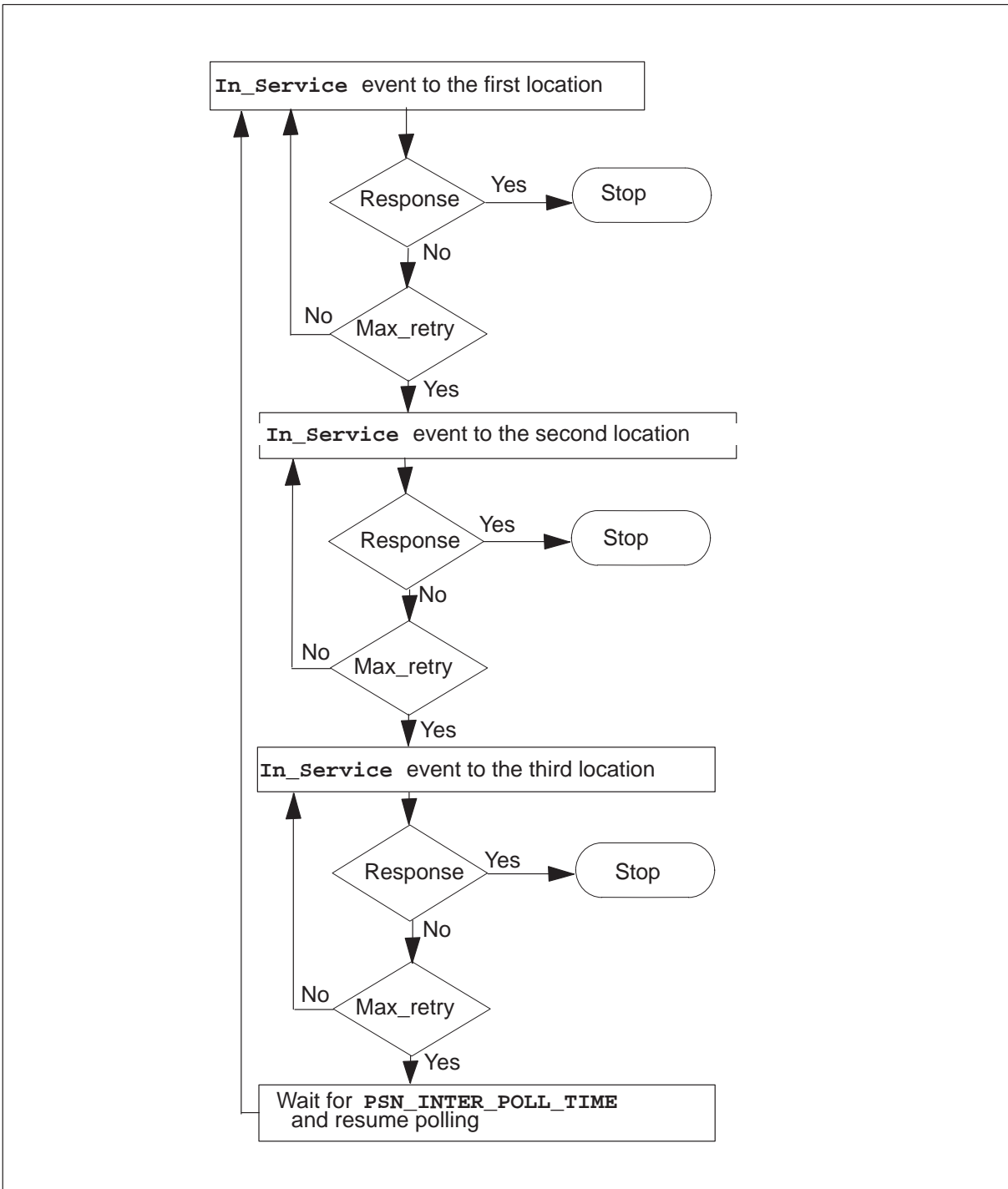
In_Service

If a **Set_IP_Address** primitive is not received from any of the points of contact, then PSN waits for office parameter `PSN_INTER_POLL_TIME` and starts the polling cycle again. This process is repeated until a contact is established between PSN and the SCU (any one of the initial points of

contact has responded to the **In_Service** event by sending a **Set_IP_Address** primitive). The **Set_IP_Address** primitive contains the address of the SCU arbitrator. For details on the SCU arbitrator address, refer to the section of this Chapter “Multiple service applications addressing”. If the PSN receives more than one **Set_IP_Address** primitive, then the most recent **Set_IP_Address** primitive overrides the SCU arbitrator address. A PSN100 log with text reason “Unable to establish SCU communication” is generated at each expiration of **RETRY_TIME**.

Refer to Figure 2-8 for the initial contact polling algorithm.

Figure 2-8
Initial Contact Polling Algorithm



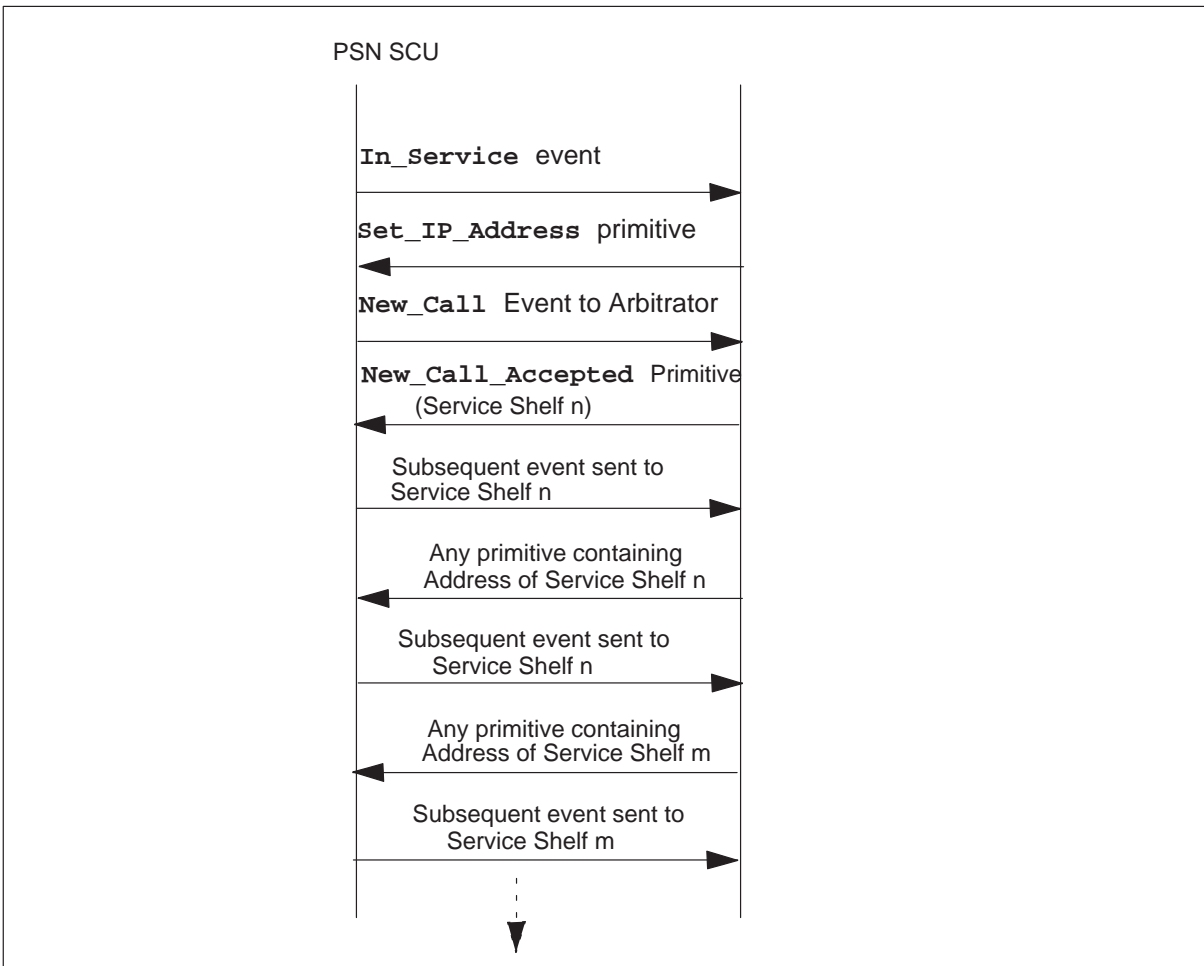
Multiple Service Applications addressing

The **Set_IP_Address** primitive contains the address (the IP Address and the UDP port) of the arbitrator at the SCU. The arbitrator address is stored at the PSN and is subsequently used to send all **New_Call** events to the SCU. Until PSN gets the IP Address of the service application, all events are sent to the Arbitrator.

The Arbitrator at the SCU directs the control of the new call to an appropriate service application and sends a **Call_Control** primitive with the address of the service application handling this agent. This address is used at the PSN to send the subsequent events related to that call, until a new address is sent in any primitive from the SCU, which is then used to send any subsequent events related to that call. The service application address information is an optional parameter of each primitive sent from the SCU.

Refer to Figure 2-9 for the PSN – SCU Physical Interface.

Figure 2-9
PSN – SCU Physical Interface



PSN Restarts

This section covers PSN restarts.

Cold and reload restarts

On cold and reload restarts, all the calls at the PSN are taken down, all the messages in transition are lost, all the protocol related processes are recreated, and the initial contact polling is invoked to inform the SCU of the re-initialization process and to get a new arbitrator's address from the SCU.

Warm restarts

On warm restarts, all the messages in transition are lost and the protocol related processes are recreated. Neither the active calls (calls in a stable state) are taken down, nor is the initial contact polling invoked at the PSN. After warm restart the audit checks all the PSN agents that were active

before the restart. If the agents are down after the restart, the audit informs the SCU that this port is down by sending an error detected with reason “PORT_DOES_NOT_EXIST_EC” to the SCU.

SCU resets

At any time during the conversation with the PSN, the SCU may go through different kinds of resets. Each of these resets have a unique impact on the call processing at the PSN and is described as follows:

System resets

On system level resets at the SCU, the SCU sends **Reset_Switch** primitive to Primitive Server with a nil IP Address and a nil UDP port number that can cause the PSN to release all the service calls. This depends on the value of office parameter `PSN_DROP_AGENTS_SCU_SYS_RESET`

The SCU may send the **Set_IP_Address** primitive with the **Reset_Switch** primitive, which causes the PSN to update the address of the SCU Arbitrator.

Shelf resets

On resets at a shelf level, the SCU sends a **Reset_Switch** primitive with the IP address of the affected service shelf with a nil port number. Depending on the value of office parameter `PSN_DROP_AGENTS_SCU_SHELF_RESET` this can cause the PSN to release all the service agents related to that shelf. A nil port number is sent to eliminate the need to send multiple port numbers servicing the PSN off of the affected service shelf.

Service resets

There may be several ports servicing the PSN from a single service shelf. On resets at the port level, the SCU sends the **Reset_Switch** primitive with the port number and IP Address of the affected service shelf. Depending on the value of a new office parameter `PSN_DROP_AGENTS_SCU_SRVC_RESET` this may cause the PSN to release all the service agents related to that particular port on the affected service shelf.

Agent resets

On agent level resets, the SCU sends a **Disconnect** primitive with the *PortInfo* parameter of the affected agent. This causes the PSN to release the agents.

EIU maintenance and link level external node maintenance

Refer to the *Peripheral Modules Maintenance Guide* for details on EIU maintenance.

I/O interface specifications

UDP/IP interface is provided to communicate application layer information between the PSN and the SCU.

Switch/network upgrade requirements

A pair of dedicated EIUs in pool configuration are recommended as routers for ethernet connectivity.

Error handling specifications – message errors

Messaging errors can occur at any time during the encoding, decoding, and transport of messages across the different protocol layers at the PSN and among peer layers across the network.

Maximum message size exceeded

A message received at the PSN with a length greater than 3000 bytes is considered an error. The message is dropped, a PSN101 log is generated with text reason “Message length exceeds maximum size”, and an OM register in the PSN_ERDC OM group is pegged.

Failure to decode a message at the custom layer

If the size of the message is less than the size of the custom header, then the message is dropped, a PSN101 log is generated with the text reason “Datacom header corrupted,” and an OM register in the PSN_ERDC OM group is pegged. If the datacom server is not able to decode the header of an incoming message, the message is dropped, a PSN101 log is generated with the text reason “Unrecognized Datacom Version” or “Unrecognized Userclass”, depending on the decoding error, and an OM register in the PSN_ERDC OM group is pegged.

Failure to establish communication to SCU

If the admin process sends an **In_Service** event and fails to receive the **Set_IP_Address** primitive, then PSN100 log is generated with the text reason “Unable to establish SCU communication,” and an OM register in the PSN_ERDC OM group is pegged.

Failure to receive SCU heartbeat message

If the admin process fails to receive the **Heartbeat** primitive during the period of office parameter **PSN_HEARTBEAT_WAIT_TIME**, then PSN100 log is generated with the text reason “SCU Heartbeat Failure,” and OM register in the PSN_ERDC OM group is pegged.

Invalid Set_IP_Address primitive parameters

When the admin process receives the **Set_IP_Address** primitive with one or more invalid parameters (Non-nil Trunk group number, Non-nil Trunk member number, Nil IP Address and/or Nil Port number), it generates a

PSN105 log with the text reason “Invalid data for Set IPAddr.” In this case, the IP Address of the SCU Arbitrator is not updated at the PSN.

Invalid sender of heartbeat message

The admin process expects the **Heartbeat** primitive message from the SCU Arbitrator application. This ensures the PSN user applications that the SCU Arbitrator is alive and able to handle the **New_Call** Events. When the admin process receives the **Heartbeat** primitive from an application other than the Arbitrator, then the admin process generates PSN105 log with the text reason “Invalid Sender of Heartbeat” and does not reset the heartbeat timer.

Error in decoding primitive message

The user applications (admin, audit, and primitive server processes) decode the incoming primitive that is encoded in LOPER by the SCU. If there is any error decoding this message, then the user application generates PSN202/PSN212 log with the appropriate error code.

Receive the Error_Detected Primitive message

The SCU can send the **Error_Detected** primitive message to the user application when the SCU detects any error in the event message sent by the user application. The user application generates PSN200 log with the error cause sent by the SCU.

Invalid primitive for a specific user class

The admin process expects “**Set_IP_Address**”, “**Heartbeat**”, “”, and “**Query_TOD**”, “**Error_Detected**” primitives. If the admin process receives a primitive other than one mentioned above, it generates PSN207 log with the text reason “Primitive not supported by Userclass”. The primitive server process as well as the audit process generates a PSN207 log when they receive any primitive they do not support.

Number of primitives exceeded

The admin process can handle up to three primitives in an incoming message. If the SCU sends more than three primitives in a message, then the admin process generates PSN209 log with the text reason “Primitives exceeded for Userclass”.

Logs – Datafill Problem

Table SCUADDR is used to datafill up to three initial points of contact at the SCU. If this table is empty, then the admin process does not send the **In_Service** event to the SCU and generates PSN103 log with the text reason “Table SCUADDR not datafilled”. The admin process waits for a period specified by a new office parameter **PSN_INTER_POLL_TIME** and then retries to access the table SCUADDR.

Logs – SCU Arbitrator Address

When the admin process receives the **Set_IP_Address** primitive that changes the SCU Arbitrator address, it generates PSN104 log with the old and new addresses of the SCU Arbitrator.

Engineering hardware information

This section covers engineering hardware information.

Ethernet Interface Units

A pair of EIUs in a pool configuration is required to provide ethernet connectivity to the PSN. The EIUs can be housed in an LPP, or in an AP, on a FLIS shelf.

The EIU is a 3 card hardware set:

- NTEX22BA/BB: Integrated PBus and FBus card (IPF)
- NT9X84AA: Ethernet Interface Card (EIC)
- NT9X85AA: Ethernet Interface Paddleboard (EIP)

Refer to *Peripheral Modules Maintenance Guide* for details on EIU installation and maintenance.

An ethernet cable is also required.

PSN datacom server's UDP Port Number

The UDP port number for the PSN datacom server on PSN is selected to be 30,726. The SCU should use this port number with the IP Address of the PSN to send data using UDP interface.

Multi-Application certification For LPP and FLIS

This activity is certified to use EIUs on FLIS and LPP with data throttled at 12Kbyte/s on the transmit link and 24Kbyte/s on the receive link. It is recommended that throttle values of 12Kbyte/s for transmitting data and 24Kbyte/s for receiving data be datafilled in table IPTHRON.

Restrictions and limitations

The following restrictions and limitations apply:

- The custom layer is a connectionless protocol that does not provide guaranteed delivery of messages to the application layer.
- UDP is the only transport layer protocol that is supported by PSN.
- The length of the custom layer message is restricted to 3000 bytes. Any messages received that exceed this length are dropped, and the SCU is not notified of the event.

- PSN traffic at any rate greater than 12 kbyte/s on the transmit link and 24 kbyte/s on the receive link is not supported.

PSN messages

This chapter contains a detailed description of the PSN Messages. The PSN messages include primitives and macros (sent from the SCU to the PSN) as well as the event notifications (sent from the PSN to the SCU).

PSN peer-to-peer application protocol

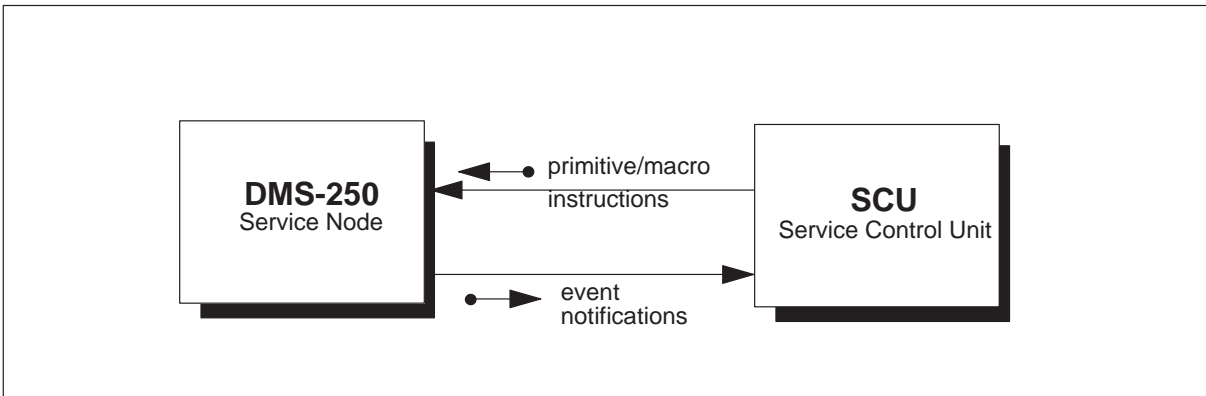
A Peer-to-Peer Application protocol has been created for the PSN platform. The protocol defines the primitives that are provided to allow the SCU to control the calls on the PSN and the event notifications that are reported to the SCU from the PSN. The protocol also defines the parameters that go along with the primitives and event notifications.

The Peer-to-Peer protocol definition utilizes generic functional references to data rather than specific PSN data requirements. The protocol, including the primitive/event notification definitions, parameter definitions, and message flow (with service implementation examples) is described later in this document.

PSN messaging

The PSN messaging consists of instructions from the SCU to the PSN and notifications from the PSN to the SCU in the form of primitives, macros, and event notifications. This is shown in Figure 3-1.

Figure 3-1
Primitives, Macros, and Event Notification Messages

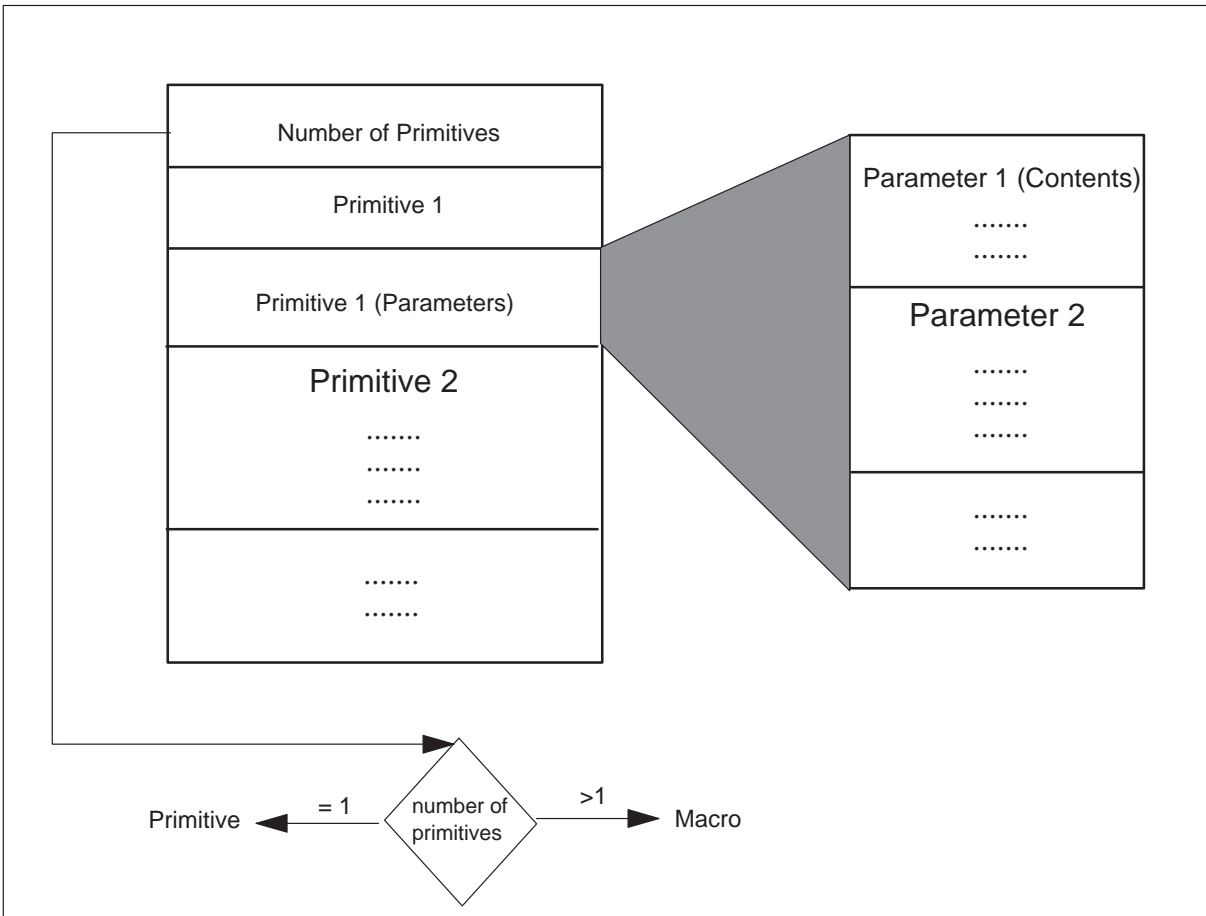


These instructions and event notifications are packaged into a message and sent back and forth between the PSN and the SCU. Each message that is sent to the SCU, or received from the PSN, causes an OM register in the PSN_USAG OM group to be pegged. The register SMSGRCVD is pegged for each message that is received from the SCU, and the SMSGSENT register is pegged every time a message is sent to the SCU.

The packaging of the data in the application layer, as shown in Figure 3-2, is done as follows for the messages that are sent from the SCU to the PSN. Each message contains the elements listed below:

- **Number of primitives:** indicates the number of primitives that are packaged in this message. If the value is greater than 1, then the message is said to contain a macro.
- **Primitives:** may be one or more. Each primitive contains one or more parameters.

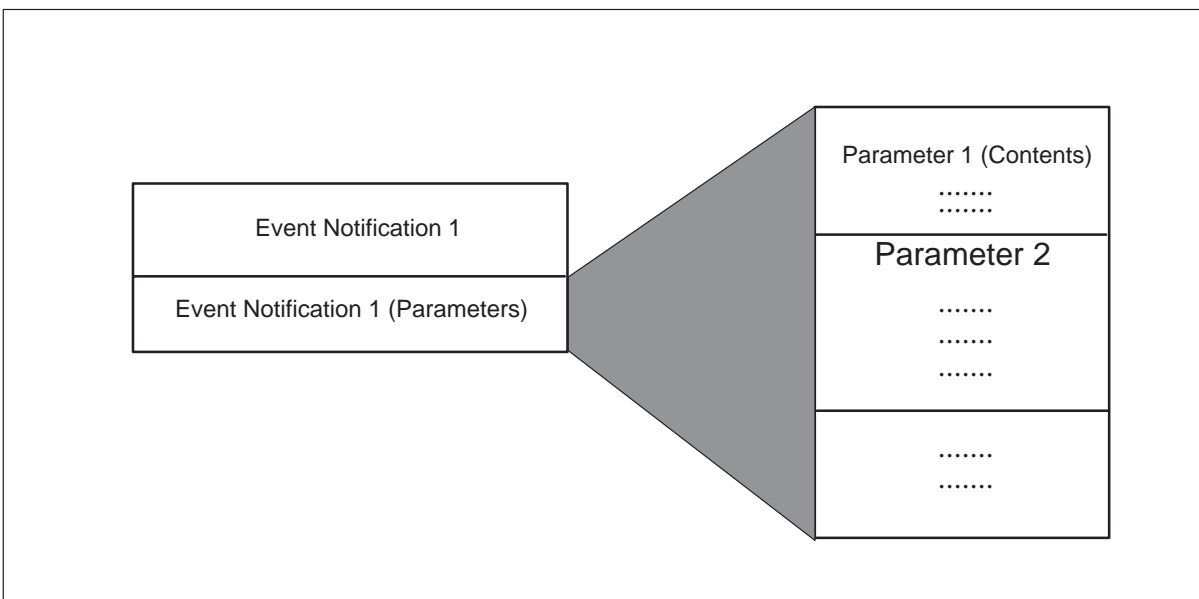
Figure 3-2
Packaging of Primitives/Macros (SCU to PSN)



The messages from the PSN to the SCU contain event notification data, and information is packaged in these messages as explained below. Each message contains the following elements:

- Event notification: only one event notification. Each event notification contains the following information:
 - Number of parameters: indicates the number of parameters sent in the event notification.
 - Parameters: may be one or more. Refer to Figure 3–3.

Figure 3-3
Packaging of Event Notifications (PSN to SCU)



The next section lists primitives and event notification messages. For each primitive and event notification message, the parameters that are sent in the message are also listed.

For each parameter that is listed, the parameter type is given. The parameter type is used to describe the layout of the parameter contents. In other words, It explains in detail how the bits and bytes of each parameter are arranged and used.

Call_Control primitives

A primitive (or a Call control command) is sent to the PSN from the SCU as a high level instruction to control the call flow. A **Call_Control** primitive may affect one or more ports in a call.

Each time a primitive is received from the SCU, the SMSGRCD register in the PSN_USAG OM group as well as the appropriate primitive register in the PSN_PRIM OM group are pegged.

The primitives listed here can be sent from the SCU to the PSN. The list is arranged in alphabetical order.

Bridge

The **Bridge** primitive instructs the PSN to bridge/conference in the specified ports (and optionally a message).

If a message is to be bridged to a number of ports, then the `Message_Info` parameter is passed. The message to be bridged may already be playing to one of the parties or it may be a new one that is yet to be started. If it is already playing on one of the ports, then it is bridged to the other ports. If the message has not yet been started, then the message is bridged and then started.

This primitive (unlike the **Reconnect** primitive) makes use of the conference circuit resource to bridge multiple parties.

Collect_Digits_&_Report

The **Collect_Digits_&_Report** primitive instructs the PSN to collect the specified amount of DTMF digits and report the collected digits to the SCU.

If the port is currently involved in a connection, then it is held. In this case, the voice path between this port and the connected port is cut in both directions, allowing for the collection of digits on this port.

Connect

There are two forms of the **Connect** primitive: **one_party Connect** and **two_party Connect**. A **one_party Connect** is identified by the originating port's Trunk Group Number and Trunk Member Number, being NIL-TRKGRP and NIL-TRKMEM respectively. A **two_party Connect** specifies an active PSN agent (NON-NIL TRKGRP and TRKMEM values) in the originating port parameter.

The **one_party Connect** primitive instructs the PSN to create a PSN call using one specified terminating trunk agency. The terminating agent is specified by the external Trunk Group Number as defined within a table of the Chapter "PSN office parameters". An idle member of the terminating trunk group is identified and the port is seized and becomes a PSN agent. The Signaling information (that's passed in the `Sig_Info` parameter) is outpulsed on the destination trunk member.

The **two_party Connect** primitive instructs the PSN to create a PSN call, using two specified trunk agencies. The originating agent is already serviced by the SCU. The terminating agent is specified by the external Trunk Group Number as defined in the Chapter "PSN office parameters". An idle member of the given trunk group is identified and the port is connected to this trunk member. The Signaling information—passed in the `SigInfo` parameter—is outpulsed on the destination trunk member. Also, voice path in both directions is established for this two-party call.

Disconnect

The **Disconnect** primitive instructs the PSN to disconnect the specified port from the call in which the port is currently active.

Error_Detected

The **Error_Detected** primitive is sent by the SCU to the PSN when the SCU has detected a non-fatal or a non-service-affecting error for a port in the server mode.

Hold

The **Hold** primitive instructs the PSN to put the given port on hold . In other words, the voice path is cut in both directions with the connected party.

Monitor

The **Monitor** primitive instructs the PSN to monitor the specified port for the given digits and/or tones using the specialized tone receiver (STR).

Mute

The **Mute** primitive instructs the PSN to cut and reestablish the voice path between the given port and the connected party in the direction of the port to the connected party. For example, in an A to B call where A and B are talking, Mute (A) cuts the voice path from A to B, but the voice path from B to A is not affected in any way. This means that A can hear B, but B cannot hear A.

If the voice path is already cut in the specified direction, then receiving a Mute primitive will un-mute the connection or reestablish voice path between the two ports.

New_Call_Accepted

The **New_Call_Accepted** primitive instructs the PSN that the **New_Call** event notification was accepted and the port specified in the **New_Call** event notification will be controlled by the SCU with further instructions.

New_Call_Rejected

The **New_Call_Rejected** primitive instructs the PSN that the **New_Call** event notification was rejected and the port specified in the **New_Call** event notification will not be controlled by the SCU. In this case, the port will be routed to the PSNF treatment.

Play_Message

The **Play_Message** primitive instructs the PSN to connect the specified port to a message, which may be an announcement or a tone.

The message being played is uninterruptible by the port. The number of cycles to play (if the message is an announcement) or the tone duration (if the message is a tone) is specified in the “*MessageToPlay*” parameter.

Play_Prompt_Collect_Digits_&_Report (PPCD)

The **Play_Prompt_Collect_Digits_&_Report** primitive instructs the PSN to play a message (an announcement or a tone), and at the same time, initiate digit collection on the specified port. The message is interruptible and as soon as the first digit is dialed, the message stops, and the digit collection on the port continues.

Reconnect

The **Reconnect** primitive instructs the PSN to re-establish a connection between any two ports in a service call.

Set_Billing_Record

The **Set_Billing_Record** primitive instructs the PSN to update the billing records for a specified port with the given information.

This primitive may be received at any time during the call to update the appropriate billing information for the given port.

Stop_Message

The **Stop_Message** primitive instructs the PSN to stop playing a message that is currently being played to one or more ports. To illustrate consider the following:

- if the port was connected to just the message and no other port, then the port is held.
- if the port was connected to other ports besides the message (bridged to several other ports and the message), then it remains connected to the other ports. The only difference being that the bridged ports do not hear the message any longer.

Transmit_Siginfo

The **Transmit_Siginfo** primitive instructs the PSN to transmit the specified signaling information on the given port.

Macros

The SCU may send multiple primitives in a message. These primitive instructions collectively are called macros.

One primitive may, or may not be, completed before the execution of the next. For example, if a macro consists of a “Play Message” followed by a “Connect”, then the message is started on the port. Immediately thereafter, a **Connect** primitive is executed that aborts the message being played and connects the port to another destination trunk member.

The following are some examples of macros.

- If there are five parties in a service call at any given time, then a “take down call” macro can be implemented. This macro consists of five Disconnects, one for each party in the call. Upon execution of this macro, the entire five party call is taken down.
- Another example is the “preempt” macro. Consider a call with parties A and B talking. The goal is for another party, C, to preempt this call, thus leaving B connected to a message, and A talking to C.

This macro can be implemented by sending two primitives: **Reconnect** (A to C), followed by **Play_Message** (B).

For the Call Processing application, the maximum number of primitives that can be sent by the SCU in a macro is five. For the Admin and Audit applications, the maximum number of primitives that can be sent by the SCU in a macro is three. If a macro is sent with more than the maximum allowed primitives, the macro is not processed.

Also, when an incoming message from the SCU contains a macro rather than a single primitive, the SMACRCVD register in the PSN_USAG group is pegged. This is in addition to the SMSGRCVD register in the PSN_USAG that is pegged every time a message is received from the SCU.

Event notifications

The PSN uses event notification messages to notify the SCU of events on the ports that are involved in a Service call.

The events that occur at the PSN may be in response to the SCU primitive/macro instruction or as a result of a peripheral event on the port. An event notification message is used to report one and only one port event to the SCU. Each primitive in a macro will have its own event notification message reported to the SCU.

Also, when an event notification is sent from the PSN to the SCU, an OM in the PSN_NOTF OM group is pegged. This group of OMs has a register for each event notification that is sent to the SCU. In addition, the SMSGSENT register in the PSN_USAG group is pegged for each outgoing message that is sent to the SCU.

The following is a list of event notification messages that may be sent by the PSN to the SCU.

Digits_Collected

The **Digits_Collected** event notification message is reported to the SCU when the PSN has collected the specified amount of digits from the port, an

end delimiter has been dialed by the user on that port, or the digit collection has timed out.

This event notification is sent in response to the following primitives received from the SCU: **Collect_Digits_&_Report** and **Play_Prompt_Collect_Digits_&_Report**.

Error_Detected

The **Error_Detected** event notification message is reported to the SCU when the PSN has detected any one of the following errors:

- an error detected while parsing an incoming message from the SCU
- unavailable resources on the PSN to complete the execution of the received instruction
- internal processing and resource errors on the PSN
- hardware failures
- force releases

This event notification is sent in response to the following primitives received from any SCU primitive.

Note: The **Error_Detected** may also be received from the SCU to report a non-fatal error. For more information on error detection and handling, see the Error Handling section in Chapter “PSN Finite State Machine.”

Instruction_Completed

The **Instruction_Completed** event notification message is reported to the SCU in response to the primitive instructions sent from the SCU.

This event notification is sent in response to the following primitives received from the SCU: **Collect_Digits_&_Report**, **Play_Prompt_Collect_Digits_&_Report**, **Disconnect Hold**, **Mute**, **Reconnect**, **Stop_Message**, **Play_Message Bridge**, **Monitor**, **Transmit_Siginfo**, **Set_Billing_Record**, **Reset_Switch** and **Set_IP_Address..**

Message_Played

The **Message_Played** event notification message is reported to the SCU when the message on the specified port has ended.

This event notification is sent in response to the following primitive received from the SCU: **Play_Message**.

New_Call

The **New_Call** event notification message is reported to the SCU when—subject to the PSN datafill—the call is to be controlled by the SCU.

Off_Hook

The **Off_Hook** event notification message is reported to the SCU when the specified port goes off-hook or answers.

On_Hook

The **On_Hook** event notification message is reported to the SCU when the specified port goes on-hook or is released.

Route_Not_Available

The **Route_Not_Available** event notification message is reported to the SCU when none of the trunk members of the specified terminating trunk group are idle.

This event notification is sent in response to the following primitive received from the SCU: **Connect**.

Route_Selected

The **Route_Selected** event notification message is reported to the SCU when a valid terminating trunk group is identified, a member of that terminating trunk group is found idle, and that member is seized.

This event notification is sent in response to the following primitive received from the SCU: **Connect**.

Signaling_Event

The **Signaling_Event** message is reported to the SCU when a **Signaling_Event** occurs on the PSN for the port. For example, the port receives a signaling message on its peripheral.

Tone_Detected

The **Tone_Detected** event notification message is reported to the SCU when the specified tone/digit is detected on the given port.

This event notification is sent in response to the following primitive received from the SCU: **Monitor**.

Non-Call related primitives & event notifications

The following primitives and associated event notifications do not control the call topology. They are used for general non-call related functions that may or may not affect one or more ports that are controlled by the SCU.

Some examples of non-call related instructions include querying for data on the PSN/SCU, updating IP Address information, and resetting the PSN or the SCU.

Agent_Data

The **Agent_Data** event notification message returns trunk type and signaling information for each agent that is datafilled in the table PSNROUTE. It is sent in response to the **Set_IP_Address** primitive if the “Send Agent Data” field is set to TRUE. The **Agent_Data** event notification is also sent when any agent that is datafilled in table PSNROUTE is added, deleted, or when table TRKGRP, TRKSGRP, or OFCVAR (ORIG_SWITCH_ID field only) information is changed for any agent that is datafilled in the table PSNROUTE.

Current time of the day

The **Current_Time_of_Day** event notification message returns the current time of the day. It is sent in response to the **Query_Time_of_Day** primitive that is received from the SCU, or asynchronously whenever the time is changed on the PSN.

This event notification is sent in response to the following primitive received from the SCU: **Query_Time_of_Day**.

Flow_Control

The **Flow_Control** primitive is sent by the SCU to initiate flow control based on the specified Duration and Gap indices.

Heartbeat

The **Heartbeat** primitive is sent to the PSN from the SCU to indicate that the arbitrator is alive and well. If this message is not received within a predefined time, then the PSN will wakeup and start polling the arbitrator with the **In_Service** message.

In_Service

The **In_Service** event notification message is reported to the SCU when the PSN comes into service. Also, this message is sent to the SCU when the arbitrator heartbeat fails.

Port_Status

The **Port_Status** is reported to the SCU in response to the **Query_Port** request from the SCU. It may also be returned by the SCU in response to the **Query_Port** from the PSN. This message is bi-directional.

The status of the queried port is returned.

This event notification is sent in response to the following primitive received from the SCU: **Query_Port**.

Query_Port

The **Query_Port** instructs the PSN to query the status of a port on the PSN. This instruction may also be sent from the PSN to query the status of a port on the SCU. This message is bi-directional.

Query_Time_of_the_Day

The **Query_Time_of_the_Day** instructs the PSN to return the time of the day. The time of the day is returned by the PSN in the **Current_Time_of_the_Day** event notification message.

Reset_Switch

The **Reset_Switch** primitive is sent to the PSN by the SCU, if the SCU wants to reset:

- all the ports associated with a particular service on a particular SCU (the optional IP Address (to reset) has non-NIL IP Address and non-NIL IP Port)

Note: The optional IP Address (to reset) differs from the Mandatory IP Address (for return). The Mandatory IP Address (for return) is used by the PSN to know where to send the **Instruction_Completed** event notification, with the idea that the SCU that sent the **Reset_Switch** may be in the middle of a restart and therefore would not be able to process the **Instruction_Completed**.

- all the ports associated with a particular SCU (the optional IP Address (to reset) has non-NIL IP Address and NIL IP Port)
- all the ports associated with a particular PSN (the optional IP Address (to reset) has NIL IP Address and NIL IP Port OR there are no optional IP Address (to reset) parameters in the primitive)

This primitive is sent to the PSN when:

- the service on the SCU is in the process of a restart
- the SCU is in the process of a restart
- the SCU decides to purge all existing calls on the PSN

Set_IP_Address

The **Set_IP_Address** primitive instructs the PSN to update the IP address information for the specified port with the given data. This new IP address information will be used to send event notifications to primitives for this port.

This primitive may be received at any time during the call to update the appropriate IP address information for the port.

Stop_Heartbeat

The **Stop_Heartbeat** event notification message is sent in response to a **Set_IP_Address** primitive. This event is sent to the IP Address of the previous SCU to inform the SCU that a new SCU has become the arbitrator for the PSN switch. If the previous SCU IP Address and the new SCU IP Address are the same, no **Stop_Heartbeat** event notification message is sent.

PSN message classifications

The primitives and event notifications that are described above are classified into application classes. The application class defines the application process on the PSN that decodes and executes the primitive. Also, each application process may send an event notification upon processing the primitive. The co-relation between the primitive that is received and the event notifications that are sent to the SCU in reply is explained in detail in section “PSN message flow” of this chapter.

There are four applications that are currently defined for PSN: Call Processing, Admin, Audit, and FCMSG. Briefly, the process functionality is as follows:

- **Call Processing:** this application is responsible for receiving, processing, and sending **Call_Control** primitives and event notifications.
- **Admin:** this application is responsible for receiving, processing, and sending link maintenance messages that affect the PSN as a whole.
- **Audit:** this application is responsible for receiving, processing, and sending port audit messages.
- **Flow Control:** this application is responsible for receiving, processing and, sending flow control messages.

Table 3-1 provides a list of primitives and events that are processed by each of the above applications.

Table 3-1
List of primitives

Application Class	Primitive (SCU → PSN)	Event (PSN → SCU)
Call Processing	Bridge	Digits_Collected
	Collect_Digits_&_Report	Error_Detected
	Connect	Instruction_Completed
	Disconnect	Message_Played
	Error_Detected	New_Call
	Hold	Off_Hook
	Monitor	On_Hook
	Mute	Port_Status
	New_Call_Accepted	Route_Not_Available
	New_Call_Rejected	Route_Selected
	Play_Message	Signaling_Event
	Play_Prompt_Collect_Digits_&_Report	Tone_Detected
	Query_Port	
	Reconnect	
	Reset_Switch	
	Set_Billing_Record	
	Set_IP_Address	
	Stop_Message	
Transmit_Siginfo		
Admin	Error_Detected	Agent_Data
	Heartbeat	Current_Time_of_Day
	Query_Time_of_Day	Error_Detected
	Set_IP_Address	In_Service
		Stop_Heartbeat
Audit	Error_Detected	Error_Detected
	Port_Status	Query_Port
—continued—		

Table 3-1
List of primitives (continued)

Application Class	Primitive (SCU → PSN)	Event (PSN → SCU)
Flow Control	Error_Detected Flow_Control	Error_Detected
—end—		

PSN message flow

The SCU sends the instructions (primitives/macros) to the PSN to control the call flow. In response to these instructions, the PSN sends event notification messages.

The PSN may also send event notification messages in response to peripheral events that occur on a port on the switch to enable the SCU to update its database.

The PSN message protocol is such that when an instruction is sent from the SCU, the PSN responds with a reply for the instruction, either immediately or after completing one or more actions on the PSN. When the PSN sends an event notification message to the SCU, the SCU can send a reply back in response to the notification.

Figures 3-4 through 3-23 show the one-to-one nature of the PSN Messages.

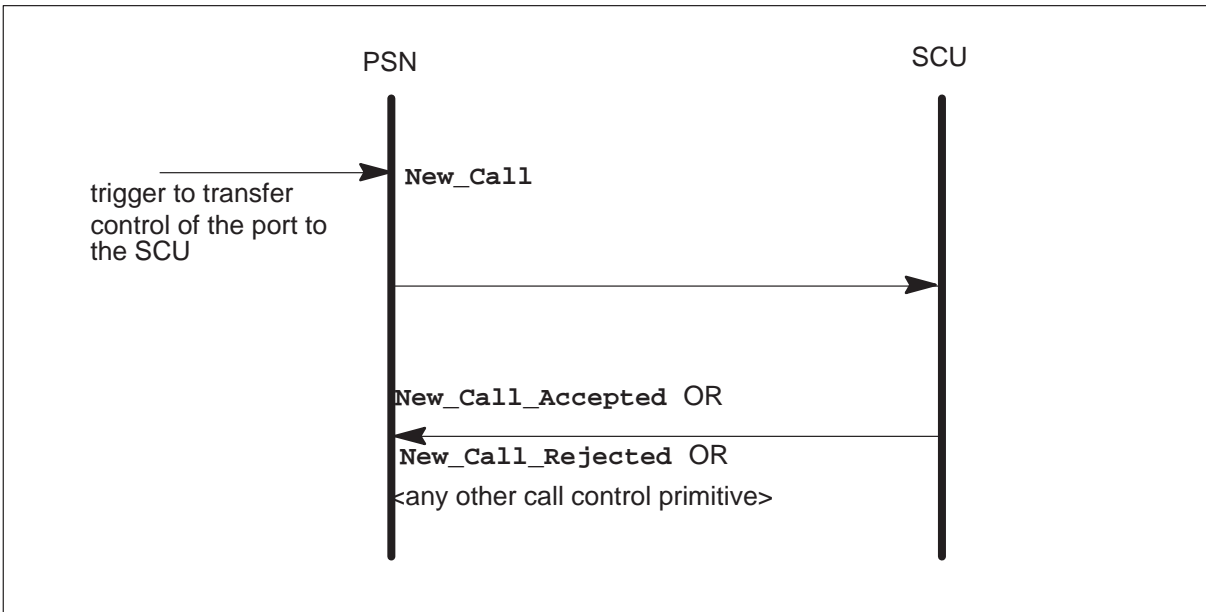
When the PSN determines that the call is to become a service call, for example, the call is to be controlled by the SCU, a **New_Call** event notification message is sent to the SCU. Also, the PSN_EVENT_TIMER is started.

In response to the **New_Call** event notification message, the SCU may send one of the following replies:

- **New Call Rejected:** the SCU does not want to control the call, and wants the PSN to route the call to the PSNF treatment.
- **New Call Accepted:** the SCU has accepted this call and will control the call thereafter.
- **Any other Call_Control primitive:** the SCU has accepted this call and will control the call thereafter.

Upon receipt of any of these messages, the PSN_EVENT_TIMER is canceled.

Figure 3-4
PSN NEW CALL message flow

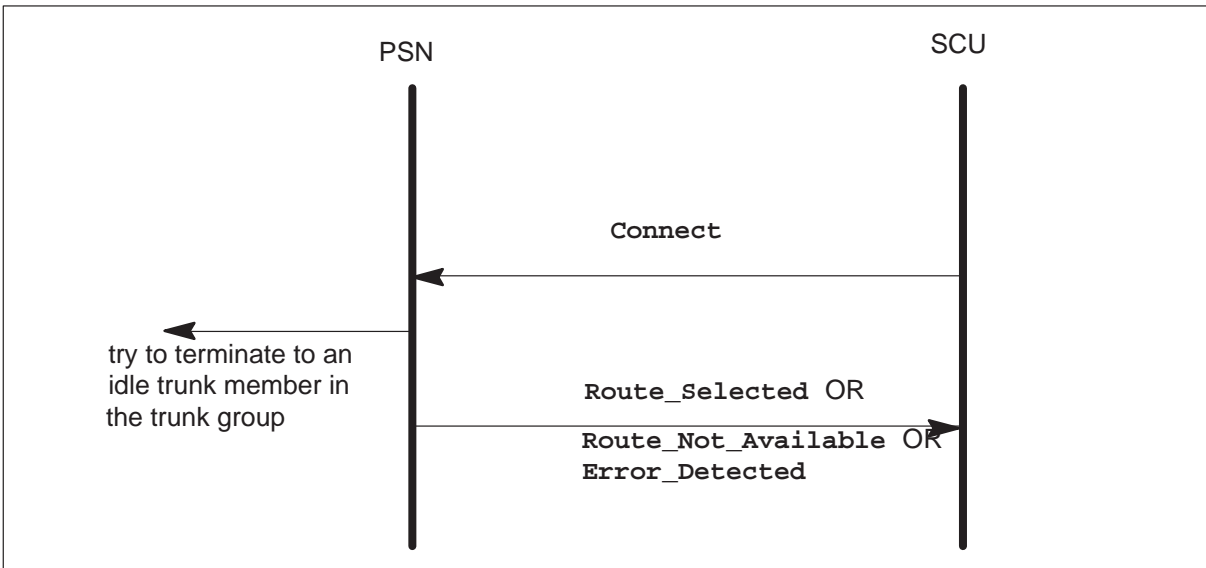


The SCU sends a **Connect** primitive instruction to the PSN when it wants to route a call to an idle trunk member.

As a result of performing this instruction, the PSN may reply with one of the following messages:

- **Route_Selected**: an idle member of the trunk group is located and the call is terminated onto this trunk member.
- **Route_Not_Available**: no member of the trunk group is found to be idle, for example, a terminating trunk member is not found.
- **Error_Detected**: an error is detected, for example, an error in the incoming instruction or an error due to unavailable resources.

Figure 3-5
PSN CONNECT Message Flow

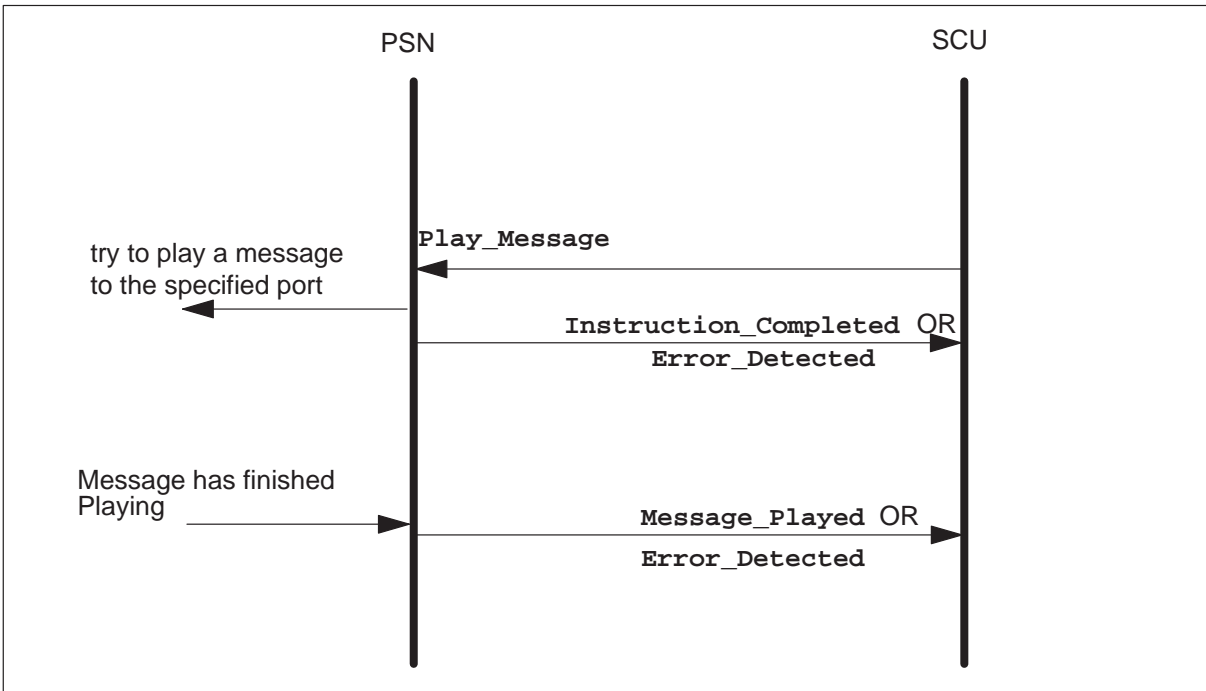


The **Play_Message** primitive instruction is sent to the PSN when the SCU wants to play a message (for example, an announcement or a tone) to a specified port.

The PSN starts playing the message on the port and sends an **Instruction_Completed** event notification to the SCU. When the message has finished playing, the PSN replies with a **Message_Played** event notification.

If for some reason the message could not be played, for example, unavailable resources, then an **Error_Detected** event notification message is sent instead.

Figure 3-6
PSN PLAY MESSAGE Message Flow



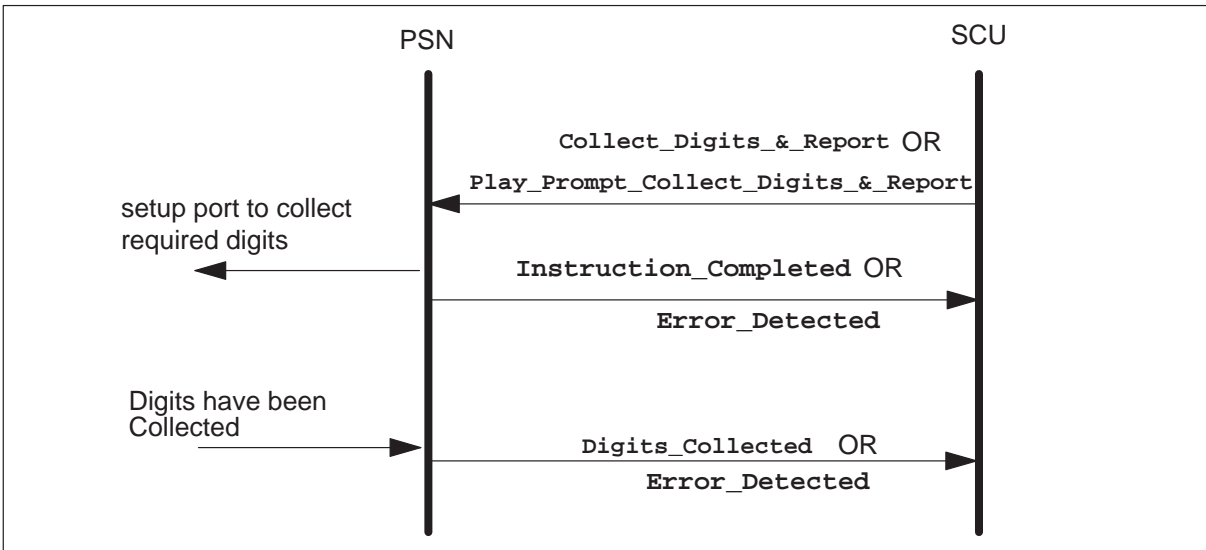
The SCU sends a **Collect_Digits_&_Report**, a **Play_Prompt_Collect_Digits_&_Report** primitive instruction to the PSN when it wants to collect a specific amount of digits on the given port. The latter is sent to play a message, for example, a tone or an announcement before the first digit is collected on the port. The message that is played in this case is hence interruptible.

Upon receipt of either of these primitives, the PSN starts digit collection and sends back an **Instruction_Completed** event notification to the SCU.

After completing these instructions, the PSN may reply with one of the two following messages:

- **Digits_Collected**: the digit collection is successfully started, and some or all the digits are collected, or the digit collection is timed out.
- **Error_Detected**: there is a problem starting digit collection, for example, UTR unavailable.

Figure 3-7
PSN COLLECT DIGITS & REPORT and PLAY PROMPT, COLLECT DIGITS & REPORT Message Flow



The **Disconnect** primitive instruction is sent to the PSN when the SCU wants to disconnect (or remove) a given port from the service call.

The **Hold** primitive instruction is sent to the PSN when the SCU wants to hold a given port.

The **Mute** primitive instruction is sent to the PSN when the SCU wants to mute a given port.

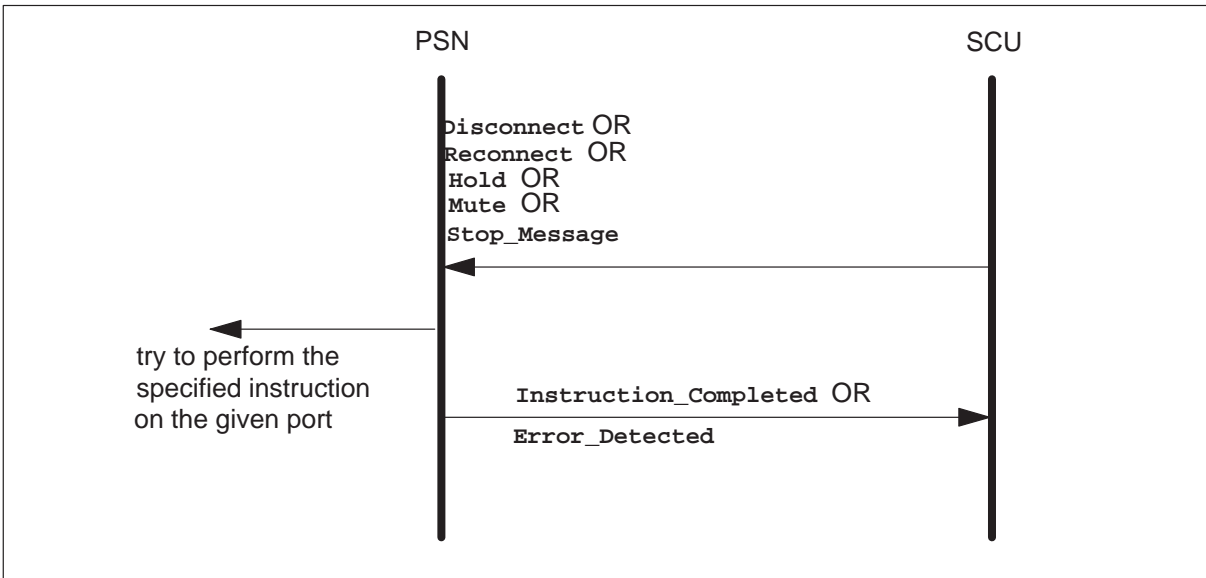
The **Reconnect** primitive instruction is sent to the PSN when the SCU wants to re-establish a connection between any two ports in the service call.

The **Stop_Message** primitive instruction is sent to the PSN when the SCU wants to stop playing the message that is currently being played on the specified port.

On receipt of any one of the above primitives, the PSN has the option of sending one of the following replies:

- `Instruction_Completed`: the received instruction is successfully completed (for example, the port is put on hold successfully).
- `Error_Detected`: an error is detected (for example, the incoming primitive had missing parameters, or resources were unavailable) and the received instruction could not be performed.

Figure 3-8
PSN Disconnect, Reconnect, Hold, Mute and Stop Message: Message Flow



The **Bridge** primitive instruction is sent to the PSN when the SCU wants to bridge (or conference in) several ports (and maybe even a message) in the service call.

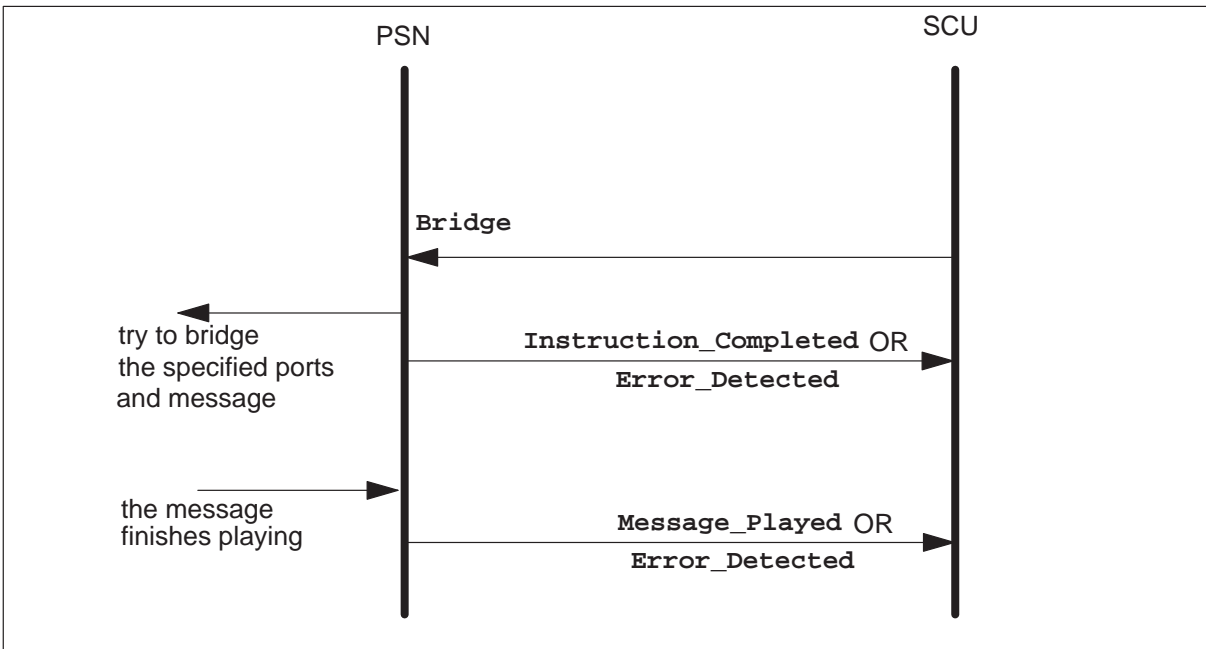
On receipt of the **Bridge** primitive, the PSN has the option of sending one of the following replies:

- **Instruction_Completed**: bridge instruction successfully bridges all the ports (and optionally the message).
- **Error_Detected**: an error is detected, for example, the incoming primitive has missing parameters or resources were unavailable, and the received instruction cannot be performed.

If a message is bridged, then after the message has finished playing, one of the following event notifications is sent to the SCU:

- **Message_Played**: a bridge instruction bridges multiple ports, including a message. This event notification is sent after the message is finished playing.
- **Error_Detected** an error is detected during or after the message has played.

Figure 3-9
PSN BRIDGE Message Flow



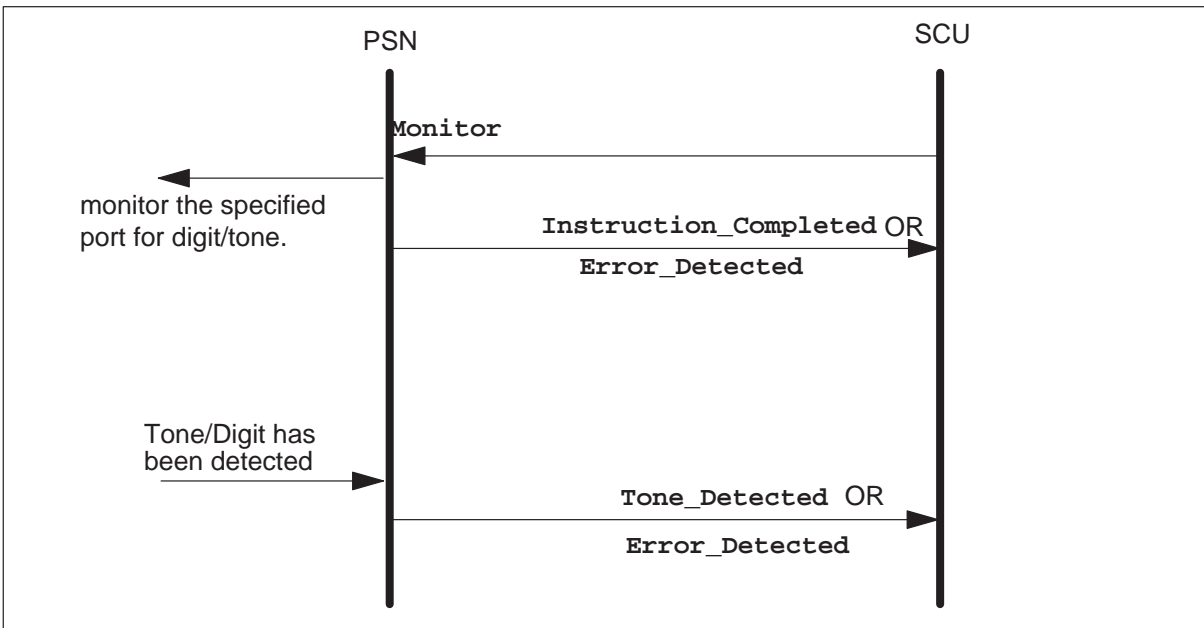
The SCU sends a **Monitor** primitive instruction to the PSN when it wants to monitor the specified port for a given tone and/or digits.

When the PSN receives this primitive, it starts monitoring the port for the specified tone and/or digits and returns an **Instruction_Completed** message to the SCU.

After completing this instruction, the PSN may reply with one of the two following messages:

- **Tone_Detected:** the specified tone and/or digit is detected.
- **Error_Detected:** there is a problem starting tone detection, for example, STR is unavailable.

Figure 3-10
PSN MONITOR Message Flow



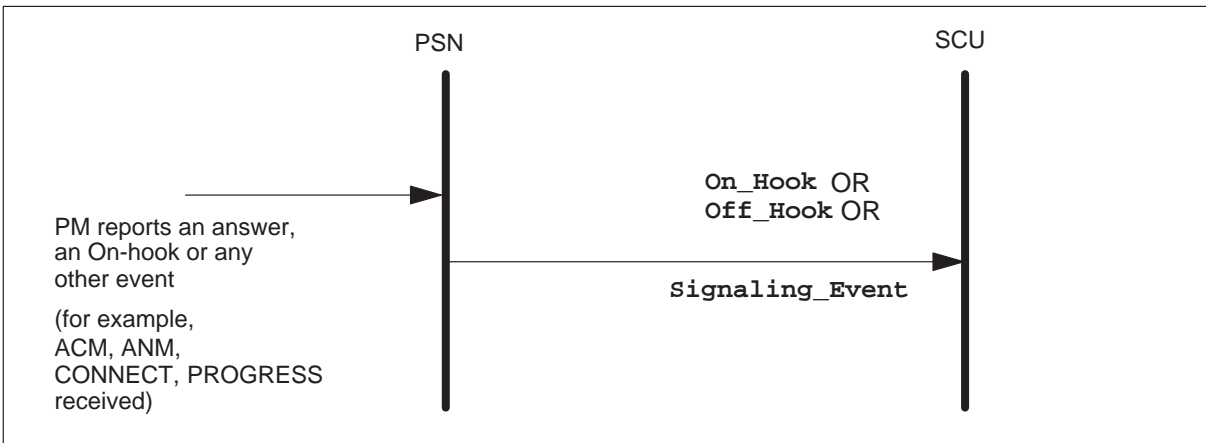
When a port in a service call goes on-hook, the PSN sends the **On_Hook** event notification message to the SCU.

When a port in a service call goes off-hook or answers, the PSN sends an **Off_Hook** event notification message to the SCU.

When a port in a service call receives signaling information on its peripheral from the network, it is reported to the SCU in the **Signaling_Event** notification message.

Upon receipt of any of the above messages, the SCU updates its internal information. The SCU does not send a reply in response to receiving these messages.

Figure 3-11
PSN On-hook , Off-hook, and Signaling Event Message

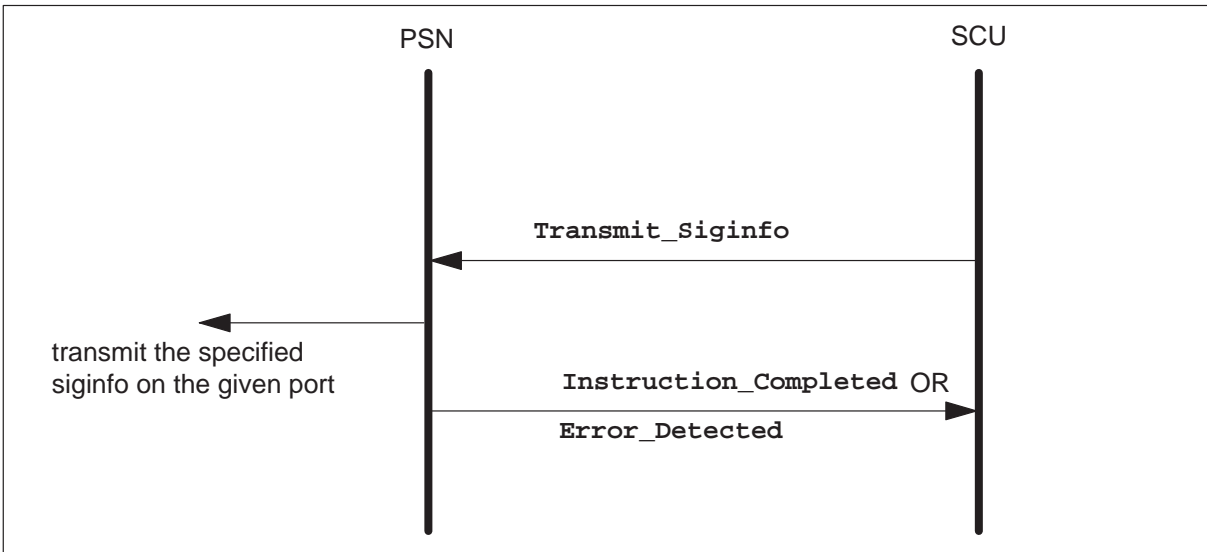


The **Transmit_Siginfo** primitive instruction is sent to the PSN when the SCU wants to transmit some signaling information on a specified port (to be sent over the network).

On receipt of this primitive, the PSN has the option of sending one of the following replies:

- **Instruction_Completed:** the received instruction is successfully completed (for example, the signaling info is successfully transmitted).
- **Error_Detected:** an error is detected, for example, the incoming primitive has missing parameters or resources are unavailable, and the received instruction cannot be performed.

Figure 3-12
PSN Transmit SigInfo Message Flow

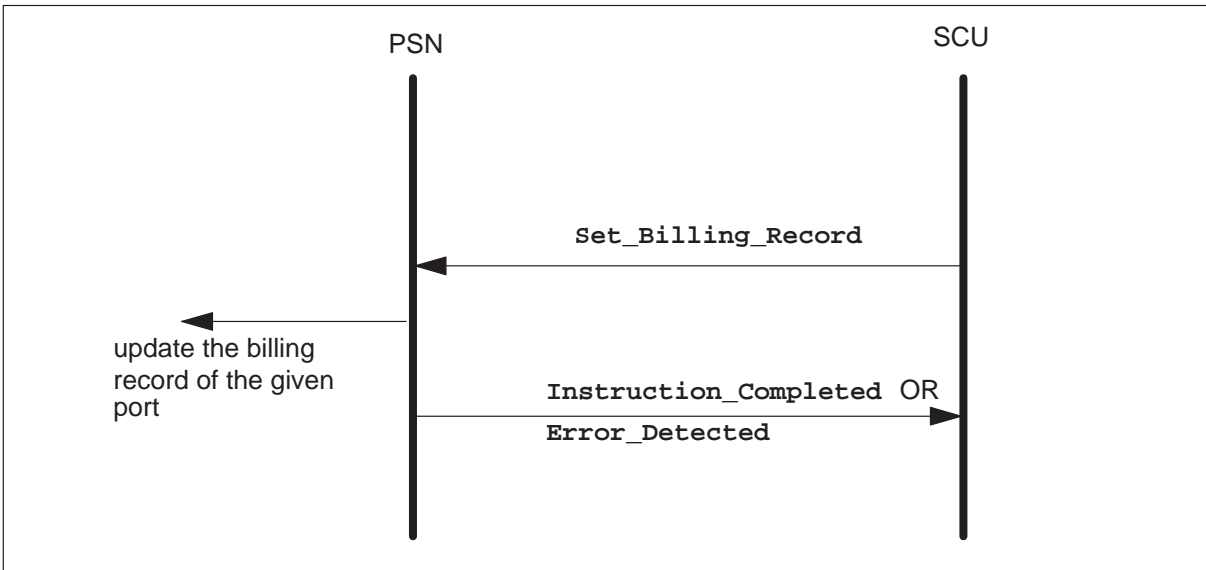


The **Set_Billing_Record** primitive instruction is sent to the PSN when the SCU wants to update the billing record of a given port in the service call.

The PSN updates the billing record associated with the given port and sends an **Instruction_Completed** event notification message to the SCU.

However, if an error is detected, and the incoming request cannot be processed, an **Error_Detected** event notification message is returned to the SCU.

Figure 3-13
PSN SET BILLING RECORD Message Flow



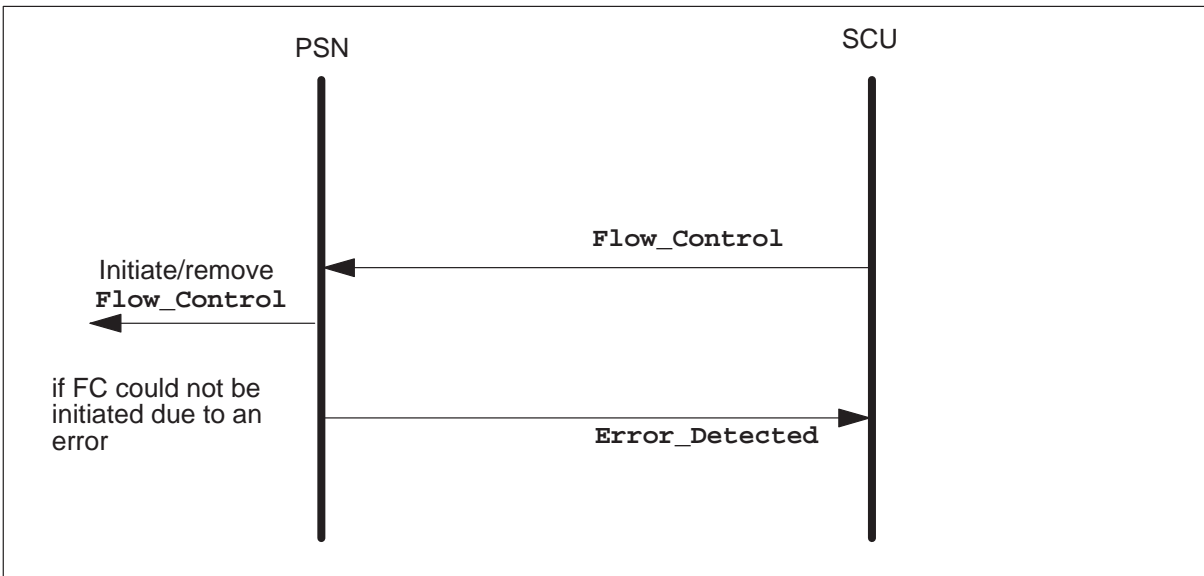
Another way to update the billing record of a given port without sending the above primitive is to include the Billing Info parameter with other primitives that are sent for this port. The Billing Info parameter is optional in the primitives other than the `Set_Billing_Record`.

For example, the Billing Info parameter may be sent in the **Hold** primitive for the port. The port will not only be held, but its billing record will also be updated appropriately.

The **Flow_Control** primitive instruction is sent to the PSN when the SCU wants to initiate **Flow_Control** or when the SCU wants to remove **Flow_Control** previously initiated by itself.

The PSN initiates the **Flow_Control** at successful decode of the primitive. If the **Flow_Control** primitive fails to decode at the PSN, an **Error_Detected** event notification is returned to the SCU.

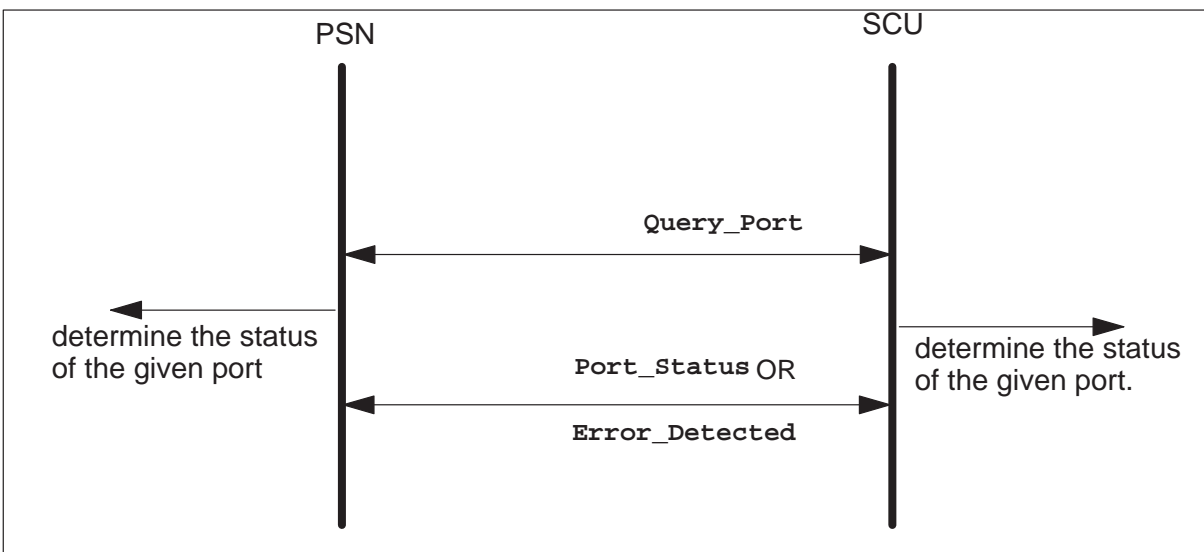
Figure 3-14
PSN Flow Control primitive Message Flow



The **Query_Port** primitive instruction is sent to the PSN when the SCU wants to determine the status of a port on the PSN.

The PSN responds to a **Port_Status** event notification message with the status of the port. However, if an error is detected, and the incoming request cannot be processed, an **Error_Detected** event notification message is returned to the SCU.

Figure 3-15
PSN QUERY PORT Message Flow



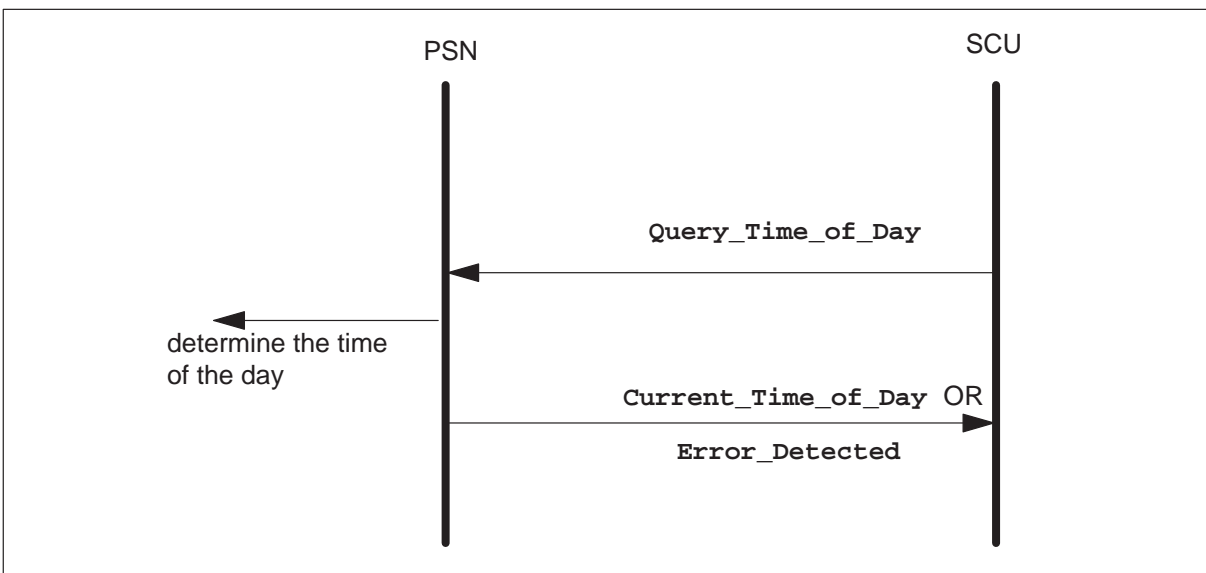
Query_Port may also be sent by the PSN to determine the status of the port on the SCU. The SCU returns the status of the port in the **Port_Status** message.

The **Query_Time_of_Day** primitive instruction is sent to the PSN when the SCU wants to know the time of the day from the PSN.

The PSN responds to a **Current_Time_of_Day** event notification message with the time of the day. However, if an error is detected, and the incoming request cannot be processed, an **Error_Detected** event notification message is returned to the SCU.

The PSN also sends the **Current_Time_of_Day** asynchronously whenever the time is changed on the PSN.

Figure 3-16
PSN QUERY TIME OF THE DAY Message Flow



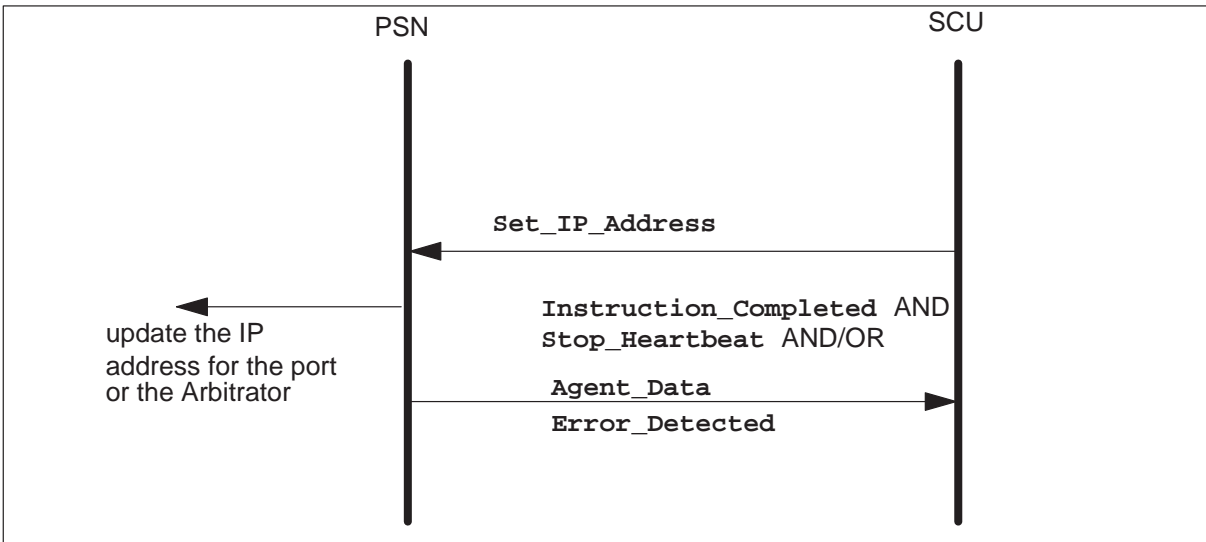
The **Set_IP_Address** primitive instruction is sent to the PSN when the SCU wants to:

- set the IP Address of a given port in the service call.
- set the address of the Arbitrator.
- receive Agent Data information for all agents in table PSNROUTE.

The PSN stores the IP address for the port to be used to send replies and responses to the SCU in the future. The Arbitrator address is used to send the **New_Call** event notification for any port that is to be controlled by the SCU.

After updating the IP address for the port, the PSN sends an **Instruction_Completed** event notification message to the SCU and may send a **Stop_Heartbeat** event notification to the previous SCU IP Address and an **Agent_Data** event notification. However, if an error is detected and the incoming request cannot be processed, an **Error_Detected** event notification message is returned to the SCU and the received message is discarded.

Figure 3-17
PSN SET IP ADDRESS Message Flow



Another way to update the IP address of a given port without sending the above primitive is to include the IP ADDRESS INFO parameter with other primitives that are sent for this port. The IP ADDRESS INFO parameter is optional in most primitives.

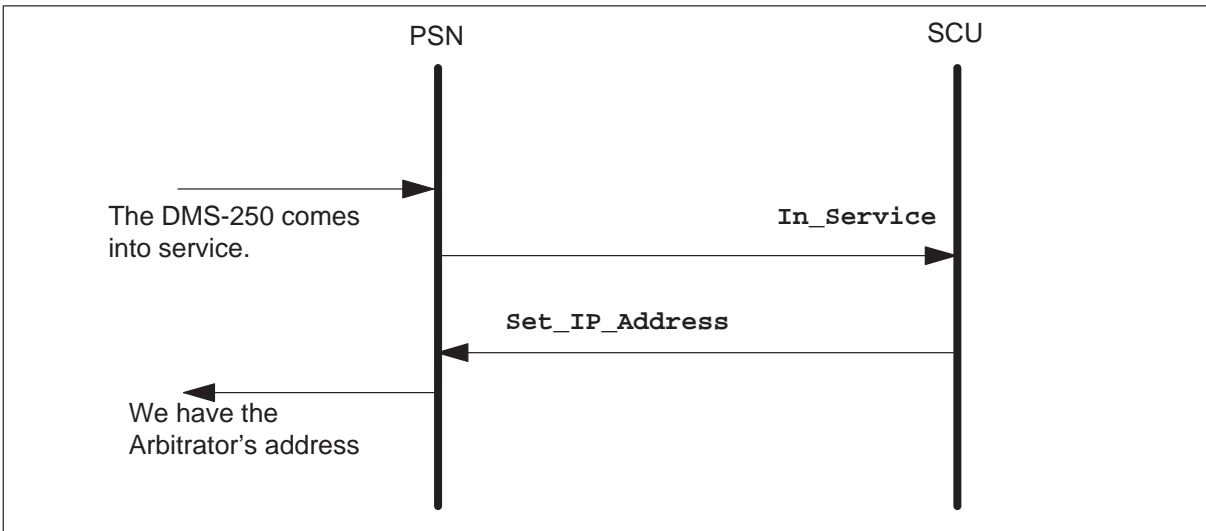
For example, two *IP_Address_Info* parameters can be sent in the **Reconnect** primitive, one for the connected and one for the connecting party. The ports will not only be reconnected, but their return IP address information will also be updated.

When the PSN comes into service, it sends the **In_Service** event notification message to the SCU. This event notification message is also sent when the SCU heartbeat fails or the PSN is just back from a COLD/RELOAD restart.

Upon receipt of this message, the SCU is to return the address of the SCU Arbitrator. The Arbitrator is the initial point of contact for all service call ports. In other words, the **New_Call** event notification message for a port is always sent to the Arbitrator.

Subsequent messages for this port may be routed to the Arbitrator or to a new address. The new address is used if the SCU sends another **Set_IP_Address** primitive to update the IP address of this port.

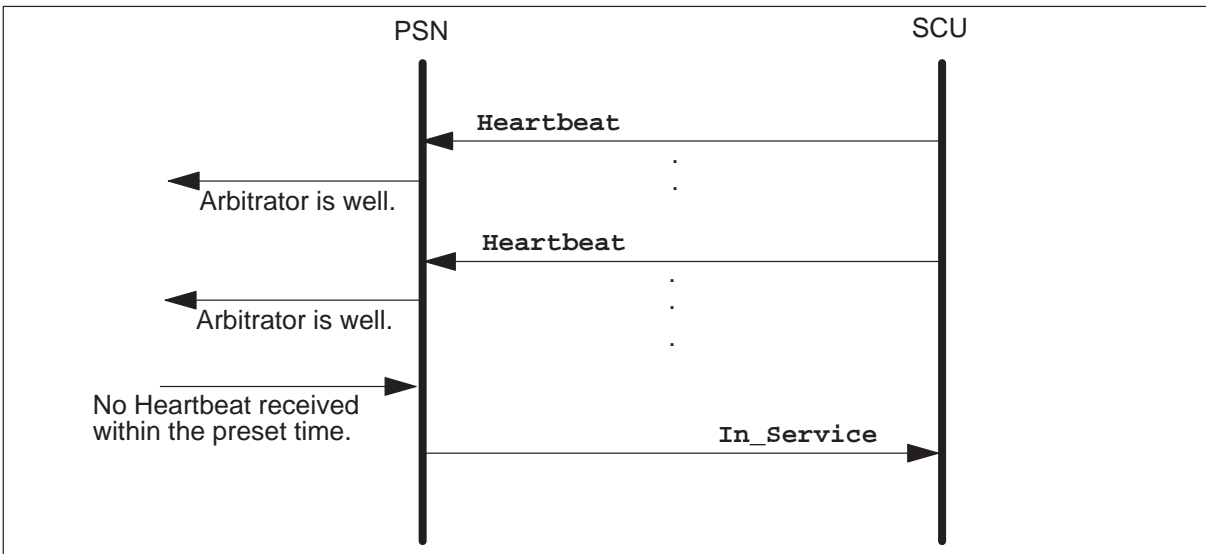
Figure 3-18
PSN IN SERVICE Message Flow



Periodically the SCU sends the **Heartbeat** primitive to the PSN to indicate that all is well with the Arbitrator.

If this message is not received within a preset time on the PSN, a heartbeat failure occurs and an **In_Service** event notification is sent to the SCU to begin polling for the Arbitrator address.

Figure 3-19
PSN HEARTBEAT Message Flow

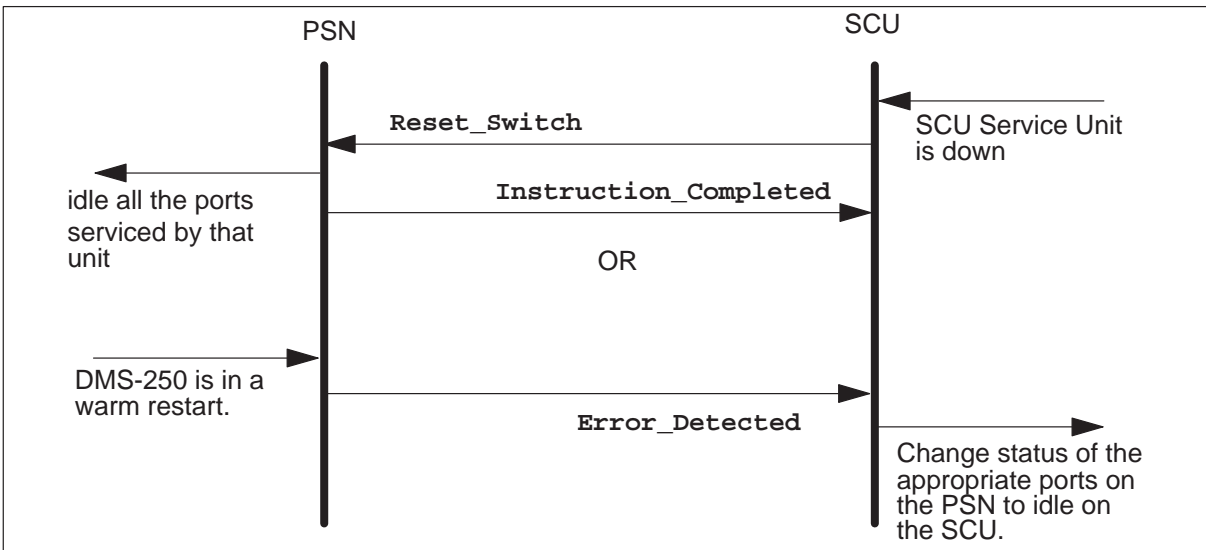


The **Reset_Switch** primitive instruction is sent to the PSN when the SCU wants to reset all the ports that are serviced by certain service providing units on the SCU. The PSN may or may not idle all the appropriate ports and returns an **Instruction_Completed** event notification message to the SCU.

The **Error_Detected** instruction is sent to the SCU after a WARM restart on the PSN when the PSN wants to reset all the PSN ports that are in a non-talking state and are controlled by the SCU. The SCU resets the status of all these ports to idle.

For COLD/RELOAD restarts, an **In_Service_Signalling_Info** event notification is sent to the SCU (with the appropriate restart/reset reason). The SCU marks all the ports that it is currently servicing on the PSN as idle. Also, the PSN takes down, or idles, all the ports that are involved in an SCU call.

Figure 3-20
PSN RESET SWITCH Message Flow



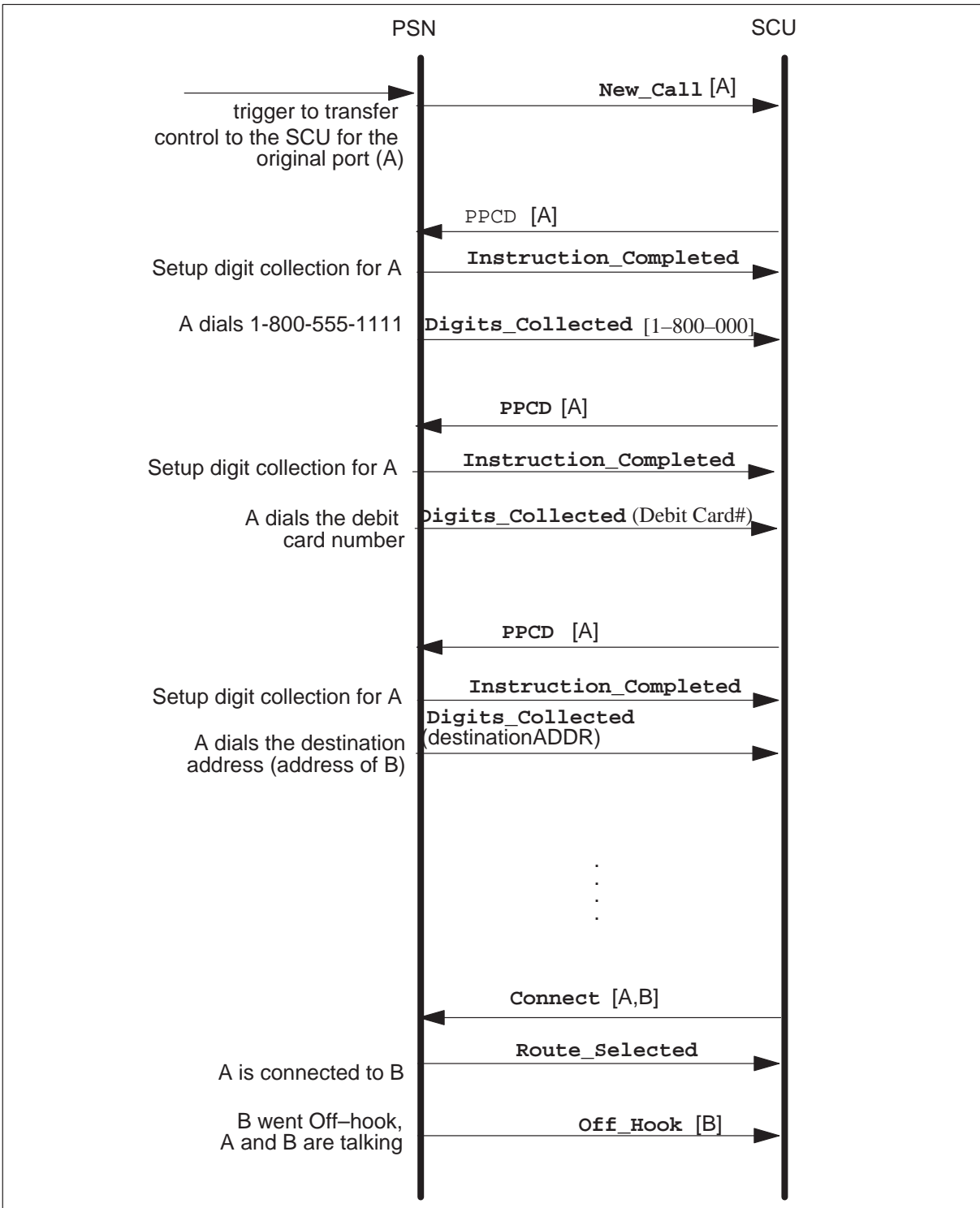
Message flow examples for services

In the previous sections, the primitive/macro instructions and event notification messages and their parameters have been given in detail. The following is an attempt to show how these messages may be used by the SCU to control the call flow on the PSN and in the process implement services.

Debit card is an example of a service that can be implemented using the PSN. The caller places a call with a prepaid debit card. This means that a pre-specified amount of time is assigned to the caller for the call, for which the caller has pre-paid.

When the call duration is nearing the pre-specified amount of time, a message is played to the caller (for example, “you have 5 minutes left on your pre-paid card”). Eventually, if the time expires and the call is still up, the debit card application can issue disconnect instructions for both the parties to take down the call.

Figure 3-21
Message Flow for a DEBIT CARD Service Implementation (Part 1)

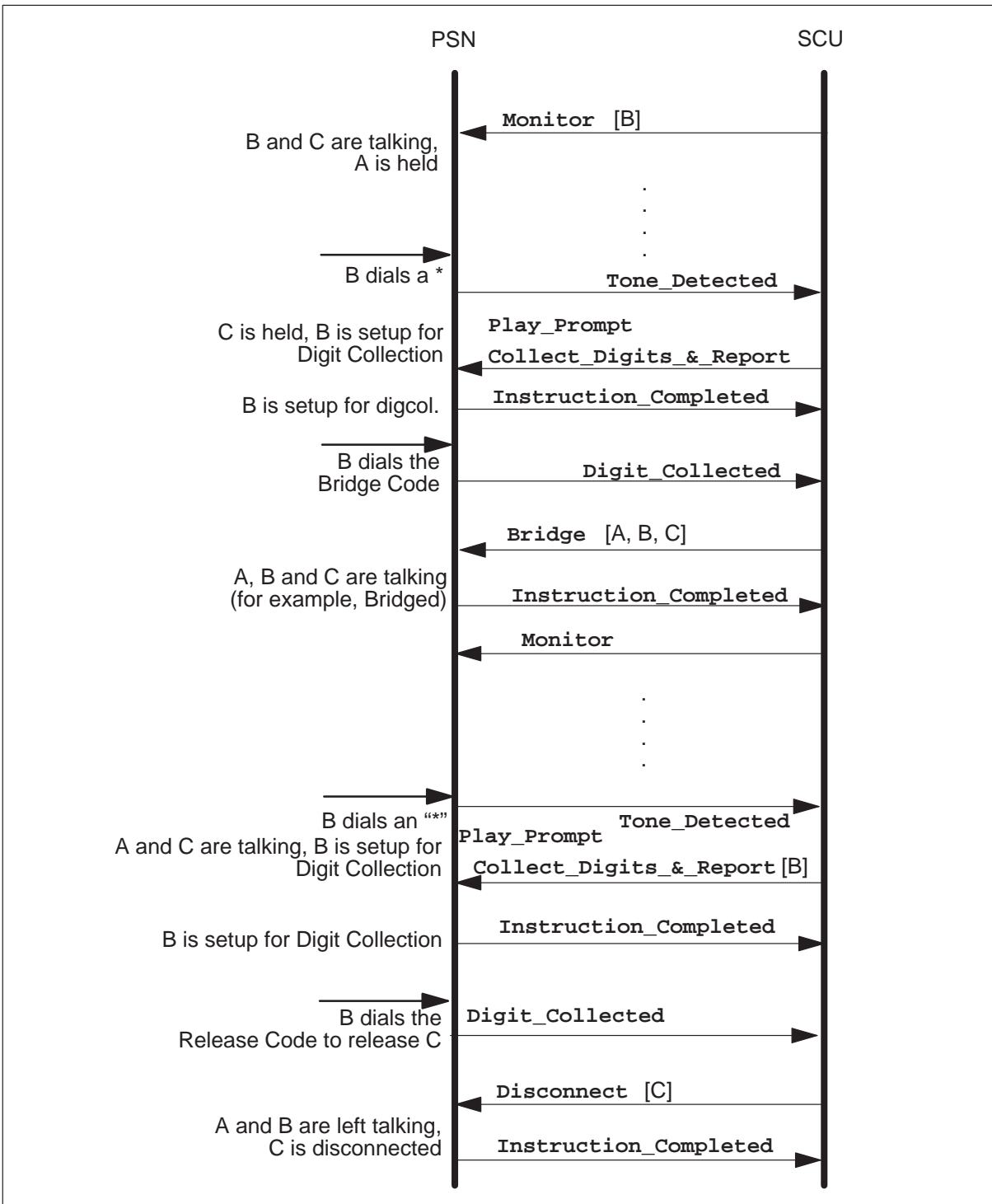


Party B is again monitored for tones. When B dials the necessary digit, for example an “*”, it is reported to the SCU. The SCU may then instruct the PSN to collect digits on party B, using the **Play Prompt Collect Digits & Report** (PPCD) instruction. Digit collection is setup on party B. When digits are collected, they are reported to the SCU.

If the digits correspond to the bridge code, the SCU sends a **Bridge** instruction to the PSN to bridge and conference the three parties, A, B, and C.

If the digits correspond to the release code, the SCU sends a **Disconnect** instruction on party C to the PSN. This disconnects party C from the redirected call and leave A and B in a regular two-party call.

Figure 3-23
Message Flow for a CALL REDIRECTION Service Implementation (Continued)



PSN finite state machine

This chapter contains the detailed description of the PSN Finite State Machine. It describes the relationship and dependencies between the various primitives and event notifications.

The PSN protocol is designed around a Finite State Machine guideline that indicates the states, events, and the allowable transitions between the states. This Finite State Machine forms the basis for the initial as well as the subsequent enhancements to the protocol.

The PSN receives instructions from the SCU for a particular port and performs these instructions on that port. The results of performing these instructions may involve sending reports and event notifications back to the SCU. An instruction may or may not be valid depending upon the state of the port at that time. This is best described using a Finite State Machine.

A port or an agent is defined as a trunk member. The port state may be described as a combination of two states: Agent state and Connection state.

The Agent state corresponds to the state of an agent and a port. The Connection state is the state of the connection in which the agent and port are involved. The primitive instructions that are received from the SCU are processed based on these states.

Agent state

The Agent state is the state of the port and agent. It represents the state of the port without describing its connection status.

The following Agent states are defined: IDLE, SEIZED, and ANSWERED.

- The Agent state is IDLE if the trunk member is idle.
- The Agent state is SEIZED if the trunk member is seized and On-hook.
- The Agent state is ANSWERED if the trunk member is seized and Off-hook (or answered).

Connection state

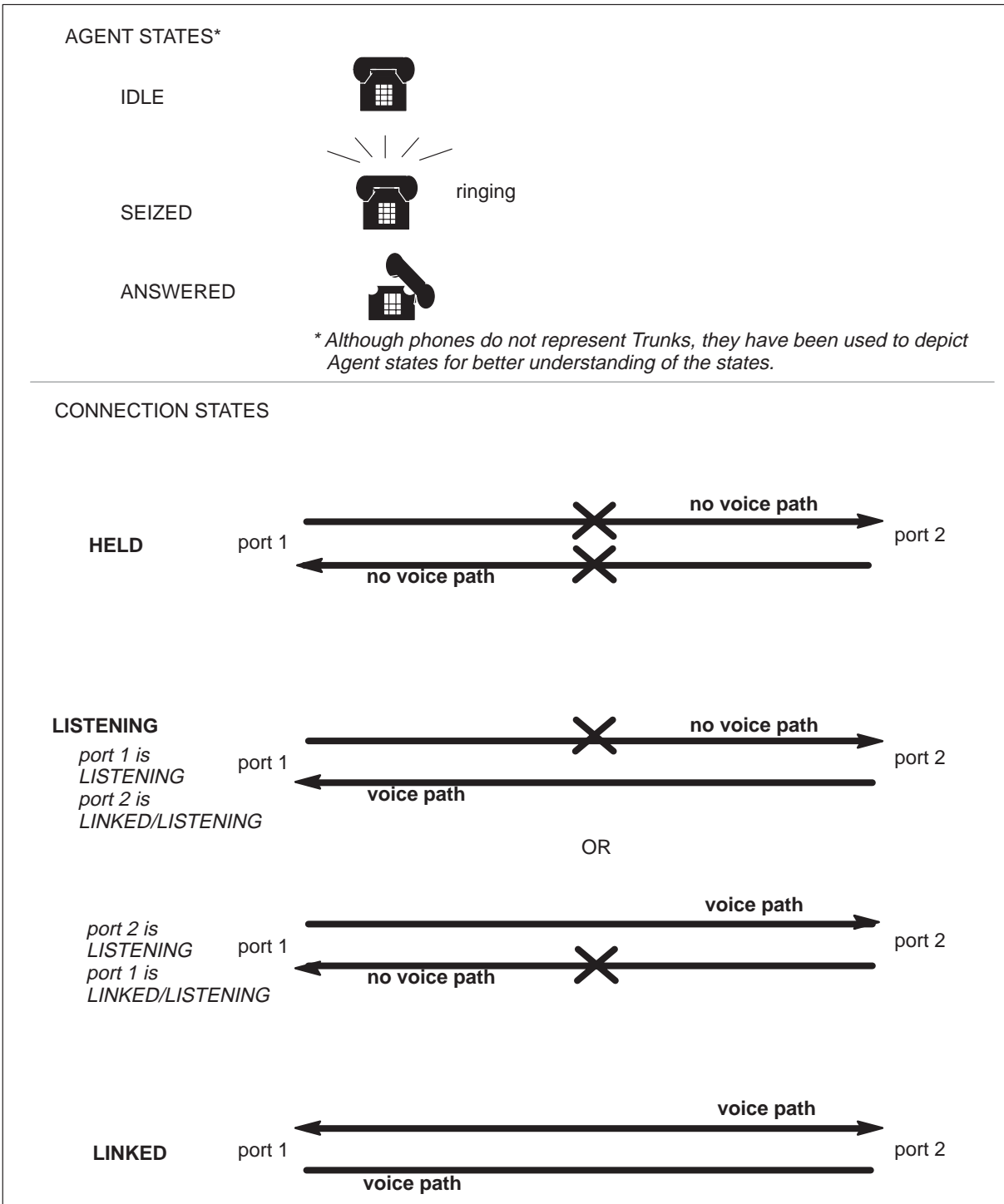
The Connection state is the state of the connection that the agent and port is currently involved in.

The following Connection states are defined as HELD, LISTENING, and LINKED.

- The Connection state is HELD if the connection has no voice path in either direction.
- The Connection state is LISTENING if the connection has voice path in one direction, but not the other.
- The Connection state is LINKED if the connection has voice path in both directions.

Figure 4-1 provides examples of PSN, Agent state, and Connection state.

Figure 4-1
PSN agent and connection states



The Finite State Machine

The Finite State Machine for the PSN is shown below. This Finite State Machine applies to any party in a service call (that is, the originator, terminator, and so on). Hence, each port has an instance of the Finite State Machine running on its behalf.

The state changes depend upon the port. For example, if the originating port is in an Answered and/or a Held state, then upon receipt of a **Connect** primitive from the SCU, the originating port state transitions from the Answered and Held to the Answered and Linked state, and the terminating port state transitions from the Idle and Held state to the Seized and Linked state. The terminating port remains in the Seized and Linked state until the party answers, at which time, the state of this port is changed to Answered and Linked.

The following section describes the state transitions in detail with an example or a call walk through.

The Finite State Machine has been shown in three figures. Figures 4-2 and 4-3 show the explicit and implicit state transitions in the PSN upon receipt of the SCU primitives. Figure 4-4 shows the state transitions upon receipt of peripheral (XPM) events. All three figures must be used to determine the complete port state transitions.

Figure 4-2 shows the explicit state transitions for ports. In this figure, a primitive can be received from the SCU on a port when the port is in any port state (which is the combination of agent and connection states). Upon receipt of the primitive, one of the following may happen:

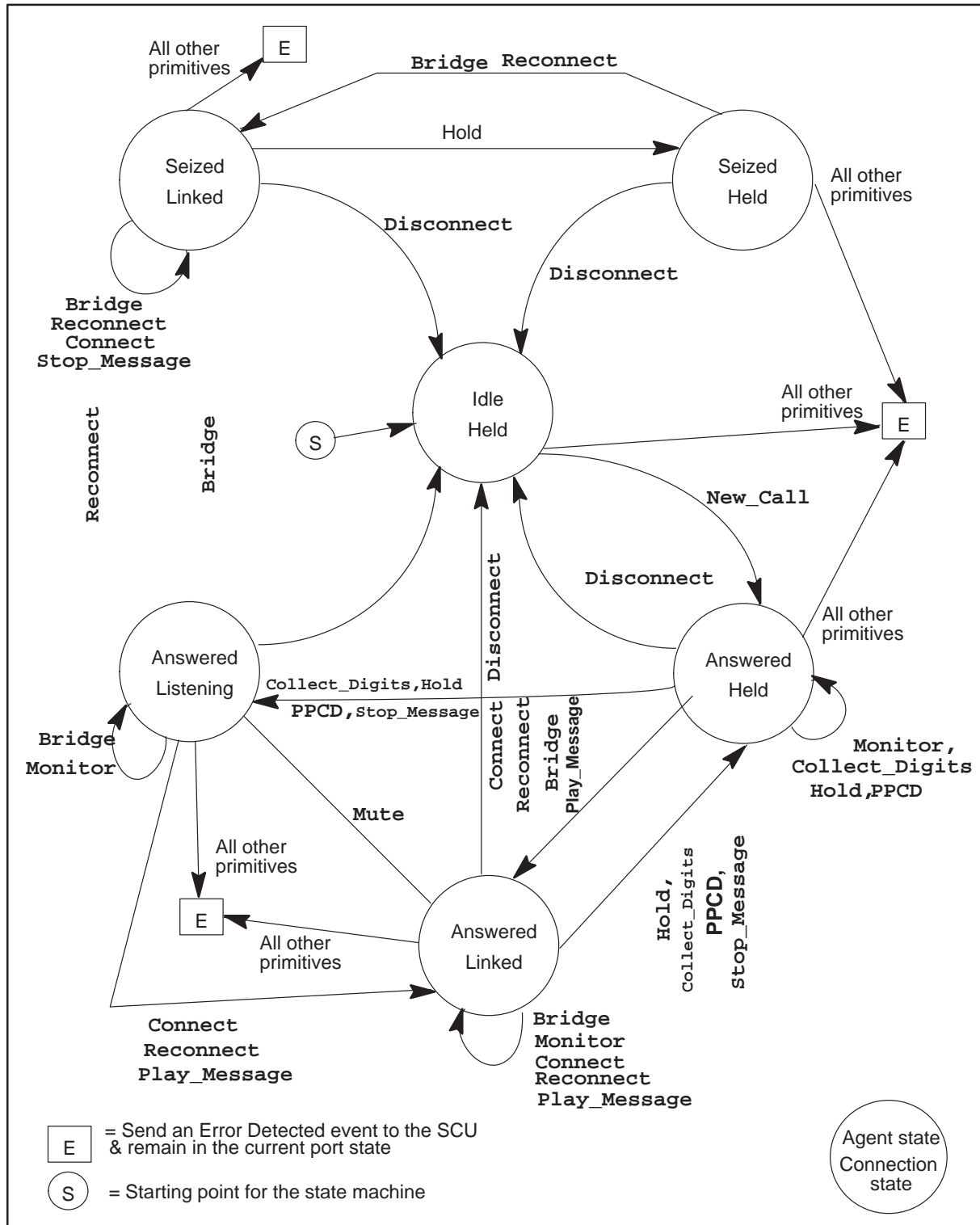
- The primitive is valid for the port in its current state. Hence, the primitive is executed in this state. As a result of this execution, there may or may not be a state change for the port.
- The primitive is invalid for the port in its current state and is not executed. The port remains in its current state. The SCU is notified of the error that was detected in the **Error_Detected** event notification message.

Note 1: The **Set_Billing_Record** and **Transmit_Siginfo** primitives do not cause any state changes and are valid in all but the Idle and Held state. Hence, they are not shown in the Finite State Machine diagram that follows.

Note 2: The **Query_Port** primitive is valid in all the states and does not cause state changes. Hence, it is not shown in the Finite State Machine diagram that follows.

Note 3: The **New_Call** is an event notification rather than an SCU primitive. Nevertheless, it is shown in the following figure to show the first state transition out of the Idle and Held state.

Figure 4-2, PSN finite state machine for the SCU primitives (implicit state transitions)



The Figure 4-3 shows implicit state transitions for ports.

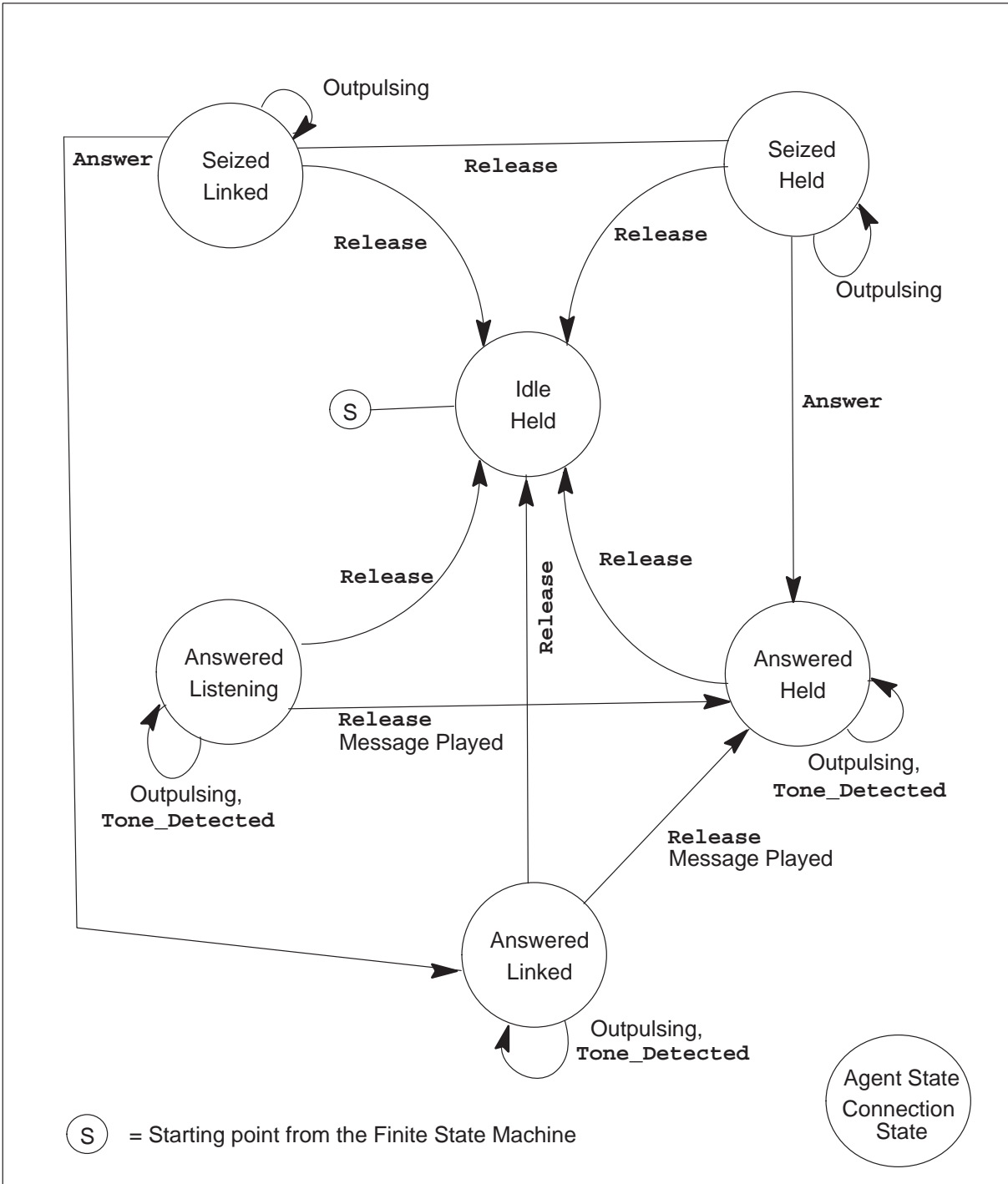
The primitives, such as **Reconnect**, **Bridge**, **Stop_Message** cause implicit state transitions. For example, in a two-party call where party's A and B are talking, a **Disconnect** on A causes an explicit state transition from the Answered or Linked state to the Idle or Held state for party A. The previous figure illustrates this point.

However, the **Disconnect** primitive causes an implicit, or indirect, state transition for B from an Answered or Linked state to an Answered or Held state.

The third part of the Finite State Machine describes the state transitions upon receipt of peripheral events from the peripheral on the PSN. These events can broadly be classified into the following categories.

- **Release:** any PM event that indicates that the port has been released, disconnected, or idled.
- **Answer:** any PM event that indicates that the port has answered or gone off-hook.
- **Digit_Collected:** any PM event that reports the digits that were collected on the port.
- **:Outputting:** any PM event that indicates that digits and/or other signaling information is being transmitted or received on this port.
- **Tone_Detected:** any PM event that reports the tone or digit that was detected on the port.
- **Message_Played:** any PM event that reports that the announcement or tone connected to a port has finished playing.

Figure 4-4, PSN finite state machine for the peripheral events



Call walk through

Figures 4-5, 4-6, and 4-7 shows an example of a call walk through.

The Finite State Machine that was described in the previous section applies to all the parties and ports involved in the service call. Hence, each port in the call has its own instance of the Finite State Machine, and the current state and the state transitions for one port vary when compared to the other ports in the call.

The following text describes a two-party call controlled by the SCU. The state transitions for the originating party (A) and the terminating party (B) are shown below.

The originating party is A. During the course of call setup, it is determined that the call is to be controlled by the SCU, and a **New_Call** event is sent to the SCU. The SCU collects digits, determines a route, and terminates the call (to party B). Next the SCU instructs the PSN to perform some actions, such as **hold** and **Mute**, on party A. Finally, party B goes **On_Hook**, and party A is connected to an announcement or tone. After the said announcement or tone completes, the SCU takes party A down.

Figure 4-5, example call walk through (Part 1)

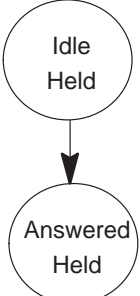
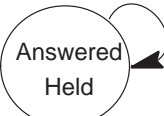
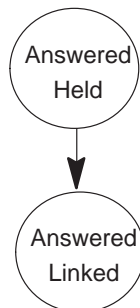
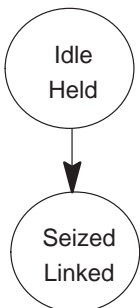
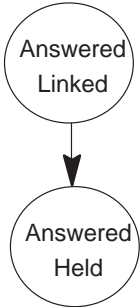
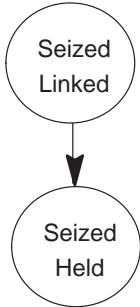
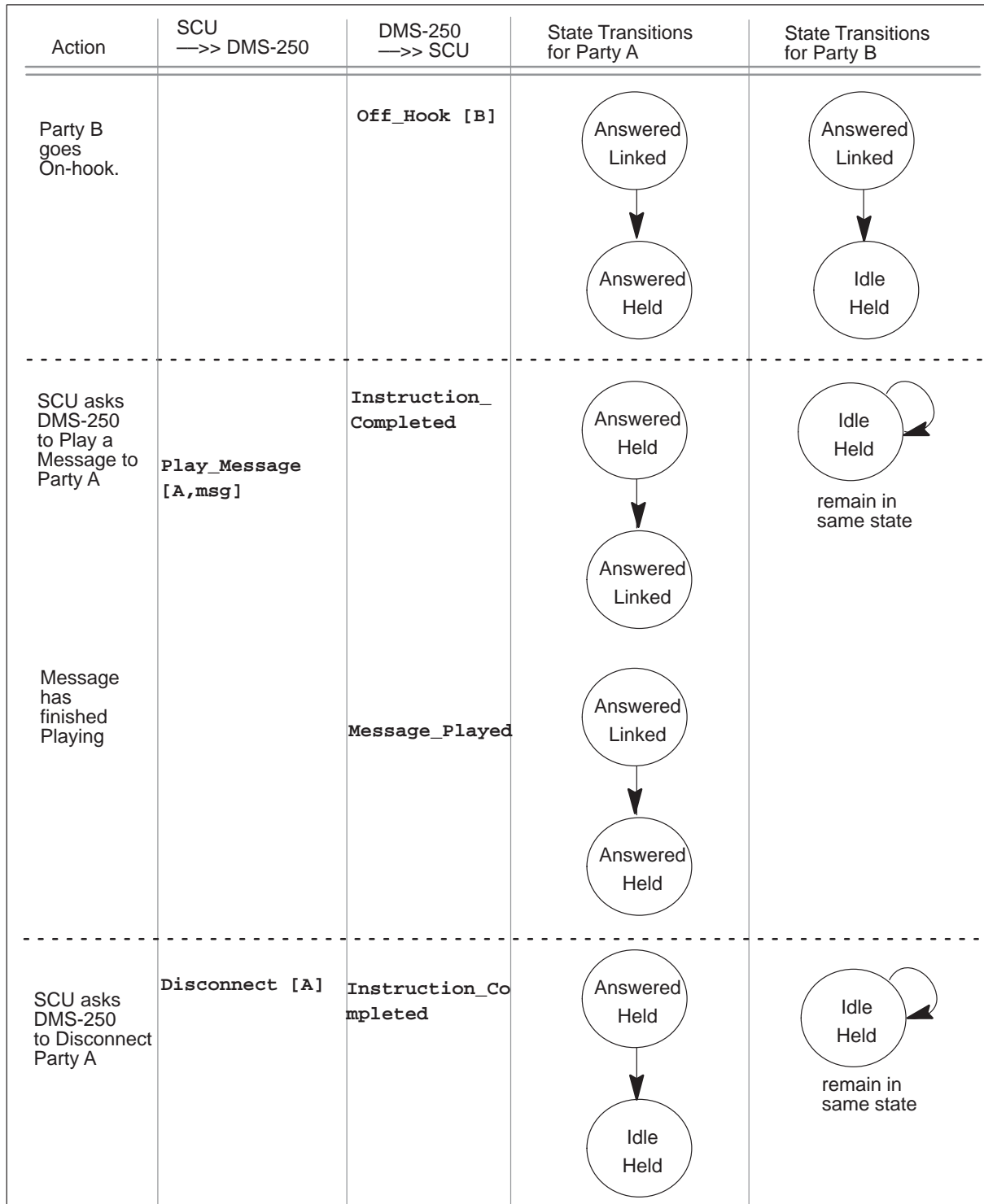
Action	SCU —>> DMS-250	DMS-250 —>> SCU	State Transitions for Party A	State Transitions for Party B
The Call triggers. The SCU is notified of a New Call event.	New_Call_Accepted [A]	New_Call	 <pre> graph TD A((Idle Held)) --> B((Answered Held)) </pre>	
SCU asks DMS-250 to collect digits on party A	Collect_Digits_&_Report [A]	Instruction_Completed Digits_Collected	 <p>remain in same state</p>	
SCU asks DMS-250 to Connect Party A to Party B	Connect [A,B]	Route_Selected	 <pre> graph TD A((Answered Held)) --> B((Answered Linked)) </pre>	 <pre> graph TD C((Idle Held)) --> D((Seized Linked)) </pre>
SCU asks DMS-250 to Hold Party A	Hold [A]	Instruction_Completed	 <pre> graph TD E((Answered Linked)) --> F((Answered Held)) </pre>	 <pre> graph TD G((Seized Linked)) --> H((Seized Held)) </pre>

Figure 4-6, example call walk through (Part 2)

Action	SCU —>> DMS-250	DMS-250 —>> SCU	State Transitions for Party A	State Transitions for Party B
Party B answers or goes Off-hook	Reconnect [A,B]	Off_Hook [B]		
SCU asks DMS-250 to Reconnect Parties A and B		Instruction_Compiled		
SCU asks DMS-250 to Mute Party A	Mute	Instruction_Compiled		
SCU asks DMS-250 to Unmute Party A	Mute	Instruction_Compiled		

Figure 4-7, example call walk through (Part 3)



General rules and restrictions for using primitives

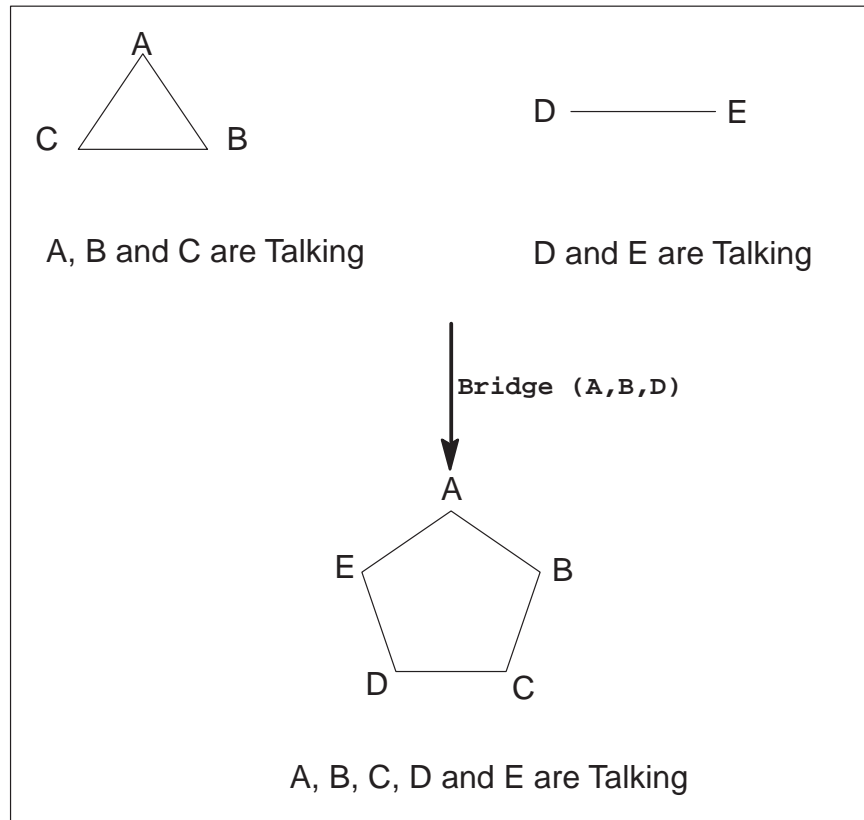
Some of the primitives are straightforward to use. Others, have rules and restrictions that are not so obvious, but if ignored, can cause errors in primitive execution, which causes the primitive to be discarded. Also, there are primitives that affect the call topology in one way if the port is involved in a two party connection, whereas the effect is totally different if the port is involved in a multi-party connection.

1 Bridge

- a. Bearer Capability Screening for ports in a bridged call will not be supported. Refer to Terminating Agent Bearer Capability (BC) Screening in this chapter for details.
- b. Updating IP addresses and Billing Records is not possible using the **Bridge** primitive. The IP Address Info and Billing Info parameters are not included in the **Bridge** primitive.
- c. A **Bridge** primitive that is received on a port that is presently involved in a connection applies to all the parties in that connection. This implies that all the parties in that connection are bridged.

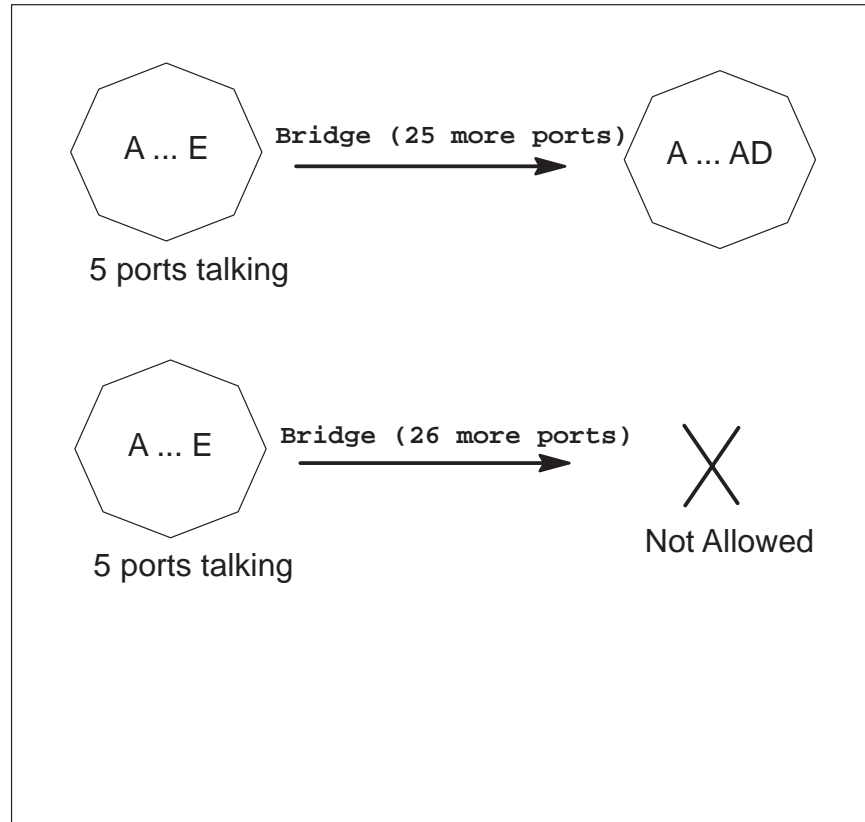
- d. The tone durations of 1 and 2 seconds are invalid for the **Bridge** primitive.

This is shown in the following diagram.



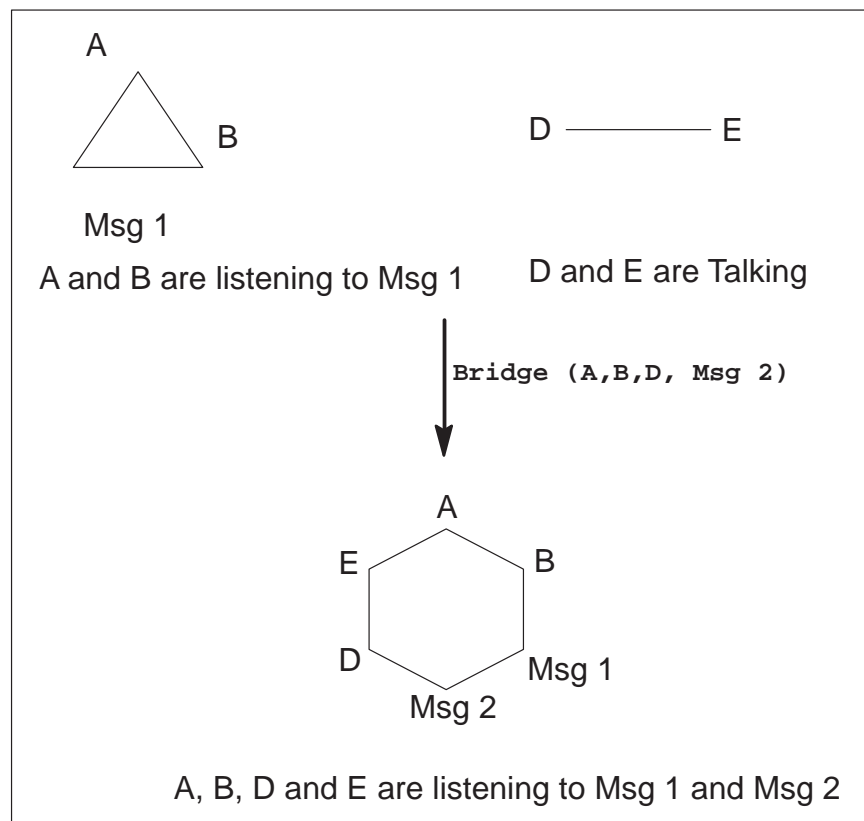
- e. All the ports to be bridged have to be non-idle. Also, all ports to be bridged must be involved in this or another service call.

- f. The maximum number of ports that may be bridged is 30. This applies to new as well as existing parties in the call. This is shown in the following diagram.



- g. Performing a **Bridge** primitive with a large number of agents (>10) is not recommended without the minimum CPU configuration (as defined in the "Restrictions and Limitations" section of this chapter). Without the minimum recommended CPU capacity (and depending on traffic load) the **Bridge** primitive with more than 10 agents in the primitive may exceed CPU usage limits and fail.

This is shown in the following diagram.

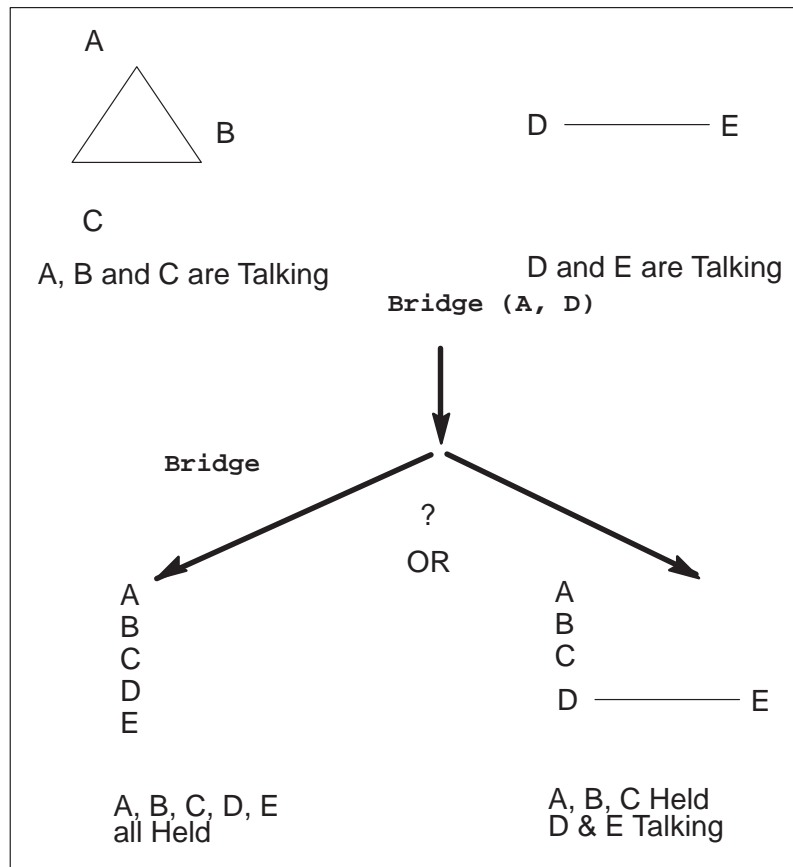


- h. If a PM containing a conference circuit or an individual conference port abnormally exits on a PSN conference, for example, by busying the CTM/MTM, by holding the conference circuit, or by force releasing a conference port at the MAP level, the entire PSN conference is held. In other words, all PSN agents on the conference are held from the Bridge and any messages on the conference are terminated.
- i. If an Announcement or Trunk (PSN agent) abnormally leaves a PSN conference, for example, the Announcement or Trunk is force released at the MAP level, it is processed as if the Announcement or Trunk had simply terminated, timed-out, or disconnected as normal. The resource is removed from the conference and the conference is optimized.
- j. When performing a **Bridge**, the previous topologies of each agent in the **Bridge** primitive are not saved. Therefore if an error occurs (for example, no idle conference circuits are available, or software resources unavailable), the resulting topology of each PSN agent in the **Bridge** is not guaranteed. The agent may be held, left in its previous topology, or conferenced (as the **Bridge** requested).

For example, consider a scenario where three parties (A, B, and C) in a conference are bridged to a call with two parties that are talking (D & E). A **Bridge** primitive is received from the SCU to bridge A, B, C, D, and E.

The bridging process may encounter an error during the execution of the primitive. This causes an **Error_Detected** to be sent to the SCU and the **Bridge** primitive is discarded by the PSN. The call topology may contain one or more parties in a Held state. Although the previous topologies are not retained by the PSN, the resulting state for each agent is sane.

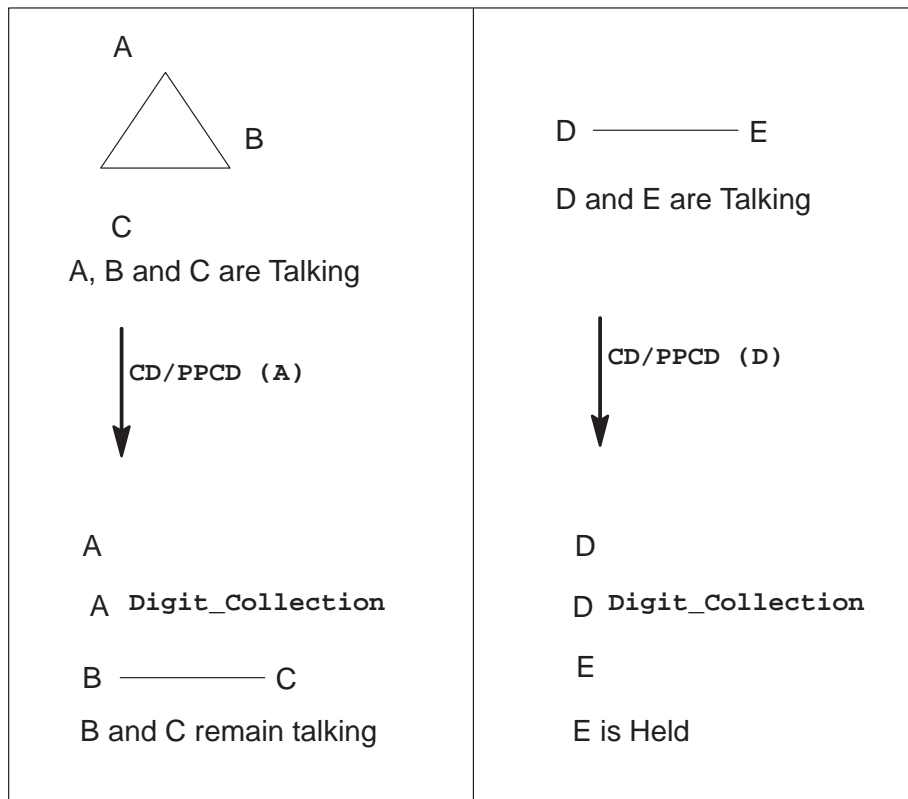
This is shown in the following diagram.



- 2 Collect_Digits_&_Report (CD), and Play_Prompt_Collect_Digits_&_Report (PPCD)

- a. When received on a port that is involved in a two-party connection, these primitives put the given port and any connected port on hold. For example, if A is talking to B and a PPCD is received on A, then digit collection starts on A and B is put on hold. If A, B, and C are talking and a PPCD is received on A, then digit collection starts on A, but B and C remain in a talking state.

This is shown in the following diagram.



- b. The **CD**, **PPCD** primitives and the **Monitor** primitive are mutually exclusive.

For example, if digit collection (primitive **CD/PPCD**) is in progress on A and a **Monitor** is received on A, then this stops the digit collection and tone/digit monitoring starts. Similarly, if the tone/digit monitoring is in progress on A and a **CD/PPCD** is received on A, then the monitoring for the tone/digit stops and the digit collection begins.

When **CD/PPCD** is stopped, a **Digit_Collected** event notification message is sent to the SCU with the digits collected so far (if any), indicating to the SCU that the digit collection is aborted.

When **Monitor** is stopped, a **Tone_Detected** event notification message is returned to the SCU, indicating that no tone was detected and that the **Monitor** was aborted.

- c. The receipt of any valid primitive after a **CD/PPCD**, aborts the digit collection and the execution of a received primitive begins. Also, a **Digits_Collected** event notification message is sent to the SCU with the digits collected so far (if any).
- d. When digit collection is aborted by another non-digit collection primitive that does not result in the PSN agent disconnecting (for example, **Disconnect** primitive), a **Digit_Collection** notification is immediately sent to the SCU. Any digits that have been buffered in the PM (the digits are dialed before the second primitive arrives, but the digits have not been reported to the CC) are lost.
- e. When “Resuming” digit collection, the Inter-Digit-Timer is used for both PSIG and PDIL timing values (the First-Digit-Timer is not used). The definition of “Resuming” digit collection is the receipt of a **CD** or **PPCD** primitive when digit collection is currently active and the Discard-Buffered-Digits is false.
- f. When a maximum of 0 digits to collect is sent in the **CD** or **PPCD** primitive; a maximum of 1 is used for the digit collection.
- g. During PSN digit collection, if a Switch Restart occurs (warm, cold or reload), the UTR Channel allocated to the PSN call remains allocated until the internal UCS DMS-250 UTR trunk audit cleans up the resource. Note that the other UTR Channels are not affected by this resource tie-up.
- h. During PSN digit collection (from a **PPCD** primitive) with an ANNC Prompt playing, if a Switch Warm Restart occurs, the ANNC continues to cycle until the Warm Restart is over.
- i. Two wire trunks (for example, DALs) may “leak” some of the received voice prompts signal into the Call Processing as user input. When the Bong Tone is recorded, starting with a tone that represents an “#”, under severe “leak” the Call Processing may interpret this as the termination of user digit stream input.
- j. When digit collection is being performed during a PSN call, the following digits are valid as the first digit dialed: 0 to 9, *, and #. Special digits A, B, C, and D, if dialed as the first digit, are ignored. However, if the special digit is dialed after the first digit, then it is collected and reported to the SCU.

For example, if the user dials A214, then 214 is reported to the SCU. However, if the user dials 214B, then 214B is reported to the SCU.

Buffering of digits on the PSN

- k. Buffering of digits on the PSN will begin as soon as either a “**New_Call**” or a “**Digits_Collected**” event notification message is sent to the PSN. It continues until a primitive other than “**Collect_Digits_&_Report**” or “**PPCD**” is received.

If during the buffering of digits, a **CD** or a **PPCD** is received with the “*DigitsCollection*” parameter’s discard buffered digits field set to 1 (true), then the digits that were buffered thus far are discarded and the digit collection is started all over again as per the received primitive. In this case, the prompt in the **PPCD** is played before digit collection.

If during the buffering of digits, a **CD** or a **PPCD** is received with the “*DigitsCollection*” parameter’s discard buffered digits field set to 0 (false), the digits that were buffered in the CC thus far are applied to the received primitive. All buffered digits in the PM that have not been reported to the CC are lost. If more digits are required, digit collection is resumed. Otherwise, the digits that were buffered in the CC are returned to the SCU in the **Digit_Collected** event notification message. In this case, the prompt received with the **PPCD** is not played before digit collection.

Note 1: The maximum number of digits that the PSN will buffer before an instruction is received from the SCU is 45.

Note 2: If the PSN has buffered digits and a **PPCD** has been received, then the buffered digits are applied to it. If more digits are required (more than the number buffered), then the digit collection is started without playing the message (tone/prompt) before collecting the first digit.

Prompt/announcement timer versus first digit timeout

In the **PPCD** primitive, the first digit timer as well as the Prompt/Announcement timer is specified.

The First Digit Timer is used to time the collection of the very first digit on the PSN.

The prompt time is the tone duration specified in the *MessageInfo* parameter. The tone will stop playing after that duration. The announcement is played for the number of cycles specified in the *MessageInfo* parameter, after which the announcement stops playing.

If the **PPCD** specifies a tone to play, then the tone and the first digit timer are started simultaneously. If the user does not dial a digit, then a timeout is reported to the SCU, after the tone is played or the first digit timeout, whichever occurs first.

If the **PPCD** specifies a tone to play that is 1 second long, the tone will play for 2 seconds instead, because the PSIG timer has a range from 2 to 30 seconds.

If the **PPCD** specifies an announcement to play, then the announcement and the first digit timer are simultaneously started. If the user does not dial a digit, then a timeout is reported to the SCU after the first digit timeout occurs, even if the announcement plays for a duration shorter than or finishes playing before the first digit timeout occurs.

When a MIN of 0 is sent in a **CD/PPCD** primitive, the 2nd Timer (the InterDigit Timer) is used for digit-collection PSIG and PDIL Timing.

3 Connect

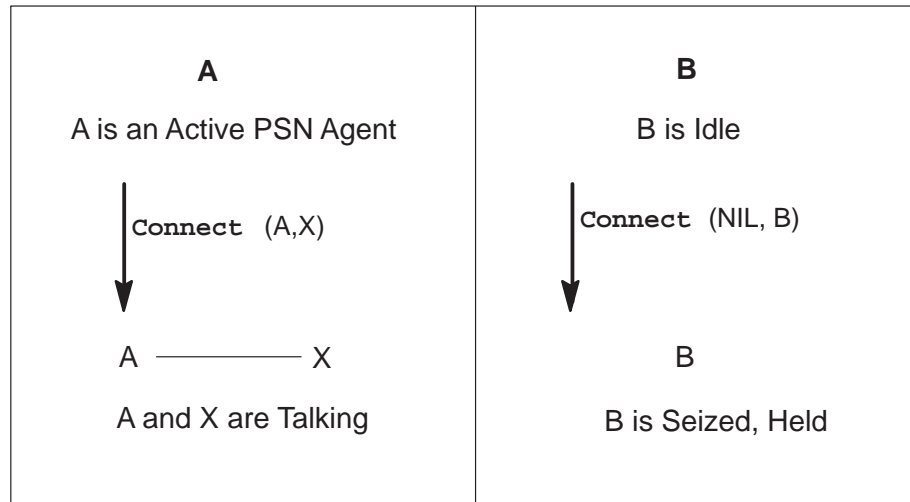
One-party vs. two-party connect

- a. There are two flavors of the **Connect** primitive: one-party and two-party.

A **one_party Connect** has NIL-TRKGRP and NIL-TRKMEM values for the *PortInfo* parameter for party A. An idle terminating agent is seized.

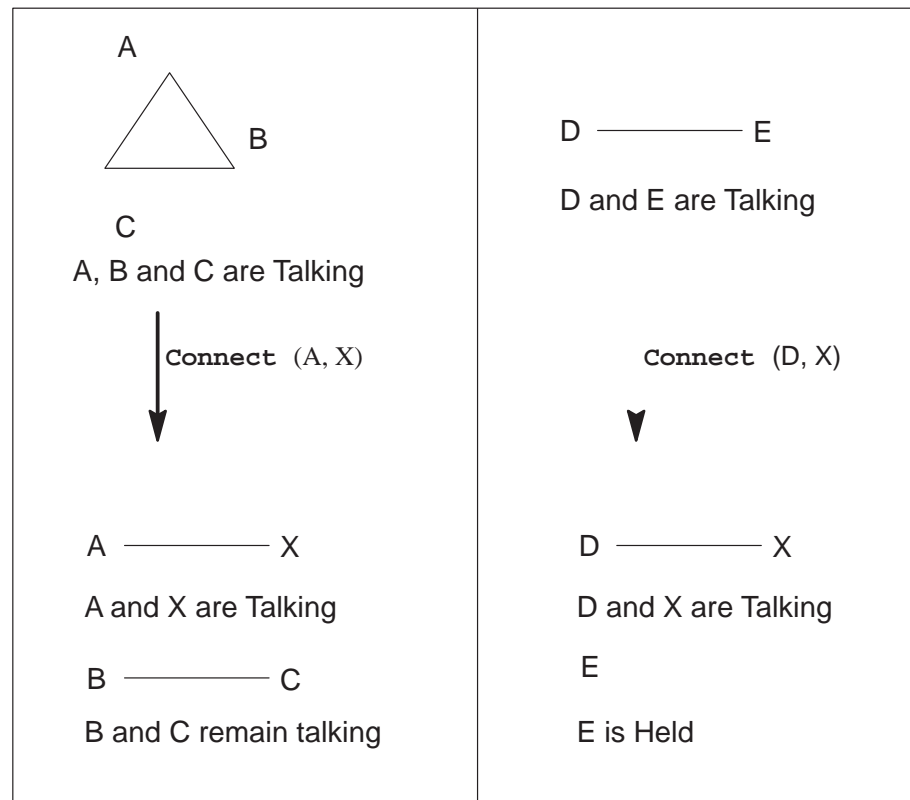
A **two_party Connect** has the TRKGRP and TRKMEM Numbers of an active PSN agent for the *PortInfo* parameter for party A. Party A is connected to an idle terminating agent.

This is shown in the following diagram.



- b. A **Connect** on a port that is presently involved in a two-party connection, connects this port to a new port and puts the connected party on hold.
- c. On the other hand, a **Connect** that is received on a port that is involved in a three or more party connection, connects this port to a new port and the remaining ports in the initial connection remain connected.

This is shown in the following diagram.



- d. The SETUP Q.931 Signaling Info parameter is mandatory in a **Connect** primitive for a PRI port. However, the IAM or the PTS Digits to Outpulse parameters in the Signaling Info are optional for the SS7/PTS ports.

Voice path connection

- e. When a **Connect** is received by the PSN, a connection is established between the two specified trunks and voice path is established in both directions. This functionality is only applicable to a **two_party Connect**
- f. If setting up the voice path in both directions is not acceptable, then the following macros may be used:
- **Connect +Hold**: to setup a connection between the two parties but with no voice path in either direction.
 - **Connect + Mute**: to setup a connection between the two parties but with the voice path setup in only one direction.

Outpulsing when a connection is established

- g. Outpulsing on an agent that is seized using the **Connect** primitive is optional. This is controlled by making the Signaling Info parameter optional in the **Connect** primitive.

If outpulsing is required, then the voice path between the two parties in both directions is setup after outpulsing is completed.

Member advancing

- h. After a **Connect** (A and B) has been successfully processed by the FSM (a Route-Selected is sent to the SCU), a `Termination_Failure` Msg may arrive for party B from the PM. This happens when the termination protocol is not followed by the other end of the terminating party B, such as the Local Exchange Carrier (LEC) did not return a Wink. The failure of this termination protocol indicates that the initial party B cannot be seized.

When this failure occurs, the next available party B member is seized and another route selected is sent to the SCU, using the IP Address stored for the previous party B. If the terminations continue to fail, the next available members are seized and Route Selected is sent to the SCU until either three subsequent member advances have occurred for the most recent **Connect** primitive, or all of the available party B members have been attempted, in which case, a `Fatal_Error_Detected` is sent to the SCU.

If **Signalling_Info** was present in the initial **Connect** primitive, then the same signaling is performed on every successive party B.

If an explicit or implicit **Hold** is performed on the connection between party's A and B, then the member-advancing ability is immediately terminated. In other words, if the PSN has not received the completion of the terminating protocol for party B (in this case a `Termination_Failure` Msg is imminent) and a Hold is performed on party A and B, then party B is Held and the `Termination_Failure` Msg does not arrive in the PSN.

Hence, member-advancing is aborted—no `Termination_Failure` Msg even though the Wink protocol did not complete—by the primitives listed below (for either party A or B).

- **Bridge**
- **Collect_Digits_&_Report**
- **Connect**
- **Disconnect**

- **Hold**
- **Play_Message**
- **Play_Prompt_Collect_Digits_&_Report**
- **Reconnect**
- **Transmit_Siginfo** (PTS On-hook, SS7 REL, PRI REL/DISC)

The following primitives (destined for party A or B) do not affect Member-Advancing:

- **Monitor**
- **Mute**
- **Query_Port**
- **Set_IP_Address**
- **Set_Billing_Record**
- **Transmit_Siginfo** (other than PTS On-hook, SS7 REL, and PRI REL/DISC)

Parameters

- i. When a **Connect** primitive contains the Billing Info or IP Address for party B, the trunk member of party B has to be NIL (32767).

Unsupported vs. unavailable agents

- j. When terminating to a PSN agency (via the **Connect** primitive), an idle member of the trunk from table TRKGRP is obtained before performing any validation on the agency. This is because validation on a trunk member is not possible without obtaining one.

If an idle member is not available, then an **Error_Detected** event notification is sent to the SCU with an error cause of “Route Not Available”.

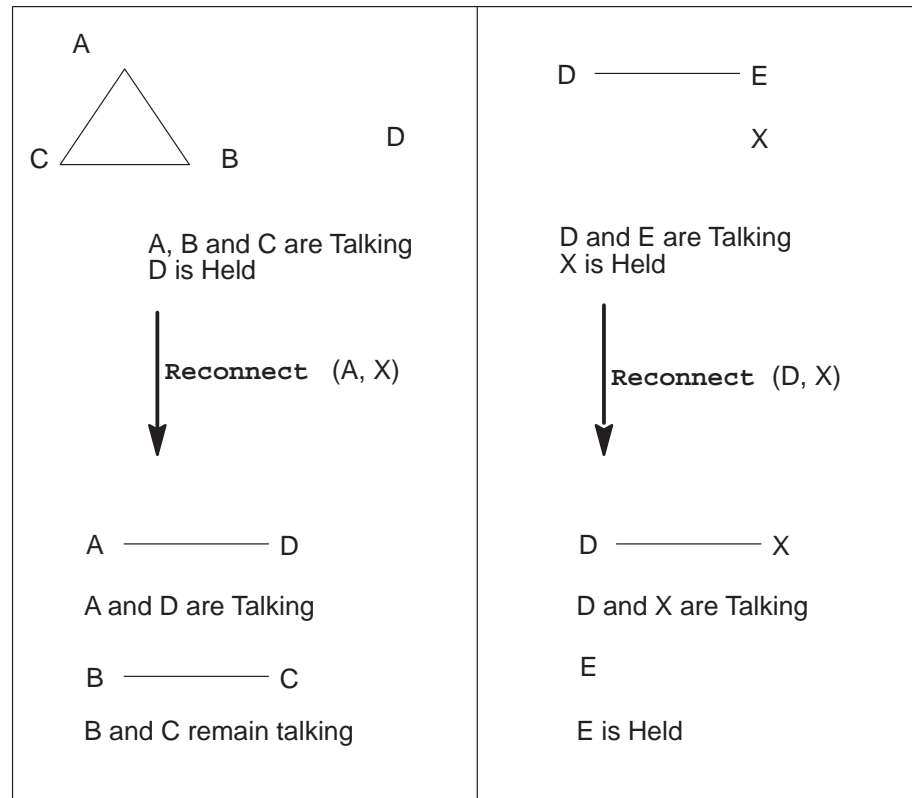
If an idle member is available, and the agency is not supported by PSN for termination, then an **Error_Detected** with an error cause of “Agent Unavailable” is sent to the SCU.

4 Reconnect

- a. A **Reconnect** on a port that is presently involved in a two-party connection, reconnects this port to a new port and puts the connected party of the previous connection on hold.

A **Reconnect** that is received on a port that is involved in a three or more party connection, reconnects this port to a new port and the remaining ports in the initial connection remain connected.

This is shown in the following diagram.

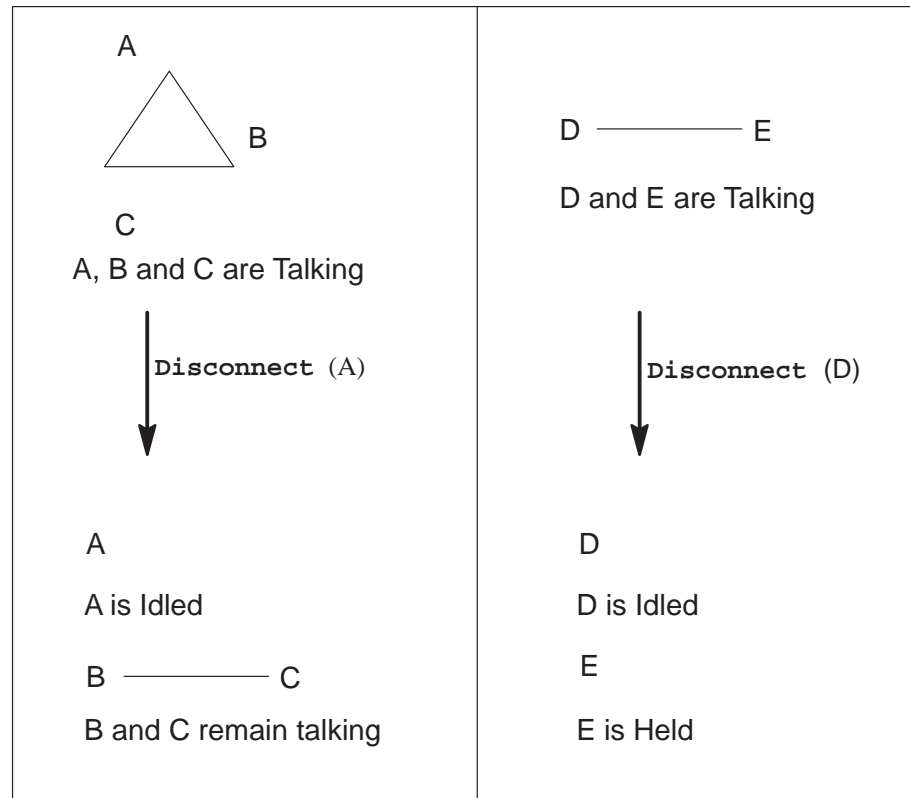


5 Disconnect

- a. If a **Disconnect** is received on a port that is in a connection with another port (and the connection does not have more than two parties), the port on which the primitive is received is idled, and the connected port state changes to Answered (Seized or Held).

If the above connection does have more than two parties, then the port on which the **Disconnect** primitive is received is idled, while the remaining parties remain in the connection as before.

This is shown in the following diagram.



- b. The receipt of a **Disconnect** primitive results in an **Instruction_Completed** event notification. When the Agent goes **On_Hook** after being disconnected, an **On_Hook** event is not sent to the SCU as the Agent is no longer considered to be controlled by the SCU.

6 Monitor

- a. Receipt of any valid primitive after a Monitor does not abort a tone or digit monitoring, unless the received primitive is **CD/PPCD**.
- b. In order to turn off monitoring for tones, the SCU may send another **Monitor** with all the bits in the **MonitorMask** parameter turned off.
- c. If a port is being monitored for tone/digit and the port reports the requested tone/digit, then the monitoring for tones is aborted and the detected digit/tone is returned to the SCU in the **Tone_Detected** event notification message.

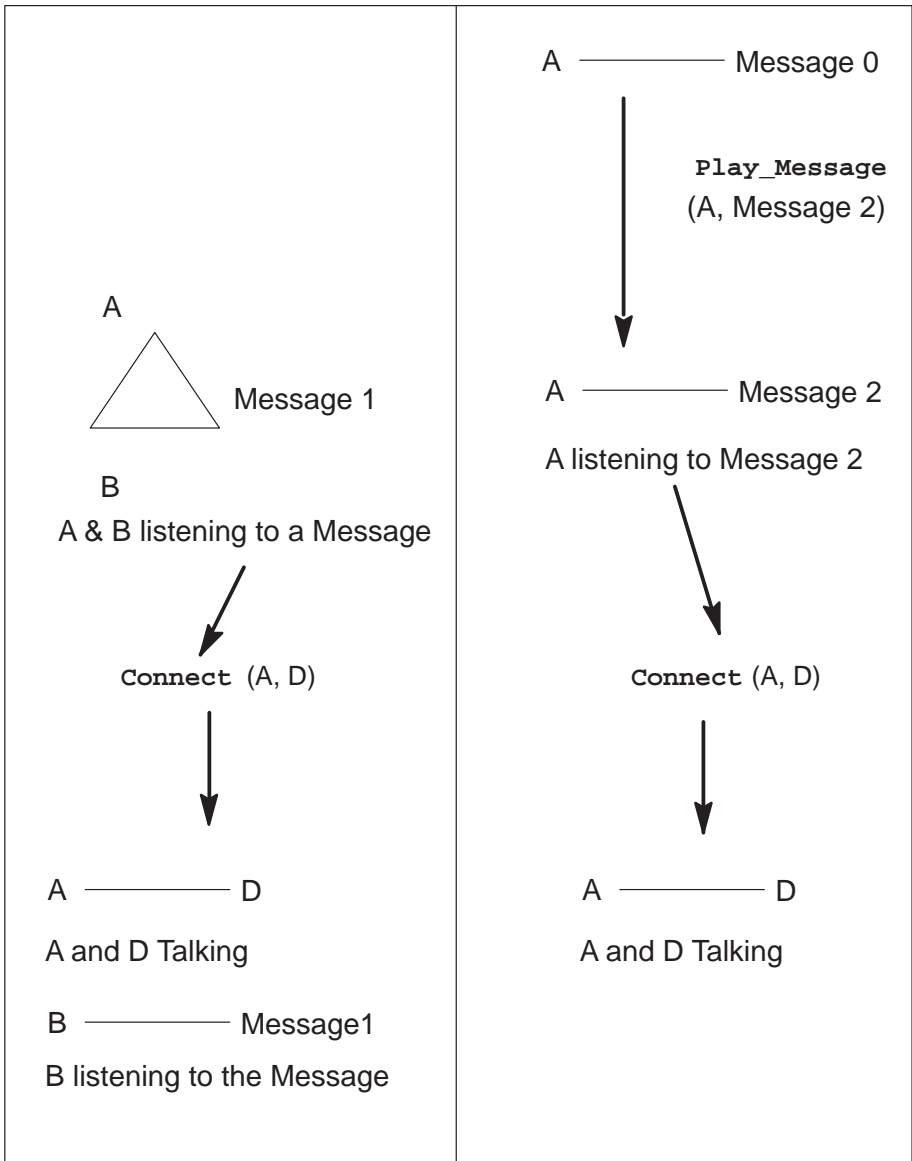
BBF (blue box fraud) support for PSN

- d. BBF monitoring is supported for an “originating” port (where “originating” indicates party A of a **Connect** or a **Reconnect** primitive), provided the “terminating” port (where “terminating” indicates party B of a **Connect** or a **Reconnect** primitive) is datafilled in Table BBTKSGRP with the necessary datafill. BBF is not supported under any other combination of agent topology or BBTKSGRP datafill.

7 **Play Message**

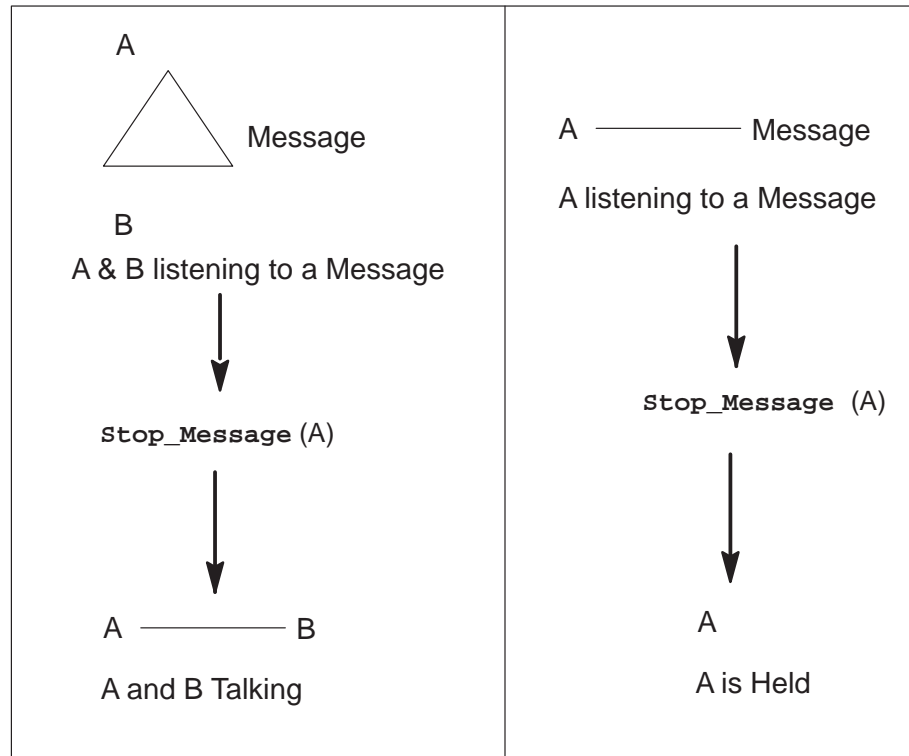
- a. When a second **Play_Message** is received on a port that is presently listening to a message, the first message is aborted and the port is connected to the second one.
- b. The tone durations of 1 and 2 seconds are invalid for the **Play_Message** primitive.
- c. If a **Connect** or a **Reconnect** is received on a port that is listening to a message (and not connected to any other party), the message is stopped and the new connection is made.

This is shown in the following diagram.



- 8 **Stop_Message**
 - a. When a **Stop_Message** primitive is received on a port that is listening to a Message, or upon completion of a Message that is being played, the port state automatically goes into an Answered Held state from the Answered Linked state (if the port is not involved in a connection with another port).

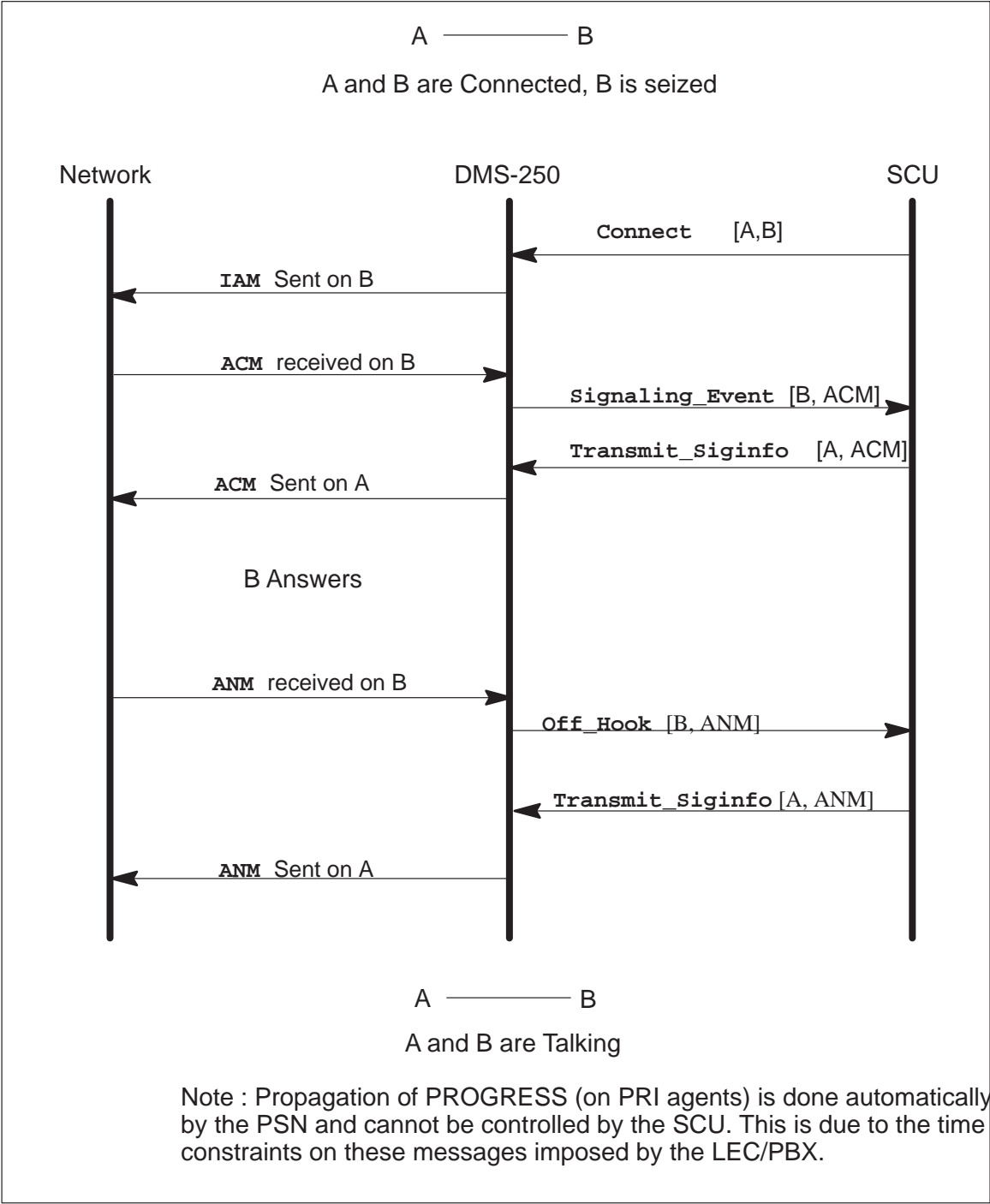
This is shown in the following diagram.



9 **Transmit_Siginfo**

- a. This primitive can be received on a port that is in any state except the Idle Held state.
- b. This primitive, together with the **Signaling_Event** notification message, can be used for answer and release propagations, for example.

The SS7 Message Propagation example is shown in the following diagram.



- c. The SCU must not send the IAM more than once on an SS7 port. For example, if the port receives an IAM in a **Connect** primitive, then it cannot be sent again in the **Transmit_Siginfo** primitive.
- d. The SETUP Q.931 message is mandatory in a **Connect** primitive for a PRI port. Therefore, the **Transmit_Siginfo** will not be used to send the SETUP to a PRI port because the SETUP message may not be sent multiple times to a PRI port.
- e. The answer messages (that is, the ANM, CONNECT, and PTS Off-hook) are invalid when received on a port that is in a Seized state.
- f. The answer message for a PRI port (the Q.931 CONNECT message) may only be sent on a port that is in the Answered Linked state.

Feature interactions

Currently, there are no feature interactions with existing in-switch features. Once a call is tagged a service call, the SCU is given full control of this call. The SCU may then send instructions to control the call flow of this call.

The SCU continues to have full control of this call until all parties in this call go On-hook or are disconnected by the SCU.

Terminating agent bearer capability (BC) screening

When a connection is made between any two ports, using either a **Connect** or a **Reconnect** primitive, the Bearer Capability (BC) of the two ports is screened for compatibility. Bearer Capability compatibilities are defined in the table, BCCOMPAT. This table defines the Bearer Capability pairs that are compatible with one another.

If the Bearer Capability screening fails, then the connection is not made and an Error Detected event notification message is sent to the SCU.

The BC compatibilities are derived from the following sources:

- 1 For the port that triggers a **New_Call** event to the SCU, the BC is obtained in-switch.

This value is obtained from either the table, TRKGRP or the table, MEMATTR (if the trunk has the MEMATTR option in table TRKGRP).

For subsequent connections on this port, the SCU can change the port's BC, using the optional Originator Bearer Capability parameter in the **Connect** primitives). When this happens, the new BC is used to check the compatibility.

- 2 For all subsequent ports that are added to a service call, the Bearer Capability is derived from the *signalingInfo* parameter that is included in the **Connect** primitive. Subsequent connection setups on this port use this Bearer Capability information for BC screening.

The Bearer Capability information is included in the *signalingInfo* parameter as the following sub-parameters and is based on the signaling type of the port. If the Bearer Capability sub-parameter is not included in the *signalingInfo* parameter, then the BC is derived from the table TRKGRP.

- the sub-parameter BEARER CAPABILITY for PTS agents
- the sub-parameter USER SERVICES INFORMATION for SS7 agents
- the sub-parameter BEARER CAPABILITY (in Q.931 format) for PRI agents

New call rejected

When the PSN determines that a call is to be controlled by the SCU, it sends a **New_Call** event notification message to the SCU. The SCU may then decide not to provide service to this call. In this case, a **New_Call_Rejected** instruction is sent by the SCU to the PSN.

When the **New_Call_Rejected** event notification is received by the PSN, the associated port will be routed to the PSNF treatment.

New call timeout

When the PSN determines that a call is to be controlled by the SCU, it sends a **New_Call** notification message to the SCU. If the data link or the SCU is down for some reason, the message may never reach the SCU.

In order to avoid the call from being hung forever, a timer (controlled by the **PSN_EVENT_TIMER** Office parameter) is started every time a **New_Call** event is sent to the SCU. If the SCU does not respond within the said time, then the timer pops and the port is routed to the PSNF treatment. Also, the PSN201 log is generated.

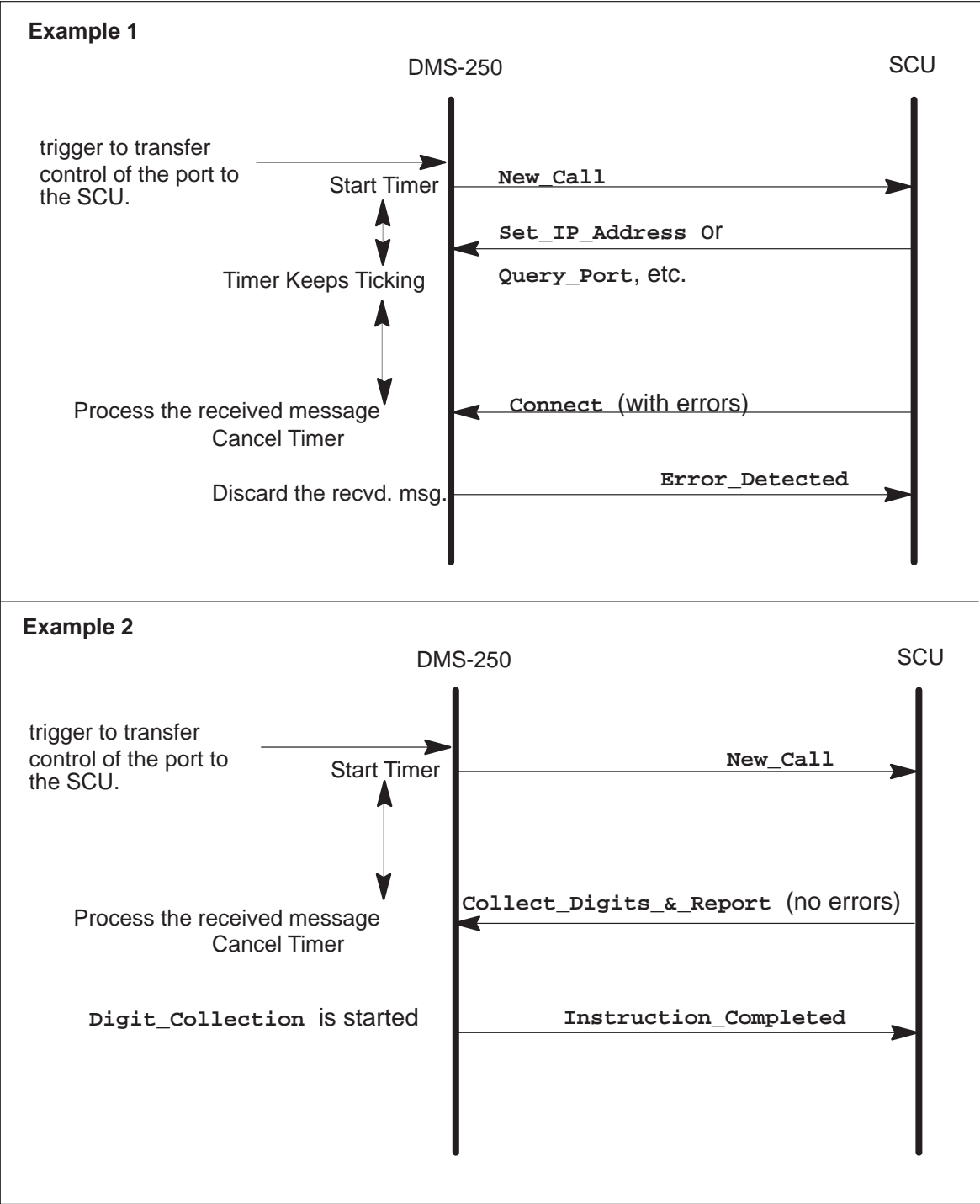
If the PSN receives a message before this timer pops, the following events will transpire:

- If the SCU responds with a **New_Call_Accepted** or any other call control primitive message, the timer is canceled.
- If the received message has no errors, then the instruction is executed.
- If the received message has errors, then the instruction is discarded and the PSN waits for the next instruction.

The following is a list of primitives that upon receipt, cancel the
PSN_EVENT_TIMER:

- 1 **Bridge**
- 2 **Connect**
- 3 **Collect_Digits_&_Report**
- 4 **Disconnect**
- 5 **Monitor**
- 6 **Mute**
- 7 **New_Call_Accepted**
- 8 **New_Call_Rejected**
- 9 **Play_Message**
- 10 **Play_Prompt_Collect_Digits_&_Report(PPCD)**
- 11 **Reconnect**
- 12 **Set_Billing_Record**
- 13 **Stop_Message**
- 14 **Transmit_Siginfo**

The following diagram shows the use of PSN Event timer during an PSN call.



Error handling

Errors that are detected on the PSN are reported back to the SCU. The SCU determines the error handling results for the call and sends the necessary primitive instructions to the PSN for execution.

The Errors detected at the PSN may be classified into three types:

- non-Context Specific Protocol Errors in the received SCU messages
When these errors are detected, the PSN202 log is generated and an OM register in the PSN_ERPS OM group is pegged.
- context Specific Application Errors in the received SCU messages
When these errors are detected, the PSN202, PSN203, PSN207 or PSN208 log is generated and an OM register in the PSN_ERFM or the PSN_ERPS OM group is pegged.
- PSN Internal Errors
When these errors are detected, the PSN204, PSN205, PSN206 or PSN208 log is generated and an OM register in the PSN_ERFM OM group is pegged.

For details on logs and OMs, please refer to chapters “PSN logs” and “PSN operational measurements”. Also, refer to *UCS DMS-250 Logs Reference Manual* and *UCS DMS-250 Operational Measurements Reference Manual*.

Fatal versus non-fatal errors

Each error that is detected on the PSN or reported by the SCU, may be Fatal or non-Fatal.

Non-fatal errors prevent the execution of the current primitive, but do not prevent the processing of future primitives on this port. Hence, the port is not routed to a treatment or taken down.

Action:

- Upon detection of non-fatal errors, an `Error_Detected` message is reported to the SCU. The port is not taken down.
- The SCU may use the `Error_Detected` message to report a non-fatal error. Such errors are logged (PSN200 logs) and the appropriate OMs are pegged, but the port is not taken down.

Fatal errors not only prevent the execution of the current primitive, but also prevent the processing of future primitives on this port. Hence, the port is taken down.

Action:

- Upon detection of fatal errors at the PSN, an `Error_Detected` message is reported to the SCU. Also, the associated port is taken down or idled.
- If the SCU detects a fatal error, then the error is reported as a **Disconnect** primitive. This results in the appropriate port being idled at the PSN.

Non-context specific protocol errors

The Non-Context protocol errors are defined as errors that occur due to protocol violations of primitive command formats.

A list of protocol errors are specified below. For each error, the action taken by the PSN is also given.

- Unrecognized Primitive

The primitive that is received in the message is not one of the pre-defined ones.

Action: an **Error_Detected** event notification is sent back to the SCU, and the primitive is discarded. The associated port is not affected in any way.

- Missing Mandatory parameters

One or more parameters that are identified as mandatory for the given primitive are missing.

Action: an **Error_Detected** event notification is sent back to the SCU, and the primitive is discarded. The associated port is not affected in any way.

- Unrecognized parameters

The parameter that is received in the primitive is not one of the pre-defined ones.

Action: the parameter is dropped and the processing of the primitive continues. The associated port is not affected in any way.

- Parameter Contents invalid (out of range, unused, spare, or reserved)

The parameter contents are invalid. This means that any field in the parameter contents has a value that is invalid.

For example, the inter digit timer field in the digits dialed parameter is five bits long. Its value may be from 0 to 32. But the valid range has been defined as 1 to 20. A value of 0 is invalid, as it is a spare.

The trunk group number is defined as a number between 0 – 9999. The value 10000 is invalid because it is out of range.

Actions:

- If the parameter is mandatory for the given primitive, then an **Error_Detected** event notification is sent back to the SCU, and the primitive is discarded.
- If the parameter is optional for the given primitive, then the parameter is dropped and the primitive is processed in its absence.
Either way, the associated port is not affected.

- Duplicate parameters received in a primitive
More than one parameter of the same type is received with a given primitive. For example, two Message Info parameters are received in a **Play_Message** primitive.

Action: The last parameter from the duplicate parameters list is the one that is used for primitive execution. The other parameters before this one are discarded and the associated port is not affected.

Context specific application errors

The context specific application errors are defined as errors that are identified during the execution of the primitive instructions.

These errors depend upon the context of the call. For example, a primitive can be valid when received in a particular port state, but invalid in another port state.

Primitive validity based on agent state

Primitives are received for a port and agent. The primitive is valid or invalid based on the Agent state. For example, a **Collect_Digits_&_Report** received on a port that is idle or seized is invalid. All such errors are treated as non-fatal errors.

Table 4-1 captures this information in detail.

Table 4-1, primitive sequence based on agent state

(I = invalid; V=valid)		Idle	Seized	Answered
	P R I M I T I V E R E C E I V E D	AGENT STATE		
Bridge		I	V	V
Collect_Digits_&_R eport		I	I	V
Connect		I	V	V
Disconnect		I	V	V
Hold		I	V	V
Monitor		I	I	V
Mute		I	I	V
New_Call_Rejected or New_Call_Accepted		I	I	V
Play_Message		I	I	V
PPCD		I	I	V
Query_Port		V	V	V
Reconnect		I	V	V
Stop_Message		I	I	V
Transmit_Siginfo		I	V	V
Set_Billing_Record		I	V	V
Set_IP_Address		I	V	V

Primitive sequence based on connection state

The order in which primitives are received from the SCU is very important. This is because most primitives change the (connection) state of the port, thus making the new state valid or invalid for the subsequent primitive that is received from the SCU.

For example, a **Reconnect** has been received on an Answered port, which is valid. If a **New_Call_Rejected** primitive is received next on the same port, it is invalid. However, a **Bridge** a **Hold**, or a **Collect_Digits_&_Report** on this port is valid. This decision is based on the port's Connection state, which after receiving the **Reconnect**, has been changed to Linked.

Table 4-2 tries to capture this information. The column of primitives contains the primitive that is received FIRST. The row of primitives identifies the primitive that is received NEXT. For example, the first row of this table shows the primitives that can be received after a Bridge.

Upon receipt of the next primitive, the PSN does one of two things:

- The PSN executes the primitive because the primitive is valid in the new port state (this condition is marked by a “V” in the table).
- The PSN sends an **Error_Detected** event notification message to the SCU because the primitive is considered invalid in the new port state (this condition is marked by an “I” in the table). The error that is detected is non-fatal and does not affect the port.

Table 4-2, primitive sequence based on the connection state

	B r i d g e	C o l l e c t _ D i g i t s _ & _ R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	New_Call_Rejected or New_Call_Accepted	P l a y _ M e s s a g e	P P C D	Q u e r y _ P o r t	R e c o n n e c t	S t o p _ M e s s a g e	T r a n s m i t _ S i g n i f o
	RECEIVED SECOND													
Bridge	V	V	V	V	V	V	V	I	V	V	V	V	V	V
(after Message_Played)	V	V	V	V	V	V	V	I	V	V	V	V	I	V
Collect_Digits	V	V	V	V	V	V	I	I	V	V	V	V	I	V
(after Digits_Collected)	V	V	V	V	I	V	I	I	V	V	V	V	I	V
Connect	V	V	V	V	V	V	V	I	V	V	V	V	I	V
Disconnect	Not Applicable as the port is idled after a Disconnect													
Hold	V	V	V	V	I	V	I	I	V	V	V	V	I	V
Monitor	All scenarios except New_Call_Accepted and New_Call_Rejected are VALID as Monitor does not change the state of the port													
(after Tone_Detected)	All scenarios except New_Call_Accepted and New_Call_Rejected are VALID as Monitor does not change the state of the port													
Mute	V	V	V	V	V	V	V	I	V	V	V	V	V	V

4-44 PSN finite state machine

	B r i d g e	C o l l e c t _ D i g i t s _ & _ R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	New_Call_Rejected or New_Call_Accepted	P l a y _ M e s s a g e	P P C D	Q u e r y _ P o r t	R e c o n n e c t	S t o p _ M e s s a g e	T r a n s m i t _ S i g i n f o
New_Call_Accepted	V	V	V	V	I	V	I	I	V	V	V	V	I	V
New_Call_Rejected	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Play_Message	V	V	V	V	V	V	V	I	V	V	V	V	V	V
(after Tone_Detected)	V	V	V	V	V	V	V	I	V	V	V	V	I	V
PPCD	V	V	V	V	V	V	I	I	V	V	V	V	I	V
(after Digits_Collected)	V	V	V	V	I	V	I	I	V	V	V	V	I	V
Query_Port	All scenarios except New_Call_Accepted and New_Call_Rejected are VALID as Query_Port does not change the state of the port													
Reconnect	V	V	V	V	V	V	V	I	V	V	V	V	I	V
Stop_Message	V	V	V	V	V	V	I	I	V	V	V	V	I	V
Transmit_Siginfo	All scenarios except New_Call_Accepted and New_Call_Rejected are VALID as Transmit_siginfo does not change the state of the port													

Primary party primitives that affect the connected party

A Connection can consist of two or more parties. One of the parties is the primary party whose port receives the primitive instruction from the SCU. The others are Connected Parties.

Upon receipt of primitives that affect the connection, the Port state of the Connected party can also change. Hence, primitives received on the Connected party can be valid or invalid depending upon the primitive that was received earlier on the primary party.

For example, a Disconnect that is received on the Primary port changes the Connection state of the Connected party to Held. Therefore, a Mute received on the Connected party immediately after the Disconnect on the Primary party, is invalid.

Table 4-3 captures this information.

Table 4-3, primary part primitives affect the connected party

		B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	New_Call_Rejected or New_Call_Accepted	P l a y - M e s s a g e	P P C D	Q u e r y - P o r t	R e c o n n e c t	S t o p - M e s s a g e	T r a n s m i t - S i g n i f o
		RECEIVED on CONNECTED PORT													
Bridge	P R I M - P O R T	V	V	V	V	V	V	V	I	V	V	V	V	V	V
Connect		V	V	V	V	V	V	V	I	V	V	V	V	I	V
Disconnect		V	V	V	V	I	V	I	I	V	V	V	V	I	V
Reconnect		V	V	V	V	V	V	V	I	V	V	V	V	I	V

Restrictions and limitations

The following restrictions and limitations apply to PSN calls:

- COT (COntinuity) checks are not supported on SS7 terminations in PSN phase 1.
- Tone Monitoring is not supported on PRIs because DTCl hardware does not support STRs.
- BBF monitoring is supported for an “originating” port (where “originating” indicates party A of a **Connect** or a **Reconnect** primitive), provided the “terminating” port (where “terminating” indicates party B of a **Connect** or a **Reconnect** primitive) is datafilled in Table BBTKSGRP with the necessary datafill.

- **Collect_Digits_&_Report** and **Play_Prompt_Collect_Digits_&_Report** primitives do not support the collection of MF or DP digits.
- The **SUSPEND** SS7 message is treated as a RELease. Upon receipt of this message, the port is taken down (similar to a release) for phase 1.

Warm restarts

- When a Warm Restart occurs, there is a short period of time when messages from the PSN agent's peripheral are not processed. For example, if a PSN agent goes On-hook during a Warm Restart and the PM Message is not processed, then the FSM (the primitive Server, Audit, and SCU) does not know the PSN agent is disconnected, and the PSN still considers the call active.
- In certain timing scenarios coupled with the signaling type of the PSN agent, when a PSN agent has changed states during a Warm Restart and that PSN agent is later supervised (due to a SCU primitive that does an implicit or explicit Hold, for example), the FSM may be able to determine the new state of the PSN agent and notify the SCU. However, when the FSM does not "become aware" of the state change, as with UTR Digit-Collection and Warm-Restart interactions, PSN depends on the existing TRK Audits to clean up the PSN agent.

No-Restarts SWACTS

- When a no-restart SWACT occurs, only simple two-party calls are left up. PSN Calls are not simple two-party calls; therefore, they are taken down during a no-restart SWACT.

Hardware

- Digit Collection is done using UTRs and Tone/Digit Monitoring is done using STRs. Hence, DTCs must be equipped with UTRs and STRs.
- The existing six-port conference bridge is used to allow more than two ports to be bridged. Also, it is used to bridge two or more parties to a message. Therefore, regardless of the physical conference circuit card (that is, the 6-port card or the 30-port card), the datafill associated with the card (in table CONF6PR) must specify logical 6-port conference circuits. In other words, a physical conference circuit card may also be datafilled as various 3-port logical circuits (for 3-way calling), but these conference circuits are not used by PSN— only 6-port logical conference circuits are used by PSN.

- Since PSN uses 6 SOS Processes (that is, Primitive Server) along with CallIP (where the FSM executes), a powerful CPU is required to realize the full potential of PSN. The minimum recommended DMS CPU is the BRISC Series 70 Extended Memory (SN70EM) CPU. All 1, 2, and 3 Party Primitives work easily on the slower CPUs (for example, ECORE or BRISC Series 60), however >3 Party Primitives (for example, a Bridge of 10 Agents) are not guaranteed to succeed on the slower CPUs.

PSN flow control

This chapter contains information on the PSN flow Control functionality.

The primitive associated with the flow control is described in detail in the “PSN messages” chapter. The OMs pegged for flow control are given in the Chapter “PSN OMs”. The logs generated for flow control are covered in the Chapter “PSN logs”.

Overview

Call traffic originating at the PSN triggers data traffic at the PSN-SCU interface. This data traffic, under overload condition, may potentially overflow the call processing layer, receive queues at the PSN and the corresponding message facility at the SCU, and result in message loss. The message loss may in turn impact the services on existing calls. In order to prevent this impact, PSN flow control provides a mechanism to control the flow of data by controlling the new call traffic generated at the PSN.

Call processing layer flow control at the PSN-SCU interface is instrumented by enabling the PSN, or the SCU, to control the rate of **New_Call** event notifications sent to the SCU. The control is applied to all **New_Call** events and no call discrimination is done at the PSN.

Flow control application process

Flow Control application process receives and processes Flow Control primitives. The userclass for Flow Control application is 04, which receives and sends the **Flow_Control** primitive and the **Error_Detected** primitive and event. If any primitive is received at the Flow Control application that is not a Flow Control or an **Error_Detected** primitive, an **Error_Detected** event is sent to the SCU with an error cause “Primitive UserClass Mismatch Ec” and a PSN207 log is generated with a text reason “Prim Not Supported For Userclass” at the PSN.

The maximum number of primitives that can be sent in a message to the Flow Control application is one. If a message is received at the Flow Control application with more than one primitive, then an **Error_Detected** event is sent to the SCU with an error cause “Max Primitive Exceeded Ec” and a PSN209 is generated at the PSN.

Flow control initiation

Flow Control at the PSN-SCU interface may be initiated by either end of the interface.

SCU as a source of flow control

The SCU may initiate Flow Control at the PSN-SCU interface by sending Duration and Gap information in the **Flow_Control** primitive. At the receipt of a **Flow_Control** primitive, the Duration and Gap indices are validated. If the Gap index is greater than the Duration index, an **Error_Detected** event notification is sent to the originator of the **Flow_Control** primitive at the SCU with an error cause “Invalid_Dur_Gap_EC” and the Duration and Gap values are not applied at the PSN. If the Duration and Gap indices are valid, then a PSN106 log with text reason “Flow Control Activated” is generated and the control is instrumented at the PSN by limiting the rate of **New_Call** event notifications by the control Gap for an amount of time specified by the control Duration.

If a subsequent **Flow_Control** primitive is received at the PSN while the control is active, the Duration and Gap values sent later override the former values.

The SCU may remove Flow Control while it is active by sending 0000 as the control Gap value in another **Flow_Control** primitive. If a **Flow_Control** primitive is received at the PSN with a control Gap value equal to 0000 while the control is active based on the Duration and Gap values previously received from the SCU, the control is removed and a PSN106 log with text reason “Flow Control Deactivated” is generated at the PSN.

If a **Flow_Control** primitive is received at the PSN while the office parameter PSN_Flow_Ctrl_Messaging is set to N, then the **Flow_Control** primitive is ignored, and the control is not instrumented at the PSN.

PSN as a source of flow control

At the PSN, a nonpreemptive priority queuing is applied to the primitives and events related to existing calls and **New_Call** events. All events that belong to existing calls are assigned the highest priority class. All primitives that belong to existing calls are assigned a priority below the events belonging to existing calls, and all **New_Call** events are assigned the lowest priority class. This ensures timely processing of events and primitives belonging to existing calls, and throttles the **New_Call** event processing at a rate proportional to the current traffic rate and the current CPU occupancy at the PSN.

The **New_Call** event queue delay increases with the traffic load at the PSN, and under heavy traffic conditions it results in New Call time-outs for the new calls that are queued for a time greater than the office parm,

PSN_EVENT_TIMER. All PSN calls that encounter a **New_Call** event timeout are dropped at the PSN, a PSN201 log is generated, and the OM register SCUTMOUT in PSN_ERFM is pegged at the PSN.

Interactions between PSN and SCU initiated flow control

The Flow Control initiated by the SCU is applied independently of the control, resulting from large queue delays at the PSN.

Flow control behavior at restarts

The Duration and Gap indices are initialized to nil values and the control is deactivated at cold and reload restarts at the PSN. At warm restarts, the Duration and Gap indices are not initialized and the existing control is allowed to proceed.

Flow control behavior at a heartbeat failure

If an SCU Heartbeat Failure is detected at the PSN, then the Flow Control is deactivated and a PSN106 log is generated with a text reason “Flow Control Deactivated” at the PSN.

Note: For details on the **Flow_Control** primitive and the associated parameters, please refer to the Chapter “PSN messages”. For details on the above office parameters, refer to the Chapter “PSN office parameters.”

PSN admin process

This chapter contains the detailed description of the PSN Admin process.

The Admin process is one of the user applications at the PSN. The functions of the administration process can be categorized as follows:

- Initialization

When the PSN comes into service after the reboot, the cold and reload restarts, or the heartbeat failure, it initiates the initial contact polling using initial point of contact from the table SCUADDR. The initiation of this polling is controlled by office parameter, **PSN_INIT_SCU_POLLING**. Administration process continues the polling by sending an event until it receives the SCU Arbitrator address in **Set_IP_Address** primitive. The administration process also initiates the initial contact polling after the Norestart SWACT.

- Heartbeat

Since the Custom Layer is a connectionless protocol and it sits on top of a connectionless transport layer protocol, the SCU heartbeat is required to keep the PSN informed of the status of the SCU. Once an initial contact has been established between the SCU Arbitrator and the PSN, the PSN must receive heartbeat messages (Heartbeat primitive) from the SCU Arbitrator within the window of 0 to n seconds. The value of n is defined on the PSN by office parameter **PSN_HEARTBEAT_WAIT_TIME**. The periodic heartbeat depicts the availability of the SCU Arbitrator application. If, for any reason, the administration process does not receive the heartbeat message within **PSN_HEARTBEAT_WAIT_TIME** time interval, it generates a PSN100 log with the text reason “SCU Heartbeat Failure,” pegs the OM register of the PSN_ERDC OM group, and initiates the initial contact polling.

Non-CallP Messages: The administration process also handles the following non-call processing messages related primitives and events:

- **Query_Time_of_Day** primitive
- **Error_Detected** primitive

- **Current_Time_of_Day** event
- **Error_Detected** event
- **Agent_Data** event
- **Stop_Heartbeat** event

Note: For details on the **Flow_Control** primitive and the associated parameters, refer to the Chapter “PSN messages”. For details on the office parameters given above, refer to the Chapter “PSN office parameters”.

PSN audit process

This chapter contains the detailed description of the PSN Audit process and the office parameters, logs, and OMs associated with that process.

Audit process

The Audit process is one of the user applications. The Audit is done to verify the integrity of the agents across the PSN and SCU.

Audit office parameters

The following office parameters are created for use by the Audit process.

PSN_AUDIT_INTERVAL_TIME: This office parameter specifies the elapsed time between subsequent audit cycles. The range of **PSN_AUDIT_INTERVAL_TIME** is 0 to 30 minutes in 5-minute intervals. A value of 0 for this parameter turns the audit off. Change in the value of this office parameter takes effect one iteration after the change.

PSN_AUDIT_MAX_RETRY: This office parameter is compared with an internal counter kept by the Audit process. It represents the number of cycles performed by the Audit before a Query Port is sent to the SCU. The range for the parameter is 0 To 5. The default is 0.

PSN_AUDIT_DROP_AGENTS: This office parameter is a Yes or No parameter. A Yes allows the Audit process to disconnect a port upon an audit failure. A No prevents the Audit process from disconnecting a port upon an audit failure.

Audit logs and OMs

Log PSN400 is created for audit and registers in the OM group PSN_ERPS, which are pegged. The following are the registers pegged and the reasons they are used in the PSN400 log:

SCU_PORT_STATUS_MISMATCH is used, and AUDPSM is pegged, when SCU sends a Port Status to the Audit process that indicates an Agent state other than Seized or Answered.

DISCN_TRK_AUD_FAILS_TO_RCV_PORT_STAT is used, and AUDPSF is pegged, when the SCU does not send a Port Status in response to a Query Port, and the Audit is allowed to disconnect the port.

INFO_ONLY_AUD_FAILS is used when SCU does not send a **Port Status** primitive in response to a Query Port, or sends a **Port Status** to the Audit which indicates an Agent state other than Seized or Answered. The Audit is not allowed to disconnect the port. This reason is also used when the Audit sends an error detected to the SCU. The registers pegged could either be AUDPSM, for wrong port status primitive, or AUDPSF, when **No_Port_Status** primitive is sent.

Log PSN207 with reason PRIM_NOT_SUPPORTED_BY_USERCLASS is used, and OM PRMUSRMS is pegged, when SCU sends a primitive to the Audit, which is not a **Port Status** primitive.

Audit description

The Audit is initiated based on the time interval specified in office parameter **PSN_AUDIT_INTERVAL_TIME**. Upon initiation, the Audit accesses all active PSN agents. For each active PSN agent, an internal counter is maintained which is used by the Audit process. This counter is initialized to zero, and is incremented when Audit is initiated. When the counter associated with a particular agent becomes equal to **PSN_AUDIT_MAX_RETRY**, a **Query_Port** event is encoded and sent to the SCU(Client) application. After sending a **Query_Port** event to the SCU for an agent, one of two actions occurs. Either a **Port Status** primitive is received for the agent, or the **PSN_AUDIT_INTERVAL_TIME** expires, and the Audit is initiated again. If the SCU responds with a **Port Status** primitive, the Audit decodes the Port Status and acts on the information contained in the Port Status. If the Audit is initiated before receiving a Port Status, the Audit sends a disconnect (If **PSN_AUDIT_DROP_AGENTS** = Y) to the primitive server for the associated active PSN agent.

If the Port Status received indicates an agent state other than Seized or Answered, the Audit disconnects the agent if office parameter **PSN_AUDIT_DROP_AGENTS** is YES, and generates a PSN400 log with reason "SCU_PORT_STATUS_MISMATCH". If the office parameter **PSN_AUDIT_DROP_AGENTS** is datafilled as NO, the Audit only generates a PSN400 log with reason "INFO_ONLY_AUD_FAILS" and does not disconnect the agent.

If the SCU fails to respond before the Audit is re-initiated, the Audit disconnects the port, (if the office parameter **PSN_AUDIT_DROP_AGENTS** is YES) and generates a PSN400 log with "DISCN_TRK_AUD_FAILS_TO_RCV_PORT_STAT". If the office

parameter `PSN_AUDIT_DROP_AGENTS` is NO, the Audit only generates a PSN400 log with reason “INFO_ONLY_AUD_FAILS”.

If SCU returns a primitive other than **Port_Status** primitive, the Audit sends an error detected to the SCU and generates a PSN207 log with reason “PRIM_NOT_SUPPORTED_BY_USERCLASS”.

Consider an example where `PSN_AUDIT_INTERVAL` is datafilled as 10 and `PSN_AUDIT_MAX_RETRY` is 5. `PSN_AUDIT_DROP_AGENTS` is datafilled as YES. The Audit is initiated every 10 minutes. The Audit checks for active PSN agents and increments the internal counter associated with each PSN agent. When the value of internal counter becomes equal to 5 (`PSN_AUDIT_MAX_RETRY`), the Audit sends a `query_Port` to the SCU. Assume for this example that the SCU returns an IDLE Port Status to the Audit. The Audit decodes the Port Status and disconnects the port (`PSN_AUDIT_DROP_AGENT = Y`). The register AUDPSM is pegged and log PSN400 with reason “SCU_PORT_STATUS_MISMATCH” is generated.

PSN logs

This chapter contains a detailed description of the PSN logs. For more information on logs, refer to the *UCS DMS-250 Logs Reference Manual*.

PSN log summary

PSN xxx log

A new class of informational logs, called PSN, is created for the PSN application.

Log PSN 100 is generated when the PSN encounters a maintenance problem on the data link.

Log PSN 101 is generated when the PSN data link is unable to process the message as received from the SCU due to incorrect message length, SPI version, or user class.

Log PSN 102 is generated when the PSN Data Comm Client is unable to process the message to be sent to the SCU due to unrecognized User Class sent by the PSN application.

Log PSN 103 is generated when a mandatory PSN table such as table SCUADDR is not datafilled.

Log PSN 104 is generated when the PSN receives a **Set_IP_Address** primitive that changes the SCU Arbitrator Address.

Log PSN 105 is generated when the PSN receives an **Invalid_Set_IP_Address** primitive or **Invalid_Sender_of_Heartbeat**.

Log PSN 106 is generated when SCU initiated Flow Control is activated or deactivated.

Log PSN 200 is generated when the PSN receives an **Error_Detected** primitive from the SCU, indicating that the PSN sent a message that contained an application or protocol error.

Log PSN 201 is generated when a `New_Call_Event` message to the SCU times out before a response primitive is received from the SCU.

Log PSN 202 is generated when decoding errors are found in primitives or macros from the SCU.

Log PSN 212 is generated when decoding errors are found in primitives or macros from the SCU. This is generated in conjunction with PSN 202.

Log PSN 203 is generated when a message from the SCU is not completed due to an invalid agent or connection state for the primitive received.

Log PSN 204 is generated when an unexpected internal DMS-250 message is received; the error may be fatal, which causes the call to be taken down, or non-fatal, so that the call survives.

Log PSN 205 is generated when a primitive from the SCU is not completed due to unavailable software resources.

Log PSN 206 is generated when a hardware resource is unavailable.

Log PSN 207 is generated when errors are found during the preliminary processing of a primitive sent from the SCU.

Log PSN 208 is generated when a primitive fails to be completed due to problems found by the finite state machine.

Log PSN 209 is generated when a macro fails due to macro decode failure.

The PSN 400 log report is generated to communicate errors encountered by, or actions taken by the PSN Audit process.

The PSN 401 log report is generated when the PSN receives a primitive “`Reset_Switch`” from the SCU.

AUD 599 log

The existing AUD599 log report is generated when a call that has a PSN Primitive or Scratchpad Extension Block associated with it completes abnormally.

AUD 617 log

The AUD617 log is associated with the PSN Scratchpad Extension Block.

TRKT 214 log

A new log is added to the TRKT class of logs. TRKT 214 is generated when the new treatment PSN call Failure (PSNF) is set.

PSN log description

The following logs are related to PSN.

PSN 100

Example

```
PSN100 APR14 15:52:09 8000 INFO PSN MAINTENANCE
REASON = UNABLE TO ESTABLISH SCU COMM
IPADDRESS = 47.122.72.10
PORT = 15
```

Overview

The PSN 100 log report is generated when the PSN encounters a maintenance problem on the data link.

Format

```
PSN100 APR14 15:52:09 8000 INFO PSN MAINTENANCE
REASON = <Text>
IPADDRESS = <IP Address>
PORT = < Integer>
```

Field descriptions

Field	Value	Description
REASON	"SCU Heartbeat Failure"	SCU Heartbeat failure
	"Unable to establish SCU Communication"	Problem with DMS-250 data communication layer
IPADDRESS	IP Address	Address of IP Connection
PORT	Integer	Port Number

Action to be taken

Not necessary.

Post-analysis

It is recommended that the problem at the SCU be investigated.

Internal

For "SCU Heartbeat Failure", the UCS DMS-250 begins polling.

“Unable to establish SCU communication” indicates that the UCS DMS-250 has initiated polling, but has not received a reply from the SCU with the arbitrator IP value.

OM register

Group PSN_ERDC – HBFAIL, NOSCOMM.

PSN 101

Example

```
PSN101 APR14 15:52:09 8000 INFO PSN MSG PROBLEM
REASON = WRONG VERSION
USERCLASS = 1
LENGTH = 1000
SPI VERSION = 1
```

Overview

The PSN 101 log report is generated when the UCS DMS-250 is unable to process the message as received from the SCU due to incorrect message length, SPI version, or user class.

Format

```
PSN101 APR14 15:52:09 8000 INFO PSN MSG PROBLEM
REASON = <Text>
USERCLASS = <byte>
LENGTH = <integer>
SPI VERSION = <nibble>
```

Field descriptions

Field	Value	Description
REASON	“Message Length exceeds maximum size” “Unrecognized SPI Version” “Unrecognized Userclass” “Client Registered Nil Mail Box” “Datacom header Corrupted”	SCU Message Problem
USERCLASS	User Class	Admin, CALLIP, Audit
LENGTH	Integer	Message Length
—continued—		

Field	Value	Description
SPI VERSION	1 to 15	SPI Version
—end—		

Action to be taken

No immediate action.

Post-analysis

It is suggested that when this log is generated, the SCU programming be modified to prevent the problem.

Internal

This is generated when the message as received from the SCU is unable to be processed.

OM register

Group PSN_ERDC – MSGSIZE, DCOMHDR, EMSGDROP.

PSN 102**Example**

```
PSN102 APR14 15:52:09 8000 INFO USER CLASS NOT REGISTERED
USERCLASS = 8
```

Overview

The PSN 102 log report is generated when the PSN Data Comm Client is unable to process the message to be sent to the SCU due to unrecognized User Class sent by PSN application.

Format

```
PSN102 APR14 15:52:09 8000 INFO USER CLASS NOT REGISTERED
USERCLASS = <byte>
```

Field descriptions

Field	Value	Description
USERCLASS	User Class	Admin, CALLP, Audit
—end—		

Action to be taken

No immediate action.

Post-analysis

It is suggested that the Nortel Networks support organization be notified to verify the proper registration of the user class.

Internal

This log is generated when the Data Comm Client does not recognize the PSN user class.

OM register

Group PSN_ERDC– MSGDROP, UAPPLMBS.

PSN 103**Example**

```
PSN103 APR14 15:52:09 8000 INFO PSN DATAFILL PROBLEM  
REASON = TABLE SCUADDR NOT DATAFILLED
```

Overview

The PSN 103 log report is generated when PSN Table SCUADDR is not datafilled.

Format

```
PSN103 APR14 15:52:09 8000 INFO PSN DATAFILL PROBLEM  
REASON = <text>
```

Field descriptions

Field	Value	Description
REASON	Table SCUADDR not datafilled	Verify the entry in Table SCUADDR
—end—		

Action to be taken

Verify the data in the table for proper entries.

Post-analysis

No immediate action.

Internal

This log is generated with the reason “Table SCUADDR NOT DATAFILLED” when the Data Comm Process cannot find the necessary entry in Table SCUADDR.

OM register

Not applicable.

PSN 104**Example**

```
PSN104 APR14 15:52:09 8000 INFO PSN MAINTENANCE
OLDIPADDRESS = 47.122.72.10
OLDPORT = 15
NEWIPADDRESS = 47.134.72.10
NEWPORT = 10
SEND AGENT DATA = TRUE
SPI VERSION = 2
```

Overview

The PSN 104 log report is generated when the PSN receives a `set_IP_Address` primitive that changes the SCU Arbitrator Address.

Format

```
PSN104 APR14 15:52:09 8000 INFO SCU PSN MAINTENANCE
OLDIPADDRESS= <IP Address>
OLDPORT= < Integer>
NEWIPADDRESS= <IP Address>
NEWPORT= < Integer>
SEND AGENT DATA= < Boolean >
SPI VERSION= <nibble>
```

Field descriptions

Field	Value	Description
OLDIPADDRESS	IP Address	Address of previous IP Connection
OLDPORT	Integer	Previous Port Number
NEWIPADDRESS	IP Address	Address of current IP Connection
NEWPORT	Integer	Current Port Number
—continued—		

Field	Value	Description
SEND AGENT DATA	TRUE (1) / FALSE (0)	Tells PSN to send Agent Data information now.
SPI VERSION	1 to 15	SPI Version
—end—		

Action to be taken

No immediate action.

Post-analysis

This is mostly an information log. However, if the SCU wants Agent Data information for all valid terminating trunks currently datafilled in table PSNROUTE, the “SEND AGENT DATA” field must be set to TRUE.

Internal

The PSN 104 log is generated when PSN changes the SCU Arbitrator Address as a result of the `set_IP_Address` primitive.

OM register

Group PSN_NOTF – AGNTDATA, STOPHTBT.

PSN 105**Example**

```
PSN105 APR14 15:52:09 8000 INFO INVALID INFO
REASON = Invalid Data for Set IPADDR
IPADDRESS = 47.122.72.10
PORT = 15
TRUNK = 670
MEMBER = 10
SPI VERSION = 1
```

Overview

The PSN 105 log report is generated when the PSN receives an `Invalid_Set_IP_Address` primitive or an `Invalid_Sender_of_Heartbeat`.

Format

```
PSN105 APR14 15:52:09 8000 INFO INVALID INFO
REASON = <text>
IPADDRESS= <IPAddress>
PORT= <Integer>
```

TRUNK= <Trunk number>
 MEMBER= <Member number>
 SPI VERSION= <SPI Version>

Field descriptions

Field	Value	Description
REASON	“ Invalid Data for Set IPADDR” “ Invalid Sender of Heartbeat”	
IPADDRESS	IP Address	Address of IP Connection
PORT	Integer	Port Number
TRUNK	Trunk number	Trunk Number
MEMBER	Member number	Member Number
SPI VERSION	1 to 15	SPI Version
—end—		

Action to be taken

No immediate action.

Post-analysis

This is an information log. No action is required.

Internal

The PSN 105 log is generated when PSN receives an **Invalid_Set_IP_Address** Primitive or **Invalid_Sender_of_Heartbeat**.

OM register

Not applicable.

PSN 106

Example

```
PSN106 APR14 15:52:09 INFO FLOW CONTROL
REASON = Flow Control Activated
FCSOURCE = SCU
DURATION = 128
```

GAP = 5
 ALL_NEW_CALLS_BLOCKED = FALSE

Overview

The PSN 106 log is generated every time SCU initiated flow control is activated or deactivated at the PSN.

Format

PSN106 APR14 15:52:09 INFO FLOW CONTROL
 REASON= <Text>
 FCSOURCE= <byte>
 DURATION = <integer>
 GAP = <integer>
 ALL_NEW_CALLS_BLOCKED = <TRUE/FALSE>

Field description

Field	Value	Description
REASON	"Flow Control Activated" "Flow Control Deactivated"	Flow control activated or deactivated
FCSOURCE	"SCU"	Flow control initiating source
DURATION	0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048	Flow control duration value by seconds
GAP	0, 1, 3, 5, 10, 20, 50, 100, 150, 300, 500, 800, 1200, 3000, 6000, 0	Flow control gap value in tenth of a second
ALL_NEW_CALLS_BLOCKED	"TRUE", "FALSE"	Whether all new calls have been blocked.
—end—		

Action to be taken

No immediate action.

Post-analysis

This is an information log. No action is required.

Internal

Fcsource "SCU" indicates that the Flow Control is initiated by the SCU.

OM register

Not Applicable.

PSN 200**Example**

```
PSN200 APR14 15:52:09 8000 INFO SCU ERROR MSG RCVD
REASON = ERROR IN PSN MESSAGE
USER CLASS = 2
ERR_CAUSE = BAD PRIMITIVE
IPADDRESS= 47.122.72.10
PORT= 15
SPI VERSION= 1
```

Overview

Log PSN 200 is generated when the PSN receives an **Error_Detected** primitive from the SCU, indicating that the PSN sent a message that contained an application or protocol error.

Format

```
PSN200 APR14 15:52:09 8000 INFO SCU ERROR MSG RCVD
REASON= <type of problem>
USERCLASS= < Integer>
ERR_CAUSE= < text>
IPADDRESS= <IP Address>
PORT= < Integer>
SPI VERSION= < SPI Version>
```

Field descriptions

Field	Value	Description
REASON	"Error in PSN Message"	PSN Message problem
USERCLASS	Integer	PSN Userclass. Always printed.
—continued—		

8-12 PSN logs

Field	Value	Description
ERR_CAUSE	"Nil Error Cause"	Indicates the error cause – always printed.
	"Header Decode Failure"	If error_cause > max_num_of_error then
	"Bad Macro Tag"	"Nil Error Cause" is printed.
	"Bad Primitive"	
	"Missing Mandatory Parameter"	
	"Mand Param Decode Failure"	
	"Optional Param Decode Failure"	
	"Out of Range Parameter"	
	"Primitive Not Supported for User Class"	
	"Number of Primitives Exceeded Maximum Allowed"	

—continued—

ERR_CAUSE	"Missing Mandatory Siginfo Parameter"
	"Not PSN agent"
	"Port is Not Datafilled in Table PSNROUTE"
	"Agent Not Supported"
	"Port down due to warm restart"
	"Prim Invalid for Current Port Status"
	"Unexpected Msg"
	"STR Not Available"
	"UTR Not Available"
	"Conference Circuit Not Available"
	"Message Not Available"
	"CCB Not Available"
	"Primitive Extension Block Not Available"
	"ScratchPad Extension Block Not Available"
	"Software Resources Unavailable"
	"Message Failure"
	"Software Error"
	"Not Min Num Ports to Bridge"
	"Max ports to bridge exceeded"
	"Bearercap Incompatible"
	"Message IDx not in PSNmsgix"
	"Unsupported Signalling Type"
	"Duplicate SigInfo Message"
	"Bad Agent State"
	"Agent Termination Failure"
	"Abnormal Exit"
	"Message Not Playing"
	"Tone Duration Unsupported"
	"Prompt Failure"

Field	Value	Description
	"Digit Collection Failure" "Q764 Protocol Problem" "Agent is Not Datafilled in Table TRKGRP" "Invalid Flow Control Duration Or Gap" "Unexpected Flow Control Message"	
IPADDRESS	IP Address	Address of IP Connection
PORT	Integer	Port Number
SPI VERSION	1 to 15	SPI Version
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the Nortel Networks support organization be notified about the encoding problem on the UCS DMS-250.

Internal

This log is caused by receiving a message from the SCU, indicating that an erroneous message was sent from the UCS DMS-250.

OM register

Group PSN_PRIM – ERRDETP

PSN 201**Example**

```
PSN201 APR14 15:52:09 8000 INFO SCU TIME OUT
TRUNK = 670
MEMBER = 15
```

Overview

The PSN 201 log report is generated when a **New_Call** event message to the SCU times out before a response primitive is received from the SCU.

Format

PSN201 APR14 15:52:09 8000 INFO SCU TIME OUT
 TRUNK = <trunk number>
 MEMBER = <trunk member number>

Field descriptions

Field	Value	Description
TRUNK	Trunk number	trunk number on which error occurred
MEMBER	Trunk member number	external trunk member number
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the values of the `SCU_TIMEOUT` office parameter be increased.

Internal

This log is caused by not receiving a `New_Call` message from SCU in time specified.

OM register

Group PSN_ERFM – SCUTMOUT.

PSN 202**Example**

PSN202 APR14 15:52:09 8000 INFO PSN DECODE ERROR 1
 REASON= Out of Range Parameter
 TRUNK= 670
 MEMBER= 15
 PRIMITIVE= CONNECT
 PARAMETER=DESTINATION TRUNK GROUP
 SUBPARM= TRUNK GROUP NUMBER
 SESSION ID= 155
 USERCLASS= 1
 SPI VERSION= 1

Overview

The PSN 202 log report is generated when decoding errors are found in primitives or macros from the SCU.

Format

PSN202 APR14 15:52:09 8000 INFO PSN DECODE ERROR 1

REASON = <text>

TRUNK = <trunk number>

MEMBER = <trunk member number>

PRIMITIVE = <PSN primitive>

PARAMETER = <PSN parameter>

SUBPARM = <PSN subparameter>

SESSION ID = <session id from SCU>

USERCLASS = <integer>

SPI VERSION = <SPI Version>

Field descriptions

Field	Value	Description
REASON	Any Error Cause value.	<p>Explanations why decoding SCU message failed.</p> <p>Indicates the error cause – always printed. Error Cause is passed as the reason for this log.</p>
TRUNK	Trunk name	<p>Trunk number on which error occurred.</p> <p>Indicates the PSN trunk group number. Printed only if the Reason = “Out of Range Parameter” and all the parameters defined before the port info parm have been successfully decoded.</p>
MEMBER	Trunk member number	<p>External trunk member number.</p> <p>Indicates trunk member number. Printed only if the Reason = “Out of Range Parameter” and all the parameters defined before the port info parm have been successfully decoded.</p>
PRIMITIVE	PSN primitive	PSN primitive. Always printed.
PARAMETER	PSN parameter	PSN parameter. Always printed.
—continued—		

Field	Value	Description
SUBPARM	PSN subparameter	PSN subparameter. Indicates the failed subparameter for signaling info parameters. Printed only if the Reason = "Out of Range Parameter".
SESSION ID	Integer	Session ID from SCU. Printed only if Reason = "Out of Range Parameter" and all the parameters before sessionid parm have been successfully decoded.
USERCLASS	Integer	PSN Userclass. Always printed.
SPI VERSION	1 to 15	SPI Version
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that programming of the SCU be investigated.

Internal

This is caused by a poorly encoded message from the SCU.

OM register

Group PSN_ERPS – BADPRIM, BADPARM, MSMPARM, OFRPARM, MANDPDEF, OPPRMDEF, DECODEFL.

PSN 203**Example**

```
PSN203 APR14 15:52:09 8000 INFO PSN APPL ERROR
TRUNK = 671
MEMBER = 15
SESSION ID = 123
AGENTSTATE = IDLE
CONSTATE = HELD
PRIMITIVE = BRIDGE
```

Overview

The PSN 203 log report is generated when a message from the SCU is not completed due to an invalid agent or connection state for the primitive received.

Format

PSN203 APR14 15:52:09 8000 INFO PSN APPL ERROR
TRUNK = <trunk number>
MEMBER = <trunk member number>
SESSION ID = <session id from SCU>
AGENTSTATE = <agent state>
CONSTATE = <connection state>
PRIMITIVE = <PSN primitive>

Field descriptions

Field	Value	Description
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	external trunk member number
SESSION ID	Integer	Session ID from SCU
AGENTSTATE	Agent State	Agent State
CONSTATE	Connection State	Connection State
PRIMITIVE	PSN primitive	PSN primitive
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the programming of the SCU be investigated.

Internal

This log is generated when the instructions from the SCU cannot be completed due to an invalid agent or connection state.

OM register

Group PSN_ERFM – BADACST, PRIMSTFL.

PSN 204**Example**

```
PSN204 APR14 15:52:09 8000 INFO UNEXPECTED ERROR MSG
TYPE = FATAL
TRUNK = 672
MEMBER = 15
SESSION ID = 123
AGENTSTATE = IDLE
CONSTATE = HELD
```

Overview

The PSN 204 log report is generated when an unexpected internal UCS DMS-250 message is received. The error is classified as fatal or non-fatal. A fatal error causes the call to be taken down; however, the call survives a non-fatal error.

Format

```
PSN204 APR14 15:52:09 8000 INFO UNEXPECTED ERROR MSG
TYPE = <fatal/non-fatal>
TRUNK = <trunk number>
MEMBER = <trunk member number>
SESSION ID = <session id from SCU>
AGENTSTATE = <agent state>
CONSTATE = <connection state>
```

Field descriptions

Field	Value	Description
TYPE	"Fatal" "Non-Fatal"	Explanations of the error type
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	external trunk member number
SESSION ID	Integer	Session ID from SCU
AGENTSTATE	Agent State	Agent State
CONSTATE	Connection State	Connection State
—end—		

Action to be taken

No immediate action.

Post-analysis

It is suggested that the NT support organization be notified to investigate the unexpected error message condition.

Internal

This log is generated when an unexpected message from the SCU is received.

OM register

Group PSN_ERFM - UNEXPENF, UNEXPEF

PSN 205**Example**

```
PSN205 APR14 15:52:09 8000 INFO SOFTWARE RES
UNAVAILABLE
REASON= Scratchpad Extension Block Not Available
TRUNK= 670
MEMBER= 15
SESSION ID= 123
```

Overview

The PSN 205 log report is generated when a message from the SCU is not completed due to unavailable software resources.

Format

```
PSN205 APR14 15:52:09 8000 INFO SOFTWARE RES
UNAVAILABLE
REASON = <text>
TRUNK = <trunk number>
MEMBER = <trunk member number>
SESSION ID = <session id from SCU>
```

Field descriptions

Field	Value	Description
REASON	"CCB Not Available" "Primitive Extension Block Not Available" "Scratchpad Extension Block Not Available"	Explanations of the software resource error

Field	Value	Description
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	external trunk member number
SESSION ID	Integer	Session ID from SCU
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the NT support organization be notified to investigate the software resource issue.

Internal

This log is generated when the software resources are not available.

OM register

Group PSN_ERFM - PRMEXTUN, SCREXTUN

PSN 206**Example**

```
PSN206 APR14 15:52:09 8000 INFO HARDW RES UNAVAILABLE
REASON = NO UTR
```

Overview

The PSN 206 log report is generated when a hardware resource is not available.

Format

```
PSN206 APR14 15:52:09 8000 INFO HARDW RES UNAVAILABLE
REASON = < text>
```

Field descriptions

Field	Value	Description
REASON	"No UTRs Available" "No STRs Available" "No Conference Ports Available" "No Idle Messages Available"	Explanations of the hardware resources error
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the hardware resources problem be investigated.

Internal

This log is generated when the hardware resources are not available.

OM register

Group PSN_ERFM – NOUTR, NOSTR, NOCNF, NOIDLMSG.

PSN 207

Example

```
PSN207 APR14 1:52:00 8000 INFO PSN PRIMITIVE PROCESS ERROR
```

```
REASON = MAX PORTS TO BRIDGE EXCEEDED
```

```
USERCLASS = 2
```

```
IPADDRESS = 47.122.72.10
```

```
PORT = 15
```

```
TRUNK = 672
```

```
MEMBER = 15
```

```
PRIMITIVE = CONNECT
```

```
SESSION ID = 155
```

Overview

The PSN 207 log report is generated when errors are found during the preliminary processing of a primitive sent from the SCU.

Format

```
PSN207 APR14 1:52:09 8000 INFO PSN PRIMITIVE PROCESS  
ERROR
```

REASON = <text>
 USERCLASS = <integer>
 IPADDRESS = <ip address>
 PORT = <integer>
 TRUNK = <trunk number>
 MEMBER = <trunk member number>
 PRIMITIVE = <PSN primitive>
 SESSION ID= <session id from SCU>

Field descriptions

Field	Value	Description
REASON	"Port is Not Datafilled in Table PSNROUTE" "Max Ports to Bridge Exceeded" "Not Active PSN Agent" "Not Min Num Ports to Bridge" "Msg IDX not in PSNMSGIX" "Prim Not Supported for Userclass" "Tone Duration Value Not Supported" "Agent is Not Datafilled in Table TRKGRP" "BBF Not Possible"	Explanations why primitive processing of SCU message failed
USERCLASS	Integer	PSN userclass
IPADDRESS	IP Address	Address of IP Connection
PORT	Integer	Port Number
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	External trunk member number
PRIMITIVE	PSN primitive	PSN primitive
—continued—		

Field	Value	Description
SESSION ID	Integer	Session ID from SCU
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that programming and/or datafill of the SCU be investigated.

Internal

This is caused by an error in the primitive from the SCU.

OM register

Group PSN_ERPS – PRMUSRMS, PSNRTFL, AGNACT, MAXBREX, NMINNOBP, PSNMSGFL.

PSN 208**Example**

```
PSN208 APR14 15:52:09 8000 INFO PRIMITIVE FAILURE
REASON = Message Failure
TRUNK = 670
MEMBER = 3
PRIMITIVE = HOLD
```

Overview

Log PSN 208 is generated when a primitive fails to be completed due to problems found by the finite state machine.

Format

```
PSN208 APR14 15:52:09 8000 INFO PSN PRIMITIVE FAILURE
REASON = <text>
TRUNK = <trunk number>
MEMBER = <trunk member number>
PRIMITIVE = <PSN primitive>
```

Field descriptions

Field	Value	Description
REASON	"Bearer Capability Incompatible" "Digit Collection Failed" "Agent Not Available" "Prompt Message Failure" "Agent Not Supported" "Message Failure" "Primitive Not Supported in Current Agent State" "Unsupported Signalling Type for Agent" "Duplicate Initial SigInfo for Agent" "Software Error" "Agent is Not Datafilled in Table TRKGRP" "Missing Mandatory Siginfo Parameter"	Type of problem causing the primitive failure
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	External trunk member number
PRIMITIVE	PSN primitive	PSN primitive
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the data at the SCU and at the PSN be investigated.

Internal

This log is generated when the PSN finite state machine is unable to process a primitive.

OM register

Group PSN_ERFM – BADAGST, UNSUPTRK, BCINCOMP,,
DIGCOLFL, AGNAVAIL, MSGFL, PROMPTFL, UNSIGTYP, DUPMSG,
AGNDTKGP, SFTWERR, MMSIPARM.

PSN 209**Example**

```
PSN209 APR14 15:52:09 8000 INFO PSN MACRO DECODE ERROR
REASON = Num of Prim Exceeded Maximum Allowed
NUMBER = 3
USERCLASS = 2
```

Overview

Log PSN 209 is generated when there is a decoding error due to the number of primitives in a macro that has exceeded the maximum allowed.

Format

```
PSN209 APR14 15:52:09 8000 INFO PSN MACRO DECODE ERROR
REASON = <text>
NUMBER = <number of primitives>
USERCLASS = <Integer>
```

Field descriptions

Field	Value	Description
REASON	"Number of Primitives Exceeded Maximum Allowed"	Decode error
NUMBER	Number of primitives	Number of primitives in the macro
USERCLASS	PSN userclass	PSN primitive
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the data at the SCU and at the PSN be investigated.

Internal

This log is generated when the PSN detects the number of primitives has exceeded the maximum allowed in a macro.

OM register

Group PSN_ERPS – MAXPMEXC.

PSN 212**Example**

```
PSN212 APR14 15:52:09 8000 INFO PSN DECODE ERROR 2
FIELDID = 25
HEXMSGINDEX = 135
HEXDUMP
```

```
FFFF 0000 3384 A640 0002 0001 2E91 91A8 0000 0000
0000 0000 FDFD 0000 0000 0000 0000 0000 0000 0000
3312 1152 0011 0000 00A0 3312 1152 0011 0000 00A0
0000 0000 0000 0000 0000 0000 0017 0200 0000 0000
0000 0000 0000 0000 2E91 85A8 0000 001B 3800 0200
0000 0000 0000 0000 0000 0000 0000 0000 0002 0000
0000 0000 0000 0002 0000 1AAA 0000 2E91 85A8 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000
```

Overview

The PSN 212 log report is generated when decoding errors are found in primitives or macros from the SCU. This is generated in conjunction with PSN 202.

Format

```
PSN212 APR14 15:52:09 8000 INFO PSN DECODE ERROR 2
FIELDID = <fieldid_type>
HEXMSGINDEX = <integer>
HEXDUM = < Hexadecimal dump of the primitive where the error occurs.
HEXMSGINDEX is the count of the byte of that primitive where the
decoding error occurred.>
```

Field descriptions

Field	Value	Description
FIELDID	Integer	FieldID Type. Indicates the field of a multifield parameter that failed to decode and is printed only if Reason = "Out of Range Parameter" (see PSN202).
HEXMSGINDEX	Hex Message Error Index	Count where the decoding error occurs. Indicates the byte of BER string that failed to decode. This field is not valid if Reason = "Out of Range Parameter" (see PSN202).
HEXDUMP	50 HEX bytes	50 bytes of the message with the error. Part of BER string that failed to decode.
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that programming of the SCU be investigated.

Internal

This is caused by a poorly encoded message from the SCU.

OM register

Group PSN_ERPS – BADPRIM, BADPARM, MSMPARM, OFRPARM, MANDPDEF, OPPRMDEF, DECODEFL.

PSN 400**Example**

```
PSN400 APR14 15:52:09 8000 INFO PSN AUDIT
REASON = SCU Port Status mismatch
TRUNK = 670
MEMBER = 15
```

Overview

The PSN 400 log report is generated to communicate errors encountered by, or actions taken by, the PSN Audit process.

Format

PSN400 APR14 15:52:09 8000 INFO PSN AUDIT

REASON = < text>

TRUNK = <trunk number>

MEMBER = <trunk member number>

Field descriptions

Field	Value	Description
REASON	"Discn TRK Audit Fails to Receive Port Status" "Info Only — Audit Fails" "SCU Port Status mismatch" "Info Only — Audit Fails — Nil Trk Grp" "Info Only — Audit Fails — Nil Trk Mem" "Info Only — Audit Fails — Nil Trk Invalid Callid"	Explanations of the audit process error
TRUNK	Trunk name	trunk number on which error occurred
MEMBER	Trunk member number	external trunk member number
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the reason the SCU did not reply or had Port Status mismatched, be investigated.

Internal

This log is generated when the PSN Audit process fails.

OM register

Group PSN_ERPS – AUDPSF, AUDPSM.

PSN 401**Example**

```
PSN401 APR14 15:52:09 8000 INFO PSN RESET SWITCH
REASON = Info Only — Port Down
TRUNK = 670
MEMBER = 15
```

Overview

The PSN 401 log report is generated when the PSN receives the primitive “Reset the switch” from the SCU.

Format

```
PSN401 APR14 15:52:09 8000 INFO PSN RESET SWITCH
REASON = <text>
TRUNK = <trunk number>
MEMBER = <trunk member number>
```

Field descriptions

Field	Value	Description
REASON	“Port Down Reset Switch” “Info Only — Reset Switch”	Explanations of the error
TRUNK	Trunk name	Trunk number on which error occurred
MEMBER	Trunk member number	External trunk member number
—end—		

Action to be taken

No immediate action.

Post-analysis

It is recommended that the reason be investigated.

Internal

This log is generated when the PSN resets the switch.

OM register

Group PSN_ERPS – PTDNRS.

AUD 599

Example

```
AUD599 MAY08 13:11:36 6900 INFO EXT DUMP 00DA 0000
FFFF 0000 3384 A640 0002 0001 2E91 91A8 0000 0000
0000 0000 FDFD 0000 0000 0000 0000 0000 0000 0000
3312 1152 0011 0000 00A0 3312 1152 0011 0000 00A0
0000 0000 0000 0000 0000 0000 0017 0200 0000 0000
0000 0000 0000 0000 2E91 85A8 0000 001B 3800 0200
0000 0000 0000 0000 0000 0000 0000 0000 0002 0000
0000 0000 0000 0002 0000 1AAA 0000 2E91 85A8 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000
```

Overview

The AUD 599 log report is generated every time a call that has a PSN Primitive or Scratchpad Extension Block associated with it completes abnormally.

Format

```
AUD599 MAY08 13:11:36 6900 INFO EXT DUMP <hex dump>
```

Field description

Not Applicable.

Action to be taken

No immediate action to be taken.

Post-analysis

No post analysis.

Internal

This log is an existing log.

TRKT 214

Example

```
TRKT214 APR14 15:52:09 8000 INFO TRTMT
CKT UEAN2WMFWK91
TREATMENT SET = PSNF
CALLED NO = 8009901234
CALLID = 123456
```

Overview

The TRKT 214 log report is generated when the PSN call fails, for example, New Call Time Out, No PSN Extension Block, New Call Rejected, and is routed to PSNF treatment.

Format

```
TRKT214 APR14 15:52:09 8000 INFO TRTMT
CKT <trunk clli name>
TREATMENT SET = <trtmt assigned to trunk>
CALLED NO = <calledno>
CALLID = <callid>
```

Field description

Field	Value	Description
CKT	Trunk clli name	Identifies originating trunk equipment
TREATMENT SET	"PSNF"	Identifies treatment assigned to trunk (PSN call fails)
CALLED NO	Called Number	Provides terminating line directory number if dialed before treatment was assigned
CALLID	Call ID	Provides number uniquely identifying the call
—end—		

Action to be taken

No action is recommended. The log is for informational purposes only.

Post-analysis

No post analysis.

Internal

The TRKT log is an existing log. Log TRKT 214 is generated when treatment PSNF is set to New Call Time Out, No Extension Block, or New Call Rejected.

OM register

TRMTFR3: register TFRPSNF.

PSN operational measurements

This chapter contains a detailed description of the PSN Operational Measurements (OMs). It also includes an association of the PSN Logs and OMs. For more information on OMs, refer to the *UCS DMS-250 Operational Measurements Reference Manual*.

OMs

There are six PSN OM Groups. These groups are dedicated mostly to monitoring the traffic between the SCU and the PSN, and monitoring any errors that occur.

OM group summary

GROUP NAME (acronym)	GROUP NAME (expanded)	NEW, CHANGED or DELETED	REASON
PSN_USAG	Programmable PSN Service Node Usage	New	Messages received/sent from the SCU
PSN_PRIM	Programmable PSN Service Node Primitives received	New	Primitives received from the SCU
PSN_NOTF	Programmable PSN Service Node Notifications sent	New	Notifications sent to the SCU
PSN_FCTR	Programmable PSN Service Node Flow Control	New	Calls blocked due to Flow Control
PSN_ERDC	Programmable PSN Service Node Error — data comm level	New	Error Detected in data comm layer
—continued—			

9-2 PSN operational measurements

GROUP NAME (acronym)	GROUP NAME (expanded)	NEW, CHANGED or DELETED	REASON
PSN_ERPS	Programmable PSN Service Node Error — primitive processing, decoding level	New	Error Detected in Decoding or primitive processing
PSN_ERFM	Programmable PSN Service Node Error in Finite State Machine	New	Error Detected in Finite State Machine
EXT	Extension Block	Changed	Added a New Extension Block Register
TRMTFR3	PSN Call Treatment	Changed	Added a PSNF Call Treatment Register
—end—			

OM register summary

PSN_USAG. Registers in this group are pegged every time a message is sent or received by the PSN. The following table lists the registers within this group:

Register	To record the number of:
SMSGRCVD	Messages received from the SCU
SMSGSENT	Messages sent to the SCU
SMACRCVD	Messages with macros received from the SCU

PSN_PRIM. Registers in this group are pegged on the receipt of a primitive from the SCU. There is one register for each PSN primitive. The following table lists the registers in this group:

Register	To record the number of:
CDRPT	Collect_Digits_&_Report received from the SCU
CNECT	Connect received from the SCU
DISCNECT	Disconnect received from the SCU
HOLD	Hold received from the SCU
MONITOR	Monitor received from the SCU
MUTE	Mute received from the SCU
PLAYMSG	Play_Message received from the SCU
PPCDRPT	Play_Prompt_Collect_Digits_&_Report received from the SCU
QURYPOR	Query_Port received from the SCU
RECONNECT	Reconnect received from the SCU
SETBLREC	Set_Billing_Record received from the SCU
STOPMSG	Stop_Message received from the SCU
XSIGINFO	Transmit_Siginfo received from the SCU
BRIDGE	Bridge received from the SCU
NCALLACC	New_Call_Accepted received from the SCU
NCALLREJ	New_Call_Rejected received from the SCU
HEARTBT	Heartbeat received from the SCU
ERRDETP	Error_Detected received from the SCU
QURYTMDY	Query_Time_of_Day received from the SCU
—continued—	

9-4 PSN operational measurements

Register	To record the number of:
RSETSWCH	Reset_Switch received from the SCU
SETIPADD	Set_IP_Address received from the SCU
PRTSTAT	Port_Status received from the SCU
FLOWCTRL	Flow_Control received from the SCU
—end—	

If one or more of the above primitives is received as a part of a macro, then the SMACRCVD register (PSN_USAG) is pegged in addition to the registers mentioned above. For example, a Reconnect and a Mute received in the macro peg registers SMACRCVD, RECNECT and MUTE.

PSN_NOTF. Registers in this group are pegged whenever an event notification message is sent to the SCU. There is one register for each PSN event notification message. The following table lists the registers in this group:

Register	To record the number of:
DIGCOL	Digit_Collected sent to the SCU
ERRDET	Error_Detected sent to the SCU
INSTMPL	Instruction_Completed sent to the SCU
MSGPLY	Message_Played sent to the SCU
NEWCALL	New_Call sent to the SCU
OFFHOOK	Off_Hook sent to the SCU
ONHOOK	On_Hook sent to the SCU
PORTSTAT	Port_Status sent to the SCU
—continued—	

Register	To record the number of:
RTEUNAV	Route_Not_Available sent to the SCU
RTESEL	Route_Selected sent to the SCU
SIGEVENT	Signaling_Event sent to the SCU
TONEDT	Tone_Detected sent to the SCU
CURTMDY	Current_Time_of_Day sent to the SCU
INSERVCE	In_Service sent to the SCU
QRYPORT	Query_Port sent to the SCU by the PSN audit.
AGNTDATA	Agent_Data sent to the SCU
STOPHTBT	Stop_Heartbeat sent to the SCU
—end—	

PSN_FCTR. This group of registers is pegged when a call is blocked because of Flow Control. The following table lists the register in this group:

Register	To record the number of
SCUCDROP	Calls blocked due to flow control initiated by the SCU

PSN_ERDC. This group of registers is pegged when an error is detected in the PSN Data Communications. The following table lists the registers in this group:

Register	To record the number of times:
MSGSIZE	The SCU message size exceeded the maximum size.
—continued—	

9-6 PSN operational measurements

Register	To record the number of times:
HBFAIL	Heartbeat Failure occurred
NOSCOMM	The SCU communication has failed
DCOMHDR	There is an error in the SCU Datacom Message header
UAPPLMBS	There is a problem in the user application mailbox
MSGDROP	The data communication has dropped the internal message
EMSGDROP	The data communication has dropped an external message
—end—	

PSN_ERPS. This group of registers is pegged when an error is detected in the PSN Primitive Server. The following table lists the registers in this group:

Register	To record the number of times:
MSMPARM	A mandatory parameter is missing in the received primitive
OFRPARM	An out of range parameter is received from the SCU
PRMUSRMS	There is a user class mismatch
MBPRDCC	There is a problem with the Data Communication mailbox
MBPRPSA	There is a problem with the Audit mailbox
AUDPSF	Audit fails to receive a Port Status
AUDPSM	There is a mismatch between the Port Status received from the SCU and the status of the port on the PSN
—continued—	

Register	To record the number of times:
PSNRTFL	The destination trunk is not found in table PSNROUTE
AGNACT	The port received is not an active PSN Agent
MAXBREX	Maximum number of ports to bridge has exceeded
MANDPDEF	There is a failure in mandatory parameter decoding
OPPRMDEF	There is a failure in optional parameter decoding
NMINNOBP	A bridge primitive is sent with less than the required minimum number of ports to bridge
PSNMSGFL	The message index in the Play Message, Stop Message, Bridge, or the PPCD primitive is not datafilled in the PSNMSGIX table
PTDNRS	Port is taken down due to reset switch from the SCU
PDRESTWM	The port is brought down due to a warm restart
DECODEFL	The decode of the header failed
BADMACRT	The macro tag is bad
MAXPMEXC	The # of primitives in the macro exceeded the maximum allowed
—end—	

PSN_ERFM. This group of registers is pegged when an error is detected in the PSN Finite State Machine. The following table lists the registers in this group:

Register	To record the number of times:
SCUTMOUT	The SCU timeout has occurred
—continued—	

9-8 PSN operational measurements

Register	To record the number of times:
BADAGST	The primitive is not supported for the current Agent state
UNSUPTRK	The destination trunk is not a supported type for PSN
AGNAVAIL	An idle member of a trunk group is not available for termination
UNEXPENF	There is a non-fatal unexpected message error
UNEXPEF	There is a fatal unexpected message error
NOUTR	A UTR is not available for digit collection
NOSTR	An STR is not available for tone monitoring
NOCNF	A 6-port conference circuit is not available for bridging
NOIDLMSG	A Message is not available for Play Message or Bridge
CCBUN	No CCBs available
PRMEXTUN	The Primitive Extension Block is unavailable
SCREXTUN	The ScratchPad Extension Block is unavailable
BBFNOPOS	BBF on the agent is not possible
BCINCOMP	The BCs of the originator/terminator is not compatible
DIGCOLFL	There is a failure to start or stop digit collection on a port
PRIMSTFL	The received primitive is invalid for the current port state
PROMPTFL	There is a failure to play a message/prompt
MSGFL	There is a failure to stop the message
UNSIGTYP	The SigInfo is not supported for this Agent
—continued—	

Register	To record the number of times:
DUPMSG	Two IAMs/SETUPs sent for the same Agent
SFTWERR	An internal error occurred during primitive processing
AGNDTKGP	The PSN Agent is not datafilled in the table TRKGRP
MMSIPARM	Missing mandatory SigInfo parameter
—end—	

EXT. This is an existing OM group, with two new registers PSN_PRIMITIVE_EXT and PSN_SCRATCHBOARD_EXT added to count the number of times a PSN_PRIMITIVE_EXT_BLK or a PSN_SCRATCHPAD_EXT_BLK is used. These registers keep track of the cumulative and regular counts in the fields EXTSEIZ and EXTHI, respectively.

TRMTFR3. This is an existing OM group, with a register added to count the number of times a new call is rejected and the port is routed to the PSNF treatment.

PSN OM and log associations

The following is a list of the PSN Logs and their associated OMs.

PSN100. These logs are used to record a failure in the Data Communications.

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN100	PSN_ERDC	HBFAIL, NOSCOMM,	See OMs definition
PSN101	PSN_ERDC	MSGSIZE, DCOMHDR, EMSGDROP	See OMs definition
PSN102	PSN_ERDC	MSGDROP, UAPPLMBS	See OMs definition
—end—			

9-10 PSN operational measurements

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN103	none	not applicable	SCUADDR not datafilled
PSN104	PSN_NOTF	AGNTDATA, STOPHTBT	To record change in the Arbitrator IP Address.
PSN105	none	not applicable	On receiving an invalid Set IP Address primitive or an Invalid Sender of Heartbeat
PSN106	none	not applicable	To log the activation and deactivation of SCU initiated Flow Control
—end—			

PSN200. These logs are used to record a failure in encoding and decoding of the message, primitive server, and call processing utilities.

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN200	PSN_NOTF	ERRDETP	To record the number of non-fatal errors detected by the SCU
PSN201	PSN_ERFM	SCUTMOUT	To record the number of SCU timeouts
PSN202 PSN212	PSN_ERPS	MSMPARM, OFRPARG, MANDPDEF, OPPRMDEF, DECODEFL, BADMACRT	To record the decoding errors in the primitives received from the SCU
—continued—			

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN203	PSN_ERFM	PRIMSTFL	To record an invalid Agent/Connection state for the port in the PSN
PSN204	PSN_ERFM	UNEXPENF, UNEXPEF	To record the receipt of an unexpected peripheral message for the port at the PSN
PSN205	PSN_ERFM	CCBUN, PRMEXTUN, SCREXTUN	To record the unavailability of software resources (for example, ext. blocks)
PSN206	PSN_ERFM	NOSTR, NOUTR, NOCNF, NOIDLMSG	To record the fact that the hardware resource is unavailable
PSN207	PSN_ERPS	PRMUSRMS, PSNRNFL, MAXBREX, NMINNOBP, PSNMSGFL, AGNACT BBFNOPOS	To record the errors found during the preliminary processing of a primitive sent from the SCU.
PSN208	PSN_ERFM	UNSUPTRK, BADAGST, BCINCOMP, DIGCOLFL, AGNAVAIL, PROMPTFL, MSGFL, UNSIGTYP, DUPMSG, SFTWERR, AGNDTKGP, MMSIPARM	To record the failure to complete the execution of a primitive due to internal PSN FSM issues.
—continued—			

9-12 PSN operational measurements

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN209	PSN_ERPS	MAXPMEXC	To record the fact that the # of primitives in the macro exceeded the maximum allowed.
PSN212	PSN_ERPS	see PSN202	This log records the other half of the PSN202 log.
—end—			

PSN400. These logs are used to record a failure related to the PSN Audit process as well as the processing of the Reset Switch primitive.

Logs	Associated OM Group	Associated OMs	Purpose of the log
PSN400	PSN_ERPS	AUDPSM, AUDPSF	See OMs definition
PSN401	PSN_ERPS	PTDNRS	See OMs definition
—end—			

PSN office parameters

This chapter contains the detailed description of PSN Office Parameters and PSN tables. For more information, refer to the *UCS DMS-250 Office Parameters Reference Manual* and the *UCS DMS-250 Data Schema Reference Manual*.

Office parameters

The following Office Parameters have been created to control the PSN functionality on the PSN. These parameters are datafilled in tables OFCVAR or OFCENG.

Office Parameter	What it is used for:
PSN_AUDIT_DROP_AGENTS	If "Y", the agents or ports are disconnected or idled upon Audit Failure (an invalid callid for the port, or a Port Status with status = idle returned from the SCU)
PSN_AUDIT_INTERVAL_TIME	The time PSN will delay between 2 consecutive agent audits
PSN_AUDIT_MAX_RETRY	The maximum number of retries the Audit process will execute before verifying the status of a port by sending a Query Port to the SCU
PSN_CALLS_ALLOWED	If "Y" New Calls are allowed to enter the server mode, else the port is routed to PSNF treatment
PSN_DROP_AGENTS_SCU_SHELF_RESET	Indicates whether the PSN ports are dropped upon a shelf reset at the SCU If "Y", all the ports serviced by the shelf in the Reset Switch primitive are disconnected or idled
—continued—	

10-2 PSN human machine interface

Office Parameter	What it is used for:
PSN_DROP_AGENTS_SCU_SRVC_RESET	Indicates whether the PSN ports are dropped upon a service reset at the SCU. If "Y", all the ports serviced by the service in the Reset Switch primitive are disconnected or idled.
PSN_DROP_AGENTS_SCU_SYS_RESET	Indicates whether the PSN ports are dropped upon a system reset at the SCU. If "Y", all PSN ports are disconnected or idled.
PSN_EVENT_TIMER	to set the maximum amount of time the PSN will wait for a message from the SCU after a New Call event has been sent to the SCU. When the timer expires without a response, the call is terminated.
PSN_FLOW_CTRL_MESSAGING	If "Y" this parameter turns ON the SCU Initiated Flow Control and the messaging related to PSN Initiated Flow Control
PSN_HEARTBEAT_WAIT_TIME	The time the PSN waits to on a Heartbeat primitive from the SCU arbitrator before it initiates the initial contact polling to get a new arbitrator address
PSN_INIT_SCU_POLLING	If "Y", this parameters specifies that the initial contact polling is to be initiated
PSN_INTER_POLL_TIME	Is used to set the maximum amount of time the PSN will wait between 2 consecutive polling cycles
PSN_MAX_MEMBER_ADVANCE	This parameter is used to control the number of reselection of IDLE trunks for a Connect primitive before a Route Not Available is sent.
PSN_PERFORM_NEWCALL_DIGCOL	This parameter is used to turn digit-collection during New-Call Events on or off. Digit-Collection is turned OFF if this office parameter is set to FALSE (N).
PSN_PRIMITIVE_NUM_EXT_BLOCKS	The number of primitive extension blocks, one per PSN port
PSN_SCRATCHPAD_NUM_EXT_BLOCKS	The number of scratchpad extension blocks, one per PSN port.
—continued—	

Office Parameter	What it is used for:
PSN_SPI_LOGS_ON	This parameter is used to turn the PSNE and PSNP logs ON and OFF. These logs are displayed for every primitive and event for the PSN if the office parameter is set to "y".
—end—	

PSN_AUDIT_INTERVAL_TIME

It is the time interval between two consecutive agent audits at the PSN. Its range is from 0 to 30 minutes. A value of 0 turns the Audit off.

PSN_AUDIT_MAX_RETRY

The Audit scans the Agent at every PSN_AUDIT_INTERVAL_TIME, and increments an internal counter until it reaches PSN_AUDIT_MAX_RETRY. At that point, the Audit sends a Query Port event to the SCU. Its range is from 0 to 5.

PSN_AUDIT_DROP_AGENTS

Specifies whether Agents can be released by the Agent Audit. Its possible values are YES and NO. If PSN_AUDIT_DROP_AGENTS is YES, then no notification is sent to the SCU when the agent is taken down.

PSN_CALLS_ALLOWED

Specifies whether or not SCU calls are allowed on the UCS DMS-250. This is a YES or NO field. New Calls will not be processed for the SCU when this value is set to "N" and will be sent to the new PSNF treatment, effectively turning off PSN calls.

PSN_DROP_AGENTS_SCU_SHELF_RESET

Specifies whether to release the Agents on Shelf Resets at the SCU. Its possible values are Yes and No. A Shelf Reset occurs when a Reset-Switch Primitive is sent with a Non-NIL IP Address and Non-NIL Port. A Shelf Reset is equivalent to all PSN agents serviced by the given shelf in an SCU, being taken down.

PSN_DROP_AGENTS_SCU_SRVC_RESET

Specifies whether to release the Agents on Service Resets at the SCU. Its possible values are YES and NO. A Service Reset occurs when a Reset Switch primitive is sent with a Non-NIL IP Address and NIL Port. A Service Reset is equivalent to all PSN Agents serviced by a SCU Service (feature) being taken down.

PSN_DROP_AGENTS_SCU_SYS_RESET

Specifies whether to release the Agents on System Resets at the SCU. Its possible values are YES and NO. A System Reset occurs when a Reset Switch primitive is sent with a NIL IP Address and NIL Port in the IP-Address-to-Reset optional parameters or a Reset Switch primitive is sent with NO IP-Address-to-Reset optional parameters. A System Reset is equivalent to all PSN Agents serviced by the given PSN being taken down.

Note: The IP-Address-to-Reset optional parameters are completely different than the Return-IP-Address Mandatory parameter which is used by the PSN to know where to send the Instruction-Completed event notification.

PSN_INIT_SCU_POLLING

Specifies whether to initiate initial contact polling. Its possible values are YES and NO.

PSN_EVENT_TIMER

Specifies the amount of time the UCS DMS-250 waits for a reply from the SCU after a NEW CALL event has been sent.

PSN_FLOW_CTRL_MESSAGING

Specifies the SCU Initiated Flow Control and the messaging related to PSN Initiated Flow Control. If the value of this office parameter is “Y”, then all Flow Control primitives received from the SCU are processed. If the value of this office parameter is “N”, then all Flow Control primitives received from the SCU are ignored. If the value of the this office parm is changed from “Y” to “N” while the flow control is effective, the control is deactivated, and a PSN106 log is generated with a text reason “Flow Control Deactivated”, at the PSN.

PSN_INTER_POLL_TIME

Specifies the time that the PSN waits between two successive polling cycles. Its range is from 0 to 15 minutes.

If the value is higher than optimum, the time delay between consecutive polling cycles will be long. SCU controlled calls are not serviced during this time.

PSN_HEARTBEAT_WAIT_TIME

Specifies the time that the PSN waits on a heartbeat message from the SCU Arbitrator before it initiates the initial contact polling to get a new Arbitrator’s address. Its range is from 1 to 30 seconds. This timer value should be higher than the time interval of the heartbeat messages sent by the SCU.

PSN_MAX_MEMBER_ADVANCE

PSN_MAX_MEMBER_ADVANCE has a range from 0 to 3, inclusive. It has a default value of 3. Member advancing occurs after a **connect** primitive has seized an idle trunk and then the seizure fails a short time later due to difficulties on the far end of the trunk. The PSN then selects another idle trunk and sends another Route selected event notification until **PSN_MAX_MEMBER_ADVANCE** member advances have occurred.

PSN_PERFORM_NEWCALL_DIGCOL

PSN_PERFORM_NEWCALL_DIGCOL specifies whether digit collection occurs on new originations (on New-Call Events). If the SCU performs digit collection with an offboard media device, then the allocation of UTRs and the extra digit collection processing is not required. Therefore, digit collection on all New-Calls can be turned OFF by setting this office parameter to FALSE (N).

PSN_PRIMITIVE_NUM_EXT_BLOCKS

PSN_PRIMITIVE_NUM_EXT_BLOCKS has a range from 0 to 32,767, inclusive, with a default value of 2,000. The PSN primitive extension block is retained for the duration of the life of an agent in a Service Node call. A Service Node call can consist of one or more agents. The new PSN Primitive extension block is necessary in order to store the parameters of the primitives received from the SCU.

PSN_SCRATCHPAD_NUM_EXT_BLOCKS

PSN_SCRATCHPAD_NUM_EXT_BLOCKS has a range from 0 to 32,767, inclusive, with a default value of 2,000. The PSN Scratchpad Extension Block is retained for the duration of the life of an agent in a Service Node call. The new PSN Scratchpad extension block is necessary in order to store digits during digit collection, outpulsing streams, and so on.

PSN_SPT_LOGS_ON

PSN_SPT_LOGS_ON is boolean (Y or N). It has a default value of Y. When this office parameter is set to Y, all primitives/events to or from the PSN are logged with PSN or PSNE logs. When the office parameter is sent to N, no PSNP or PSNE logs are generated.

Use of existing extension blocks for PSN

In addition to the new extension blocks mentioned in the previous section, PSN calls also use existing UCS DMS-250 defined extension blocks. The following section covers these details.

Feature extension blocks

There are two feature extension blocks that are used to process and manipulate data for a Service Node call: the feature control block and the feature data block.

The number of feature control block and feature data block allocated for each Service Node call depends on the number of agents in that call. For example, if there are three agents in a Service Node call, then there will be three feature control blocks and three feature data blocks allocated for that call.

The existing office parameter `NO_OF_FTR_CONTROL_BLKs` in table OFCENG determines the maximum number of feature control blocks available on the switch. The size of a feature control block is 122 words. Therefore, total storage for feature control blocks equals:

Storage required for feature control blocks =
`NO_OF_FTR_CONTROL_BLKs * FTR_CONTROL_BLK_SIZE`

The maximum storage required for feature control blocks is
 $32,767 * 122 = 3,997,574$ words

The existing office parameter `NO_OF_LARGE_FTR_DATA_BLKs` in table OFCENG determines the maximum number of feature data blocks available on the switch. The size of the feature data block used is 37 words. Therefore, the total storage for feature data blocks equals:

Storage required for feature data blocks =
`NO_OF_LARGE_FTR_DATA_BLKs * LARGE_FTR_DATA_BLK_SIZE`

The maximum storage required for feature data blocks is
 $32,767 * 37 = 1,212,379$ words

In order to calculate the number of feature translation, feature control blocks, and feature data extension blocks needed on a particular switch, the number of Service Node enabled calls must be estimated.

Tables

The following new tables have been added for PSN:

- The PSNMSGIX table maps the message ID into a Tone CLLI (through Table TONES) or an Announcement CLLI (through Table ANNS).
- PSNROUTE: every trunk that is sent as the destination trunk in the Connect primitive must be datafilled in this table. The key to this table is the external trunk group number of the destination trunk.

- **SCUADDR:** stores address information of the initial contacts and retry and delay time information for the SCU. This information is used to make an initial contact with the SCU when the PSN comes into service. PSN attempts initial contact after reboot, a cold or reload restart, and in the event of Heartbeat Failure from the SCU.

PSNMSGIX

The PSNMSGIX table maps a MESSAGE ID received from the SCU for SCU calls to an ANNSCLLI/TONECLLI in table ANNS/TONES.

The key to table PSNMSGIX is SCUINDEX, which is received from the SCU. The next field in the tuple is MSGTYPE with a value of TONE (for tones) or ANNS (for announcement). The CLLI can be datafilled in either the table TONES or the table ANNS.

This allows the UCS DMS-250 switch to play either tones or announcements as specified by the SCU.

Datafill sequence

Table TONES or ANNS must be datafilled before table PSNMSGIX.

If the CLLI name in the CLLI field is not datafilled in table ANNS or table TONES, the tuple can not be added to table PSNMSGIX. Also, the following error message is displayed:

```
ERROR: THE "CLLI" FIELD MUST FIRST BE DATAFILLED IN TABLE  
ANNS/TONES.
```

A CLLI name must be deleted from a table that refers to the tables ANNS or TONES before it can be deleted from tables ANNS or TONES. This functionality exists today. The following error message is displayed:

```
ERROR: TUPLE REFERRED TO BY ANOTHER TABLE.
```

The same logic is implemented for table PSNMSGIX and the same error message is displayed if an attempt is made to delete a tuple from table ANNS/TONES before deleting from table PSNMSGIX.

Table sizing

The minimum table size is 0.

The maximum table size is 255.

Example Datafill

SCUINDEX	MSGTYPE	CLLI
1	TONE	BUSY
2	TONE	CONF
3	TONE	CCAP
4	ANNS	BONGTONE
8	ANNS	LCNVANN
9	ANNS	LECVANN

PSNROUTE

Table PSNROUTE maps an external trunk number received from the SCU for SCU calls to a trunk CLLINAME.

The key to table PSNROUTE is EXTNUM, which is received from the SCU. The next field in the tuple is CLLI against which a CLLI is datafillable.

Datafill sequence

Tables CLLI, TRKGRP, TRKSGRP, and TRKMEM (at least one member) must be datafilled before table PSNROUTE.

If the information pertaining to the CLLI name in the CLLI field is not datafilled in any of the above tables, the tuple can not be added to table PSNROUTE. Furthermore, if the CLLI being datafilled is not a valid or a supported terminating agent in PSN, the tuple cannot be added to the table PSNROUTE.

Table sizing

The minimum table size is 0.

The maximum table size is 9999

Example

EXTNUM	CLLI
220	DAL220TWDTGS
8191	EAN670TWMFWK

SCUADDR

Table SCUADDR stores the address information of the initial contacts and retry and delay time information for the SCU. This information is used to make an initial contact with the SCU when the PSN comes into service. PSN attempts initial contact after reboot, a cold or reload restart, and in the event of `Heartbeat_Failure` from the SCU.

Description

- Specifies the index into the table. Range { 1 TO 3 }
- `IP_ADDR` – Specifies the IP address of an initial point of contact at the SCU. Range TABLE[0 TO 3] OF BYTE
- `PORT_NUM` – Specifies the UDP port number of an initial point of contact at the SCU. Range { 1 TO 65535 }
- `RETRY_TIME` – Specifies the time that the PSN waits for a response to an In Service event sent to an initial point of contact at the SCU before a new In Service event is sent to that location. Range { 1 TO 60 } in seconds
- `MAX_RETRY` – Specifies the maximum number of times an initial point of contact can be queried to get a new arbitrator's address at the SCU before a new location is tried. Range { 0 TO 2 }

Datafill sequence

Table SCUADDR is datafilled independently of any other existing tables. This table should be datafilled prior to setting office parameter, `PSN_INIT_SCU_POLLING`, to true.

Table sizing

The maximum table size is 3 tuples.

Example

INDEX	IP_ADDR	PORT_NUM	RETRY_TIME	MAX_RETRY
1	47.96.192.58	300	5	1
2	47.122.64.210	555	3	0
3	47.122.64.212	707	7	2

Datafill changes to existing tables for PSN

Datafill changes may be required for the following existing tables on PSN.

- IPNETWRK: stores all Internet specific information about the Internet network and the DMS network
- IPROUTER : stores the Internet specific information of each EIU
- IPTHRON : stores the Internet Protocol (IP) THROtting Numbers
- LIUINV : maintains hardware information about the EIU
- MSCDINV : define where the FLIS connects to the MS
- MSILINV : define the inter-message-switch links if necessary
- SUSHELF : datafill the F-Bus components of the FLIS

PSN billing

This chapter contains the detailed description of the PSN Billing. PSN Billing is mostly customer-specific, as each customer may have a different billing scheme. For more information, refer to the *UCS DMS-250 Billing Records Application Guide*.

Billing summary

After a port becomes involved in an SCU call, there are two ways in which the billing records for a port may be updated through the SCU: directly or indirectly.

- Billing Records Updated by the SCU (Directly)
 - In this method, the SCU sends the *Billing_Info* parameter with the billing information to be used to update the port's billing records. The PSN decodes this parameter and updates the billing record with the appropriate data.
 - This parameter may be sent by the SCU in one of two ways:
 - The first way is sent in the **Set_Billing_Record** primitive. This primitive can be sent at any time in the call for the port, as long as the port is involved in the SCU call.
 - The second way is as a part of any one of the other “call control” primitives in the *Billing_Info* parameter. The billing record for the port is updated at the end of the successful execution of that primitive if the received parameter is free of errors.
- Billing Records Updated by the SCU (Indirectly)
 - In this method, the received primitive is used to populate the billing records. For example, the **Disconnect** primitive may be used to implicitly update “disconnect” related billing information, such as disconnect time.

PSN billing details

The details on how and what billing information is captured depends entirely upon the customer using PSN. The customer may decide to capture some information using in the CDRs (Call Detail Records) or decide to do the billing entirely on the SCU.

UCS SPI introduction

The remaining chapters contain information on the Service Programming Interface (SPI) between the Programmable Service Node (PSN) and Service Control Unit (SCU). This chapter includes a high level description of SPI backwards compatibility for both platforms.

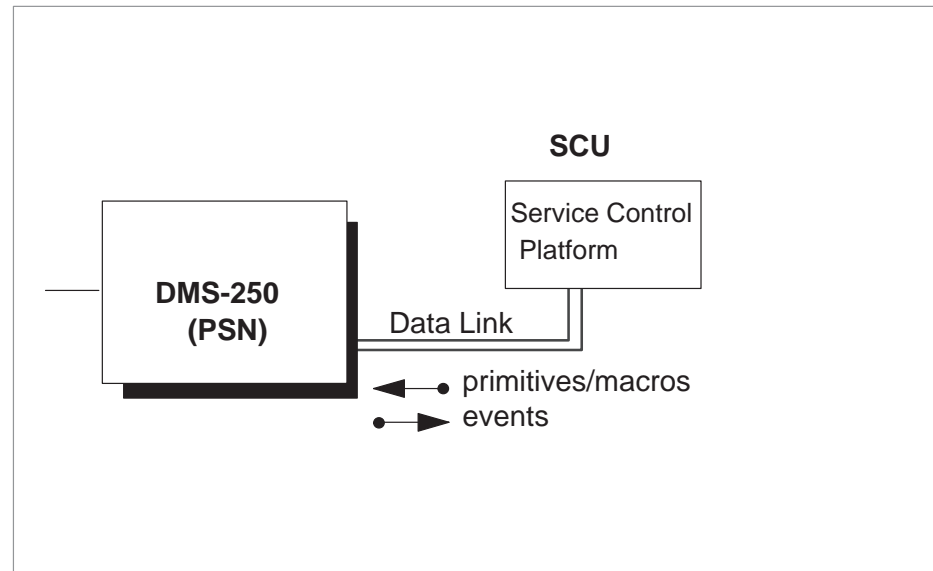
Feature synopsis

A Peer-to-Peer Application protocol (called SPI) has been created for the Programming Service Architecture (PSA). The protocol defines the primitives that are provided to allow the SCU to control the calls on the PSN and the event notifications that are reported to the SCU, from the PSN. The protocol also defines the parameters that go along with the primitives and event notifications.

Networking overview

Figure 12-1 shows the PSN switching matrix configuration.

Figure 12-1
PSN switching matrix configuration



An enhanced switching matrix is defined at the PSN, and the PSN is connected to the SCU using an ethernet data link. The data link is used to send call control primitives and macros from the SCU to the PSN to provide SCU services to calls on PSN. Events are sent from PSN to SCU to notify the SCU of the completion of execution of the primitives and the occurrence of external events that affect the call. Refer to the previous figure.

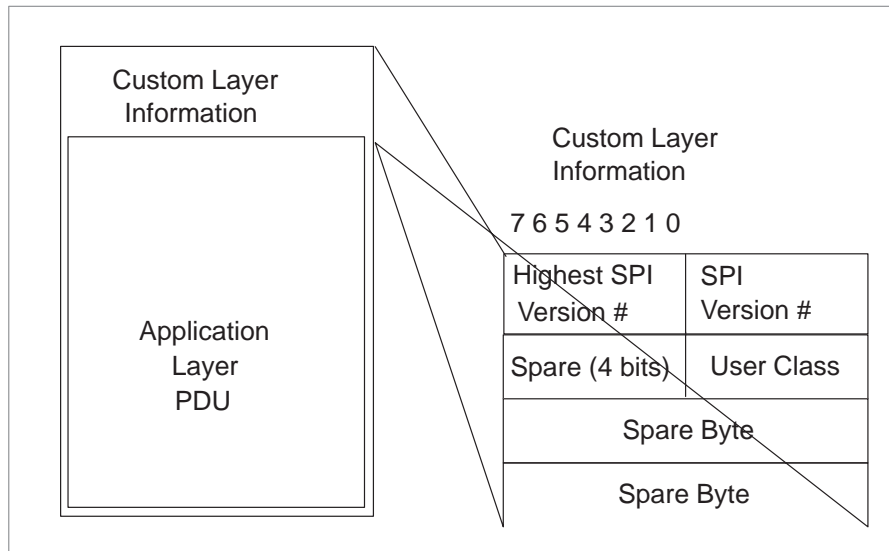
Custom layer protocol

The Custom Layer provides a connectionless transfer of application layer's PDU (Protocol Data Unit) across the network. Since it is a connectionless protocol, messages are not guaranteed to be delivered, nor are those that are delivered, guaranteed to be passed to the application layer in the order in which they were sent by the peer layer at the SCU. The application layer must be designed to accommodate and recover from these situations.

The custom layer implements a Datacom Client/Server model to communicate data across the network. The Datacom Client/Server acts as an interface between the application layer and the underlying transport layer. The Datacom Client transfers the application layer's PDU to the underlying transport layer and the Datacom Server receives the application layer's PDU from the transport layer and transfers it to the application layer.

Each application layer's PDU is wrapped in a Custom Layer's envelope before being sent to the SCU. The PDU is unwrapped by the peer layer at the SCU. Figure 12-2 shows the customer layer header.

Figure 12-2
Custom layer header



The SPI Version Number (four bits) specifies the version of the custom protocol being used.

The Highest SPI Version Number (four bits) indicates the highest supported SPI version on the PSN.

The user class field (four bits) is used to differentiate between the different users of messages going across this layer. Only four types of messages are identified by this layer, which are encoded as follows:

Unused	0000
Admin Class	0001
CallP Class	0010
Audit Class	0011
Flow Control (FCMSG)	0100
Unused	0101–1111

SPI version 1 for UCS

Up to three versions of each primitive and event are supported at any given time. The following table describes where detailed descriptions for SPI Version 1 can be found in this document:

Chapter	Description
PSN parameters Version 1	This chapter contains a description of the UCS PSN parameters used in PSN primitives and/or PSN event notifications.
PSN LOPER messages and parameters Version 1	This chapter contains a description of the Low Overhead Protocol Encoding Rule (LOPER) for all of the UCS PSN parameters.
PRI messages	This chapter contains a description of the special requirements for encoding and decoding PRI messages. Currently, there is no plan to have an SS7 section since the coding standard used can be found in TR444.

SPI version 2 for UCS

Up to three versions of each primitive and event are supported at any given time. The following table describes where detailed descriptions for SPI Version 2 can be found in this document.

Chapter	Description
PSN parameters Version 2	This chapter contains a description of the UCS PSN parameters used in PSN primitives and/or PSN event notifications.
PSN LOPER messages and parameters Version 2	This chapter contains a description of the Low Overhead Protocol Encoding Rule (LOPER) for all of the UCS PSN parameters.
PRI messages	This chapter contains a description of the special requirements for encoding and decoding PRI messages. Currently, there is no plan to have an SS7 section since the coding standard used can be found in TR444.

The following table presents a summary of the differences between SPI Version 2 and the previous SPI Version:

Functionality Change	Description
New Agent-Data Event	<p>The Agent Data event is added, which is sent in response to Set-IP-Address primitives and when certain Trunk datafill changes on the PSN. This event allows the SCU to keep accurate track of what Trunk Agencies are available for termination dynamically.</p> <p>A “Send-Agent-Data” BBOL field is added to the Set-IP-Address to not have an Agent-Data sent in response to the Set-IP-Address primitive.</p>
Additions to New-Call	<p>Four fields are added to the New-Call. These fields mimic the fields in the Agent-Data (which is essentially based on Party B’s) to allow for the SCU to have the same info on Party A’s.</p> <ul style="list-style-type: none"> • Mandatory Signalling-Type • Mandatory Agent-Type • Optional ISUP-Index • Optional COT Req
Additions to Control-Info Parameter	<p>Three BOOL fields are added to allow for greater Event control from the SCU:</p> <ul style="list-style-type: none"> • Send Offhook Events • Send Onhook Events • Send Termination Signalling Events (ACM, Digits-Outpulsed, and so forth)
New Stop-Heartbeat Event	<p>A new event is added so that when a PSN receives a Set IP Address which changes the Arbitrator IP Address, the PSN sends a Stop-Heartbeat event to the old IP Address so that the Arbitrator does not continue to send Heartbeats (which will now be logged due to Heartbeats should only come from the currently active Arbitrator).</p>

SPI version 3 for UCS

Up to three versions of each primitive and event are supported at any given time. The following table describes where detailed descriptions for SPI Version 3 can be found in this document.

Chapter	Description
PSN parameters Version 3	This chapter contains a description of the UCS PSN parameters used in PSN primitives and/or PSN event notifications.
PSN LOPER messages and parameters Version 3	This chapter contains a description of the Low Overhead Protocol Encoding Rule (LOPER) for all of the UCS PSN parameters.
PRI messages	This chapter contains a description of the special requirements for encoding and decoding PRI messages. Currently, there is no plan to have an SS7 section since the coding standard used can be found in TR444.

The following table presents a summary of the differences between SPI Version 3 and the previous SPI Version:

Functionality Change	Description
New-Call Digits-Collected Changes	<p>The New-Call Digits-Collected Changes parameter is enhanced to be more consistent and complete:</p> <ul style="list-style-type: none"> • the STS and CRID New-Call Parameters were folded into the Digits-Collected Parameter since they are simply digit-registers anyway • in addition to the new STS and CRID Digits-Types added, OPart and TPart Digits-Types are also added • The CPC and CRID are no longer copied from the IAM and into the Digits-Collected parms. The Digits-Collected parameters are solely populated by data in the CCB and OCCB while the IAM and SETUP are sent untouched (and uncopied) • The population of the Digits-Collected parameters from the CCB and OCCB were cleaned up and made much straight forward and dependant on Standard CallP logic. Therefore, the digits sent may now sometimes be more, different, or less than in the previous SPI Version
PSN-Member-Advance Office Parameter	<p>This new office parameter is added to allow for Member-Advancing to be turned off. This functionality often wreaks havoc and is not only useful (to be able to turn down or off member-advancing) in the field, but also in the lab.</p>
BBF STR and CCB Error Handling Cleanup	<p>When BBF (using STRs) fails or there are no available CCBs, the Error-Handling has been cleaned up by adding some Error-Causes, better logs and new OMs are implemented for better tracking of the real failure. Previous functionality was misleading.</p>
Support for SS7 IMT Originations	<p>SS7 IMT Originations are now supported through CAIN, PSN CAIN Datafill, and through PSN New-Call processing.</p>

Functionality Change	Description
Billing Info Parameter changes	<ul style="list-style-type: none"> • Removal of the Port Info Parameter from the Billing-Info Parameter, so any change to the billing on a port is done by sending a primitive with a Port Info parameter set for that particular port. • Addition of one byte to Billing Info. One bit “Target Port” is used as a bool with two values, 0 → Port A, and 1 → Port B. Since Connect and Reconnect primitives target two ports at the same time (that is, port A and port B) there exists the need to specify a targeted port in the Billing Info optional parameters.
Control Info Parameter changes	<ul style="list-style-type: none"> • Removal of the Port Info Parameter from the Control-Info Parameter, so that the only way to effect the call processing on another port is through a primitive that is sent directly to that particular port. • Addition of one bit “Target Port” to Control Info Parameter. One bit is used as a bool with two values, 0 → Port A, and 1 → Port B. Since Connect and Reconnect primitives target two ports at the same time (that is, port A and port B) there exists the need to specify a targeted port in the Control Info optional parameter. • Adding the contents of Siginfo Mask parameter as new fields to Control Info parameter provides functionality to the primitives. Any primitive that contains the Control Info parameter (as an optional parameter) carries information that represents the connection states and allows the SCU to control which of the optional SS7/PRI messages are reported to the SCU in the Signalling Event event notification message. • Removal of Send Termination Signaling Events bool bit from the Control-Info Parameter to avoid duplicate functionality with the ACM/Alert bit provided by Siginfo Mask field added to Control Info Parameter.

Functionality Change	Description
Port Service Info Parameter changes	<ul style="list-style-type: none"> • Removal of the Port Info Parameter from the Port Service Info Parameter, so any change to the IP address and/or a change to the SPI version on a port is done only by sending a Set IP Address primitive to that particular port. • Addition of one bit “Target Port” to Port Service Info Parameter. One bit is used as a bool with two values, 0 -> Port A, and 1 -> Port B. Since Connect and Reconnect primitives target two ports at the same time (that is, port A and port B) there exists the need to specify a targeted port in the Port Service Info optional parameter.
Set Billing Record Primitive changes	<ul style="list-style-type: none"> • Port Info parameter is added as a mandatory parameter.
Set IP Address Primitive changes	<ul style="list-style-type: none"> • Port Info parameter is added as a mandatory parameter.
Transmit Siginfo Primitive changes	<ul style="list-style-type: none"> • Signalling Info parameter is added as a mandatory parameter.
Siginfo Mask Parameter changes	<ul style="list-style-type: none"> • Removal of Siginfo Mask parameter as a PSN parameter in UCS08 to avoid duplicate functionality with Control Info Parameter.

SPI version 4 for UCS

Up to three versions of each primitive and event are supported at any given time. The following table describes where detailed descriptions for SPI Version 3 can be found in this document.

Chapter	Description
UCS PSN parameters version 4	This chapter contains a description of the UCS PSN parameters used in PSN primitives and/or PSN event notifications.
UCS PSN LOPER messages and parameters version 4	This chapter contains a description of the Low Overhead Protocol Encoding Rule (LOPER) for all of the UCS PSN parameters.
PRI messages	This chapter contains a description of the special requirements for encoding and decoding PRI messages. Currently, there is no plan to have an SS7 section, since the coding standard used can be found in TR444.

UCS PSN parameters version 1

This chapter contains a detailed description of the UCS PSN Parameters. These PSN parameters may be a part of a PSN primitive and/or a PSN Event Notification. The use of the parameter depends upon the primitive or event notification in which it appears.

PSN peer-to-peer application protocol

A peer-to-peer application protocol has been created for the PSN platform. The protocol defines the primitives that are provided to allow the SCU to control the calls on the PSN and the event notifications that are reported to the SCU from the PSN. The protocol also defines the parameters that go along with the primitives and event notifications.

The peer-to-peer protocol definition utilizes generic functional references to data rather than specific PSN data requirements. The protocol including the primitive or event notification definitions, parameter definitions and message flow (with service implementation examples) is described later in this document.

PSN parameter definitions

The SCU sends instructions via primitives to the PSN to control the service call, PSN notifies the SCU of the events that occur at the switch. The instructions and event notification messages contain one or more parameters with the appropriate information.

Each parameter has one or more bytes containing the parameter information. The division of parameter contents into fields, and the encoding of these fields, is given in detail in this chapter.

The primitives and/or the event notification messages which the parameter may be a part of was described earlier.

Tables 13-1 and 13-2 lists primitives, event notifications, and the associated parameters. Each parameter is marked with an “M” for mandatory, an “O” for Optional ,or a “-” if the parameter is not associated with the primitive or event notification.

Table 13-1
Parameters for call control primitives (Part 1)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n i f o
Parameters															
<i>AccessType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>BearerCapability</i>	-	-	O	-	-	-	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	O	O	O	O	O	O	O	O	O	O	O	M	O	O
<i>Calltype</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>COntrolInfo</i>	-	O	O	O	O	O	O	O	-	O	O	O	O	O	O
<i>DestinationTrunkgroup</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitCollection</i>	-	M	-	-	-	-	-	-	-	-	M	-	-	-	-
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitstoOutputpulse</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsOutputpulsed</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	B r i d g e	C o l l e c t _ D i g i t s _ & _ R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w _ C a l l _ A c c e p t e d	N e w _ C a l l _ R e j e c t e d	P l a y _ M e s s a g e	P P C D	R e c o n n e c t	S e t _ B i l l i n g _ R e c o r d	S t o p _ M e s s a g e	T r a n s m i t _ S i g n f o
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M
<i>MessageInfo</i>	O	-	-	-	-	-	-	-	-	M	M	-	-	M	-

Table 13-2
Parameters for call control primitives (Part 2)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n i f o
Parameters															
<i>MonitorMask</i>	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	M	M	M	M	M	M	M	M	M	M	M	-	M	M
<i>PortServiceInfo</i>	-	O	O	O	O	O	O	M	-	O	O	O	O	O	O
<i>PortStatus</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ServingTranslationScheme</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>SessionID</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g I n f o
<i>SignalingInfo</i>	-	-	M / O	O	-	-	-	-	-	-	-	-	-	-	O
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O
<i>SwitchID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>Timeofday</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Tables 13-3 and 13-4 lists parameters for event notifications.

Table 13-3
Parameters for event notifications (Part 1)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - H o o k	O n - H o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>AccessType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>BearerCapability</i>	-	-	-	-	M	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>Calltype</i>	-	-	-	-	M	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	O	-	-	-	-	-	-
<i>Controlinfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	M	-	-	-	O	-	-	-	-	-	-

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - H o o k	O n - H o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
<i>DigitstoOutpulse</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	M	-
<i>ErrorCause</i>	-	M	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	O	-	-	-	-	-	-
<i>InstructionID</i>	-	O	M	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	O	M	M	M	M	M	M	M	M	M
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-	-	-

Table 13-4
Parameters for event notifications (Part 2)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - H o o k	O n - H o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	O	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	O	M	M	M	M	M	M	M	M	M
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortStatus</i>	-	O	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - H o o k	O n - H o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
<i>ServingTranslationScheme</i>	-	-	-	-	O	-	-	-	-	-	-
<i>SessionID</i>	M	O	M	M	-	M	M	M	M	M	M
<i>SignalingInfo</i>	-	-	-	-	O	O	O	-	-	M	-
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SwitchID</i>	-	-	-	-	M	-	-	-	-	-	-
<i>Timeofday</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	M

Tables 13-5 and 13-6 list parameters for non-call related events and primitives.

Table 13-5
Parameters for non-call related events and primitives (Part 1)

	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t S w i t c h	S e t - I P - A d d r e s s
Parameters									
<i>AccessType</i>	-	-	-	-	-	-	-	-	-
<i>BearerCapability</i>	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-
<i>Calltype</i>	-	-	-	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-
<i>Controlinfo</i>	-	-	-	-	-	-	-	-	-
<i>DestinationTrunkgroup</i>	-	-	-	-	-	-	-	-	-
<i>DigitCollection</i>	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-
<i>DigitstoOutpulse</i>	-	-	-	-	-	-	-	-	-
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	0	-	-	-	-	-	-	-

	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t S w i t c h	S e t - I P - A d d r e s s
<i>FlowControl Encountered</i>	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	M	-	-	M	M	M	M	M
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-

Table 13-6
Parameters for non-call related events and primitives (Part 2)

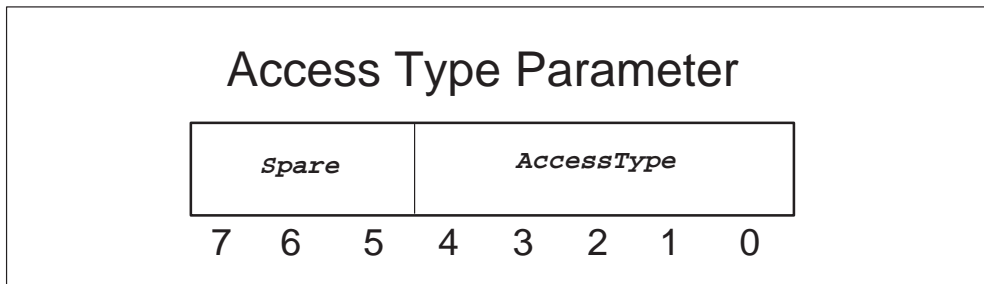
	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t S w i t c h	S e t - I P - A d d r e s s
Parameters									
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	-	-	-	-	M	M	-	-	-
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	M / O	M
<i>PortStatus</i>	-	-	-	-	M	-	-	-	-
<i>ResetReason</i>	-	-	M	-	-	-	-	-	-
<i>ServingTranslationScheme</i>	-	-	-	-	-	-	-	-	-
<i>SessionID</i>	-	-	-	-	O	O	-	-	-
<i>SignalingInfo</i>	-	-	-	-	-	-	-	-	-
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-

	C u r r e n t _ T i m e _ o f _ D a y	F l o w _ C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t _ S t a t u s	Q u e r y _ P o r t	Q u e r y _ T i m e _ o f _ D a y	R e s e t _ S w i t c h	S e t _ I P _ A d d r e s s
<i>SwitchID</i>	-	-	-	-	-	-	-	-	-
<i>Timeofday</i>	M	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-

Note: There is one mandatory parameter called the Return IP Address parameter which is used to send the Instruction Completed. There are 0 to 20 Optional PSI parameters called the IP Address to Reset Parameters which are used to do the actual “reset” of calls.

AccessType

This information is sent in the `New_Call` event notification and contains the originating trunk information. The only values supported for this parameter are FGD, DAL, and PRI.



Mandatory parameter for a **New_Call** event.

Parameter length: 1 byte

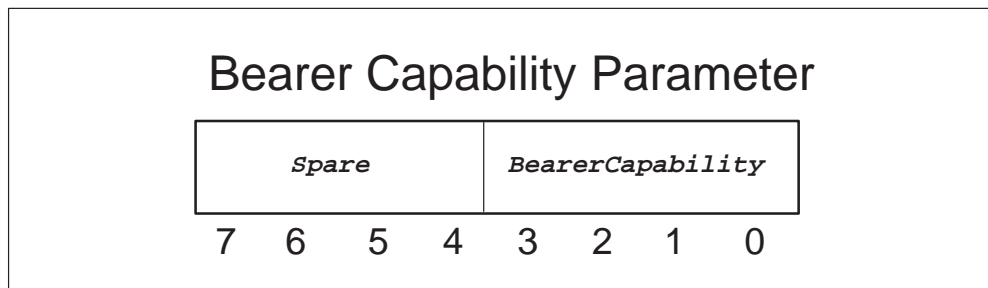
Parameter contents:

- The ACCESS TYPE field consists of three bits, and is the access type of the port, which is encoded as follows:

0 0 0 0: Unused (Spare)
1 0 0 1: PTS FGD
2 0 1 0: SS7 FGD
3 0 1 1: DAL 4-wire
4 1 0 0: DAL 2-wire
5 1 0 1: PRI
1 1 0 to 1 1 1: Unused (Spare)

BearerCapability

This parameter contains the Bearer Capability information.



Optional parameter ID: 1

Parameter length: 1 byte

Parameter contents:

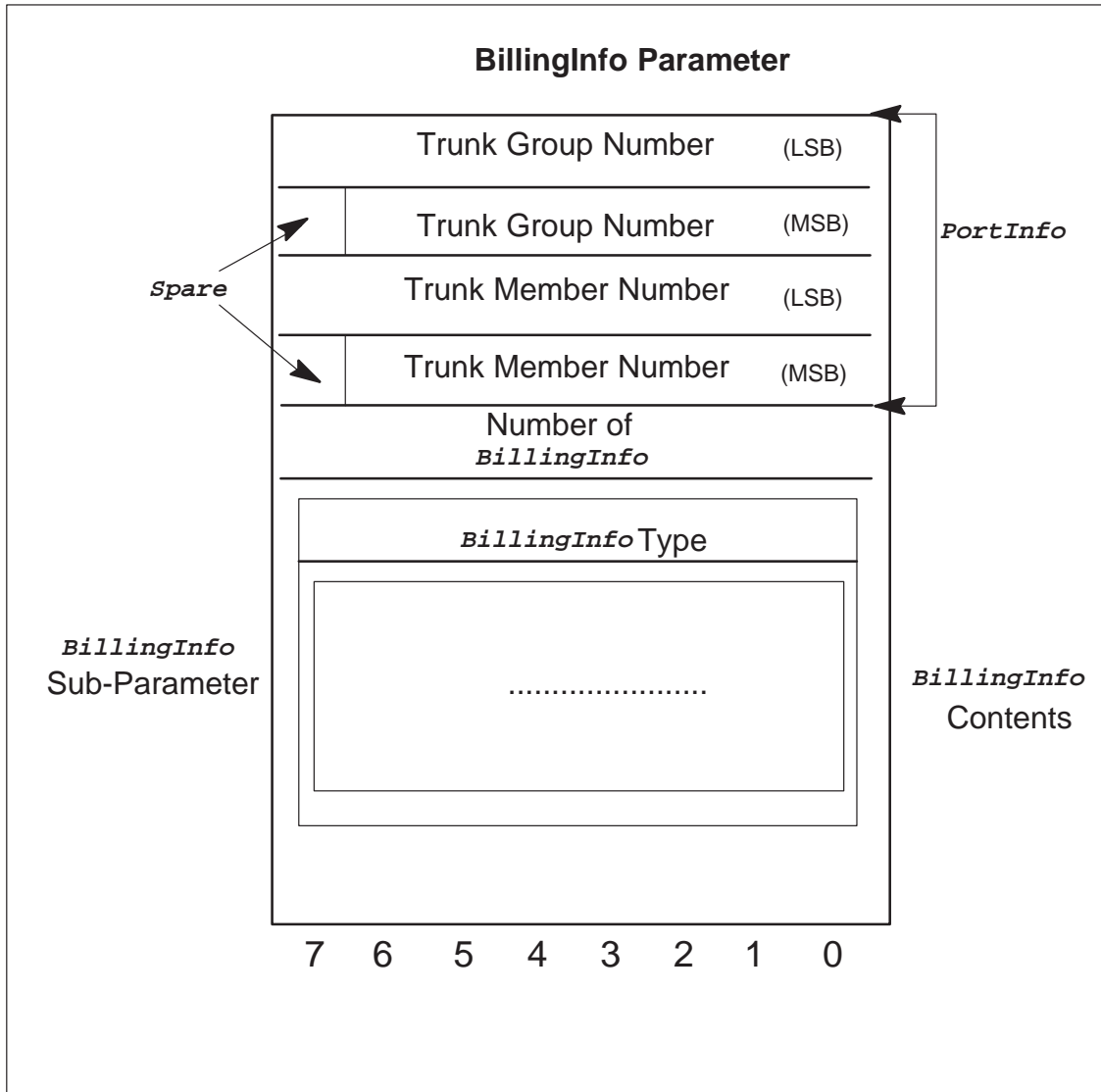
- The BEARER CAPABILITY field consists of four bits, and is the location which indicates the speed and type of information that is to be carried by the call. The values include:

0 0 0 0: Not Used
1 0 0 1: Speech
2 0 0 1 0: 64K data
3 0 0 1 1: 64kX25
4 0 1 0 0: 56k Data
5 0 1 0 1: Data Unit
6 0 1 1 0: 64K Restricted
7 0 1 1 1: 3.1 kHz

- 8 1 0 0 0: 7 kHz
- 9 1 0 0 1: Voice Data
- 10 1 0 1 0: 64K Rate Data
- 1 0 1 1 to 1 1 1 1: Spare Values

BillingInformation

This parameter contains the billing information for the port in the service call. This parameter may be used to update multiple billing record fields for a port.



Optional parameter ID: 2

Parameter length: Variable in size

Parameter contents:

- PORT INFO field consists of four bytes and is the port where the billing records are updated. The external Trunk Group Number and the Trunk Member Number are specified in the Port Info. The range is from 0 to 9,999 for the or both: 0 – 9,999.

A value of 32,767 for Trunk Group Number indicates a NIL_TRUNK_GROUP. The Trunk Group Number of NIL_TRUNK_GROUP in this parameter is considered invalid.

A value of 32,767 for Trunk Member Number indicates a NIL_TRUNK_MEMBER. The Trunk Member Number may be NIL_TRUNK_MEMBER only when the billing information is received for the destination port in the Connect primitive.

Information in the following table illustrates how the combination of Trunk Group Number and Trunk Member Number in the *BillingInfo* parameter is interpreted.

Trunk group number	Trunk Member Number	What is the Billing Info used for
nil_trunk_group	nil_trunk_member	INVALID
Non nil_trunk_group	nil_trunk_member	<i>BillingInfo</i> for destination trunk group in <i>Connect</i> primitive only.
nil_trunk_group	Non nil_trunk_member	INVALID
Non nil_trunk_group	Non nil_trunk_member	<i>BillingInfo</i> for the port specified by the trunk group/member.

- The NUMBER OF BILLING INFO field consists of one byte and is the number of Billing Info sub-parameters to follow. Each Billing Info sub-parameter consists of the Billing Info Type and the Billing Info Contents. The maximum number of Billing Info in Billing Info Parameter is one.

0 00000000: Zero Number of Billing Info

1 00000001: One Number of Billing Info

00000010 to 11111111: Spare Values

- The BILLING INFO TYPE field consists of one byte, which indicates how to interpret the Billing Info Contents. Each application may define its own set of types. The Billing Info Contents are interpreted depending upon the type, which are encoded as follows:

0 0 0 0 0 0 0 0: Not used

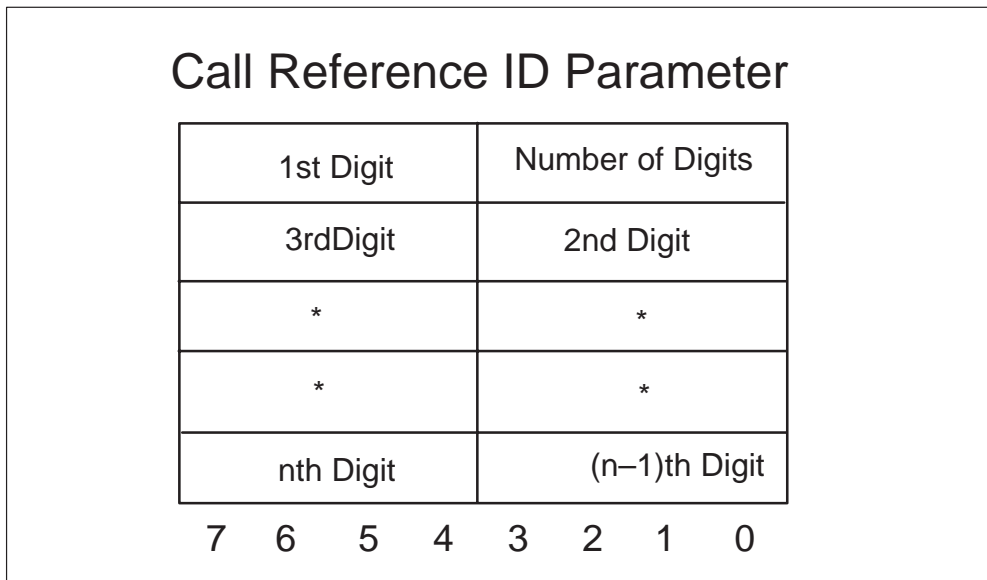
0 0 0 0 0 0 0 1: CRID

0 0 0 0 0 0 1 0 to 1 1 1 1 1 1 1 1: Spare Values

- The BILLING INFO CONTENTS field contains the Billing Info Contents, which actually contain the billing information to place in the billing record.
- This parameter supports the following billing types:
 - Call Reference ID (CRID)

CallReferenceID (CRID)

This parameter is used for the transport of the call reference identifier. The CRID is encoded in a variable length digits field with the digits encoded in TBCD format. The valid range of digits for the CRID parameter is 1 to 9 digits.



Optional parameter ID: 3

Parameter length: Variable in size- maximum of 5 bytes

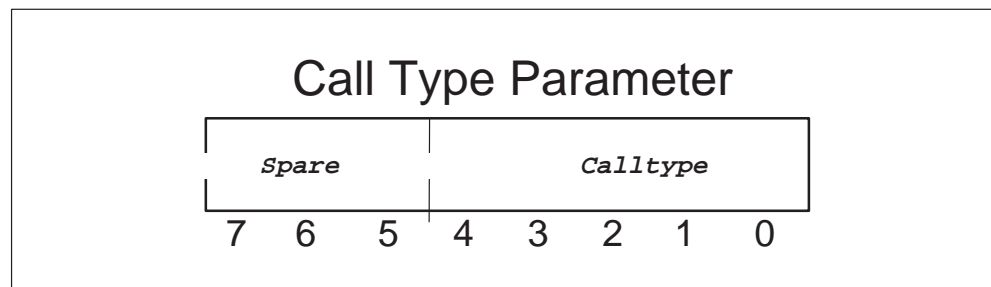
Parameter contents:

- NUMBER OF DIGITS: 4 bits – This field contains the number of CRID digits. The values include: 1 to 9.
- 1ST DIGIT to 9TH DIGIT: Each digit is 4 bits – This field contains the 9 digits in the TBCD format, which are encoded as follows:

0 0 0 0: Filler
 1 0 0 0 1: Digit 1
 2 0 0 1 0: Digit 2
 3 0 0 1 1: Digit 3
 4 0 1 0 0: Digit 4
 5 0 1 0 1: Digit 5
 6 0 1 1 0: Digit 6
 7 0 1 1 1: Digit 7
 8 1 0 0 0: Digit 8
 9 1 0 0 1: Digit 9
 10 1 0 1 0: Digit 0
 11 1 0 1 1: *
 12 1 1 0 0: #
 13 1 1 0 1: D
 14 1 1 1 0: E
 15 1 1 1 1: F

Calltype

This parameter contains the call type for the call when it is determined by the PSN that the call is to be controlled by the SCU.



Mandatory parameter for a New Call event.

Parameter length: 1 byte

Parameter contents:

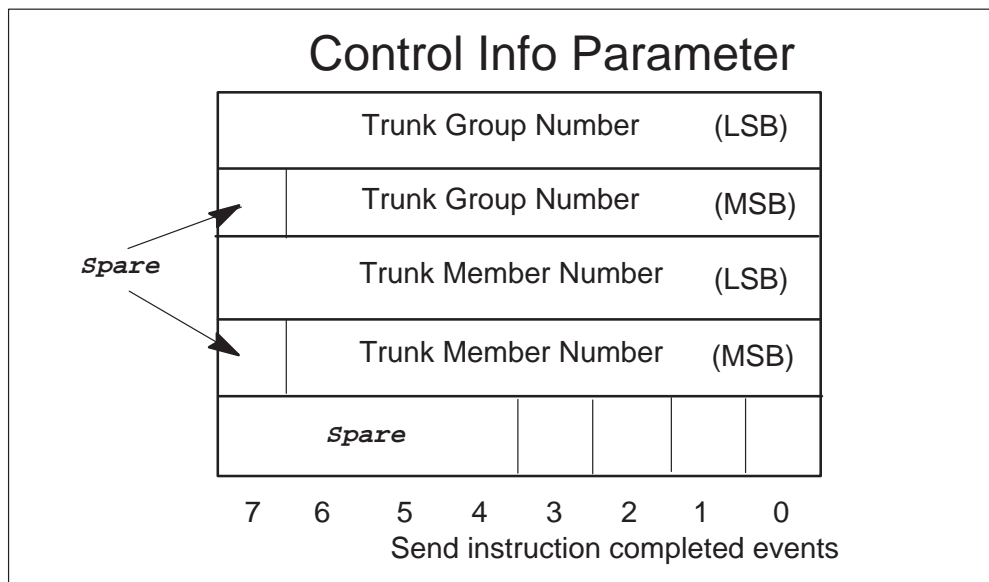
- The CALL TYPE field consists of five bits, which provides information on the type of call being made. The values include:
 0 0 0 0 0: Undetermined *

- 1 0 0 0 0 1: Onnet *
- 2 0 0 0 1 0: Offnet *
- 3 0 0 0 1 1: Public Speed
- 4 0 0 1 0 0: Private Speed
- 5 0 0 1 0 1: Hotline Speed
- 6 0 0 1 1 0: N00*
- 7 0 0 1 1 1: Zero Plus – Onnet
- 8 0 1 0 0 0: Zero Plus – Offnet
- 9 0 1 0 0 1: INTOA *
- 0 1 0 1 0 to 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

Controlinfo

This parameter contains the action to be taken if the primitive is successfully processed.



Optional parameter ID: 4

Parameter length: 5 bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the external Trunk Group Number. The range is from 0 to 9,999. A value of 32,767 indicates a NIL_TRUNK_GROUP.

- The TRUNK MEMBER NUMBER field consists of 15 bits and is the location which contains the Trunk Member Number. The range is from 0 to 9,999. A value of 32,767 indicates a NIL_TRUNK_MEMBER.
 - The SEND INSTRUCTION COMPLETED EVENTS field consists of one bit and is a boolean, which indicates that the agent should or should not send any Instruction Completed events that occur on the agent. The values include:
 - 0: OFF – Do not send any the Instruction Completed events.
 - 1: ON – Send all Instruction Completed events.
- 0 0 0 0 0 0 0: Not used

CRID Parameter

This parameter is used in the New Call event.

CRID Parameter							
Spare				Count			
Spare				Spare			
2nd Digit				1st Digit			
4th Digit				3rd Digit			
6th Digit				5th Digit			
8th Digit				7th Digit			
Spare				9th Digit			
7	6	5	4	3	2	1	0

Optional parameter ID: 30

Parameter length: 7 bytes

Parameter contents:

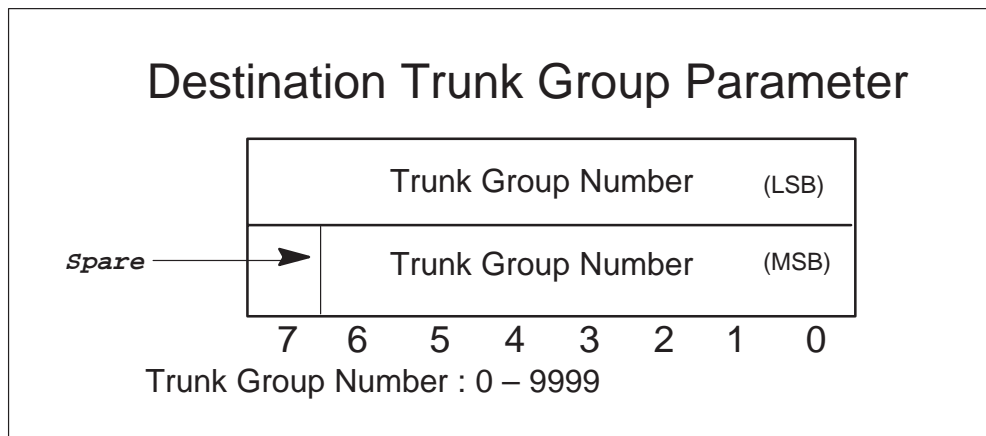
- The COUNT field indicates the number of actual CRID digits stored within the parameter.

- The 1ST DIGIT to NTH DIGIT field consists of four bits for each digit and contains n digits. These DTMF digits are encoded in the TBCD format as follows:

0	0 0 0 0:	Filler
1	0 0 0 1:	Digit 1
2	0 0 1 0:	Digit 2
3	0 0 1 1:	Digit 3
4	0 1 0 0:	Digit 4
5	0 1 0 1:	Digit 5
6	0 1 1 0:	Digit 6
7	0 1 1 1:	Digit 7
8	1 0 0 0:	Digit 8
9	1 0 0 1:	Digit 9
10	1 0 1 0:	Digit 0
11	1 0 1 1:	*
12	1 1 0 0:	#
13	1 1 0 1:	D
14	1 1 1 0:	E
15	1 1 1 1:	F

Destinationtrunkgroup

This parameter contains the external Trunk Group Number of the trunk to route to. The call terminates to an idle member of this trunk group.



Optional parameter ID: 5

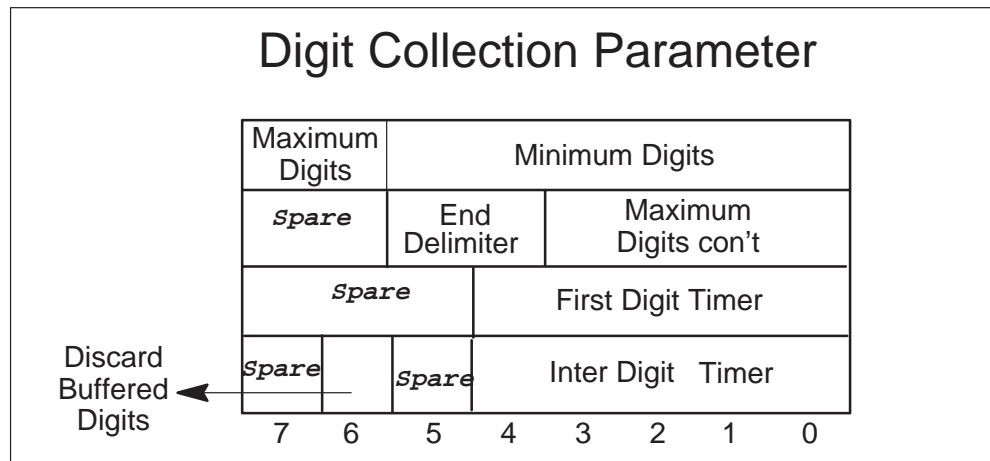
Parameter length: 2 bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the external Trunk Group Number. The valid range is from 0 to 9,999. The NIL_TRUNK_GROUP number is defined as 32,767.

DigitsCollection

This parameter contains information required to collect digits on a specified port.



Optional parameter ID: 6

Parameter length: 4 bytes

Parameter contents:

- The MINIMUM_DIGITS field consists of six bits and is the location which contains the Minimum Number of Digits to collect. This value must be less than or equal to the MAXIMUM_DIGITS field. The values include: 0–45.

0 0 0 0 0 0: (Minimum number of digits to collect = 0)
0 0 0 0 0 1

1 0 1 1 0 1: (Minimum number of digits to collect = 45)
1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (Spare)

- The MAXIMUM_DIGITS field consists of six bits and is the locations which contains the Maximum Number of Digits to collect. The values include: 0–45.

0 0 0 0 0 0: (Maximum number of digits to collect = 0)
0 0 0 0 0 1

1 0 1 1 0 1: (Maximum number of digits to collect = 45)

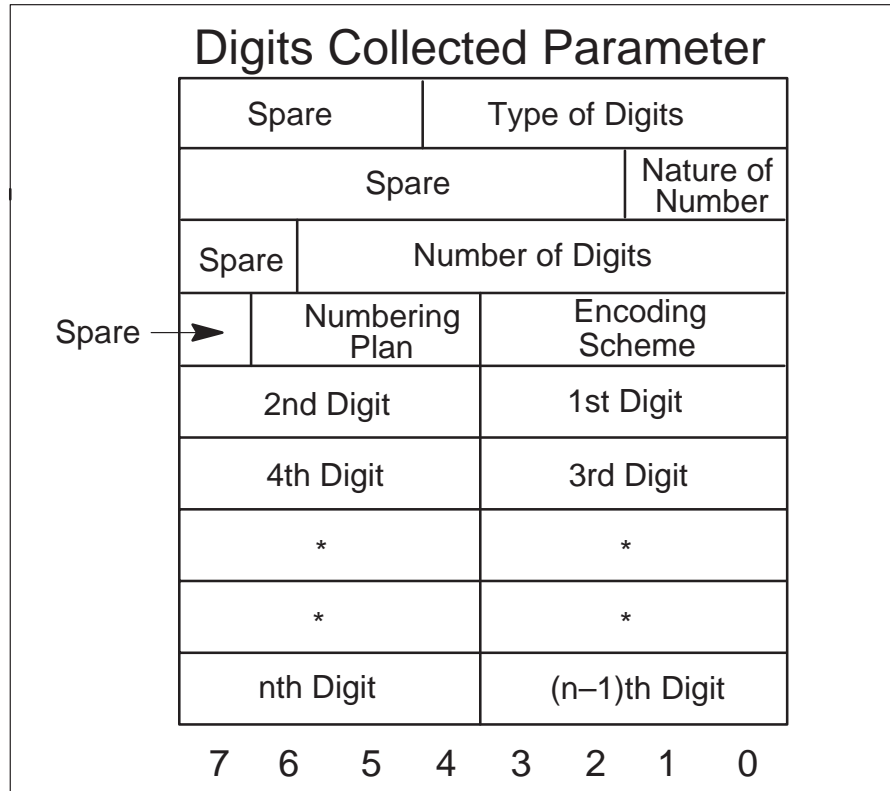
1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (Spare)

- The END_DELIMITER field consists of two bits and is the location which identifies the End of Digits Delimiter; upon detection of the End of Digits Delimiter, digit collection stops and the collected digits are reported. If the delimiter is dialed as the very first digit, then the digit collection stops immediately. The values include:
 - 0 0: no delimiter specified
 - 0 1: *
 - 1 0: #
 - 1 1: * and #
- The FIRST_DIGIT_TIMER field consists of five bits and is the location which specifies the time to wait until the first digit is dialed. The values include : 2–30 seconds.
 - 0 0 0 0 0: Unused (Spare)
 - 0 0 0 0 1: Unused (Spare)
 - 0 0 0 1 0: Timer Value = 2 second
 -
 - 1 1 1 1 0: Timer Value = 30 seconds
 - 1 1 1 1 1: Unused (Spare)
- The INTER_DIGIT_TIMER field consists of five bits and is the location which specifies the Inter Digit Timer value, for example, the time to wait when collecting the second and subsequent digits. The values include: 2–20 seconds.
 - 0 0 0 0 0: Unused (Spare)
 - 0 0 0 0 1: Unused (Spare)
 - 0 0 0 1 0: Timer Value = 2 seconds
 -
 - 1 0 1 0 0: Timer Value = 20 seconds
 - 1 0 1 0 1 to 1 1 1 1 1: Unused (Spare)
- The DISCARD BUFFERED DIGITS field consists of one bit and is a bool, which if true, indicates that the PSN had been buffering digits prior to receiving this parameter. It then discards it and starts digit collection all over again. Refer to Chapter “PSN finite state machine” for details on buffering of digits.

DigitsCollected

This parameter contains the digits collected or received on the port/agent. The digits contained in this parameter are always in the TBCD format. Also,

included is COUNT, which specifies the number of digits included in this parameter.



Optional parameter ID: 7

Parameter length: Variable in size

Parameter contents:

Note: The contents of this parameter (including the Nature of Number, Numbering Plan, and Encoding Scheme fields) have been encoded in accordance with the following documents: TR-NWT-000317 Switching Systems Requirements for Call Control Using ISDNUP, TR-NWT-000394 Switching Systems Requirements for IEC Interconnection using ISDNUP, and TR-NWT-000444 Switching System Requirements Supporting ISDN Access Using the ISDNUP. However, there are some values marked “Private” which are application specific.

- The TYPE OF DIGITS: 5 Bits– This field contains the type of digits encoded as given below. The Type of Digits is known when the collected digits are sent in the **New_Call** event notification. When this parameter is sent in the Digits **Digits_Collected** event notification, a value of “Unknown” is used.

0	0 0 0 0 0	: Unknown
1	0 0 0 0 1	: Called Party Address
2	0 0 0 1 0	: Calling Party Address (ANI)
3	0 0 0 1 1	: Caller Interaction
4	0 0 1 0 0	: Routing Number
5	0 0 1 0 1	: Billing Number
6	0 0 1 1 0	: Destination Number
7	0 0 1 1 1	: Local Access and Transport Area (LATA)
8	0 1 0 0 0	: Carrier Identification
9	0 1 0 0 1	: Referral Number (Private)
10	0 1 0 1 0	: True Billing Number (Private)
11	0 1 0 1 1	: Alternate Preferred Carrier (Private)
12	0 1 1 0 0	: Preferred INC (Private)
13	0 1 1 0 1	: Primary Preferred Carrier (Private)
14	0 1 1 1 0	: Personal Identification Number (PIN)
15	0 1 1 1 1	: Authorization Code (Private)
16	1 0 0 0 0	: TCM (Private)
17	1 0 0 0 1	: Second Alternate Preferred Carrier (Private)
18	1 0 0 1 0	: Business Customer ID (Private)
19	1 0 0 1 1	: Hop-off Office (Private)
20	1 0 1 0 0	: Outpulse Number (Private)
21	1 0 1 0 1	: Originating Station (DN)
22	1 0 1 1 0	: MCCS Card Number
23	1 0 1 1 1	: Account Code Number
24	1 1 0 0 0	: COSOVE Number*
25	1 1 0 0 1	: Generic Digits Number
26	1 1 0 1 0	: Dialed Digits Number
27	1 1 0 1 1	: Facility Code
28	1 1 1 0 0	: Country Code
29	1 1 1 0 1	: STS Digits
30	1 1 1 1 0	: OPart Digits
31	1 1 1 1 1	: TPart Digits

- The NATURE OF NUMBER field consists of two bits and is encoded as follows:

0	0 0	: Not Applicable
1	0 1	: International
2	1 0	: National
3	1 1	: Network Specific

- The NUMBER OF DIGITS: 6 Bits - This field contains the number of digits that are sent in this parameter. This number may be as low as 0 and as high as 45.
- The ENCODING SCHEME field consists of four bits and is the location which contains the scheme used to encode the digits that are sent in this parameter and it is encoded as follows:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Binary Coded Decimal (BCD)
 - 0 0 1 0 to 1 1 0 1: Spare Values
 - 14 1 1 1 0: Telephony Binary Coded Decimal (TBCD)
 - 1 1 1 1: Spare
- The NUMBERING PLAN field consists of three bits and is encoded as follows:
 - 0 0 0 0: Unknown or Not Applicable
 - 1 0 0 1: ISDN Numbering Plan (E.164)
 - 2 0 1 0: Telephony Numbering Plan (E.163)
 - 3 0 1 1: Data Numbering Plan (X.121)
 - 4 1 0 0: Telex Numbering Plan (F.69)
 - 5 1 0 1: Maritime Mobile Numbering Plan (E.120,211)
 - 6 1 1 0: Land Mobile Numbering Plan (E.212, 213)
 - 1 1 1 1: Spare Value
- The 1ST DIGIT to NTH DIGIT field consists of four bits for each digit and is the location which contains n digits. These DTMF digits may be encoded in the BCD format as follows:
 - 0 0 0 0 0: Digit 0
 - 1 0 0 0 1: Digit 1
 - 2 0 0 1 0: Digit 2
 - 3 0 0 1 1: Digit 3
 - 4 0 1 0 0: Digit 4
 - 5 0 1 0 1: Digit 5
 - 6 0 1 1 0: Digit 6
 - 7 0 1 1 1: Digit 7
 - 8 1 0 0 0: Digit 8
 - 9 1 0 0 1: Digit 9
 - 10 1 0 1 0: Filler
 - 11 1 0 1 1: *
 - 12 1 1 0 0: #
 - 13 1 1 0 1: D
 - 14 1 1 1 0: E
 - 15 1 1 1 1: F

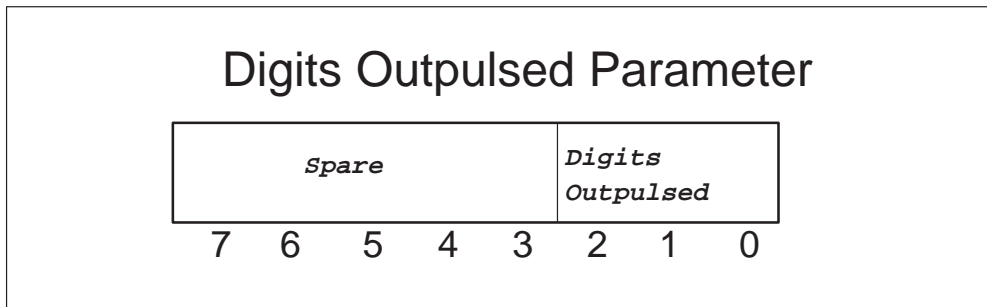
Or the DTMF digits may be encoded in the TBCD format as follows:

- 0 0 0 0 0: Filler

- 1 0 0 0 1: Digit 1
- 2 0 0 1 0: Digit 2
- 3 0 0 1 1: Digit 3
- 4 0 1 0 0: Digit 4
- 5 0 1 0 1: Digit 5
- 6 0 1 1 0: Digit 6
- 7 0 1 1 1: Digit 7
- 8 1 0 0 0: Digit 8
- 9 1 0 0 1: Digit 9
- 10 1 0 1 0: Digit 0
- 11 1 0 1 1: *
- 12 1 1 0 0: #
- 13 1 1 0 1: D
- 14 1 1 1 0: E
- 15 1 1 1 1: F

DigitsOutpulsed

This parameter indicates information on the digits that were outpulsed on a PTS trunk agent. This is especially useful in multi-stage outpulsing. This parameter is included in the Signaling event notification message to let the SCU know that all the digits were outpulsed on the trunk agency.



Optional parameter ID: 8

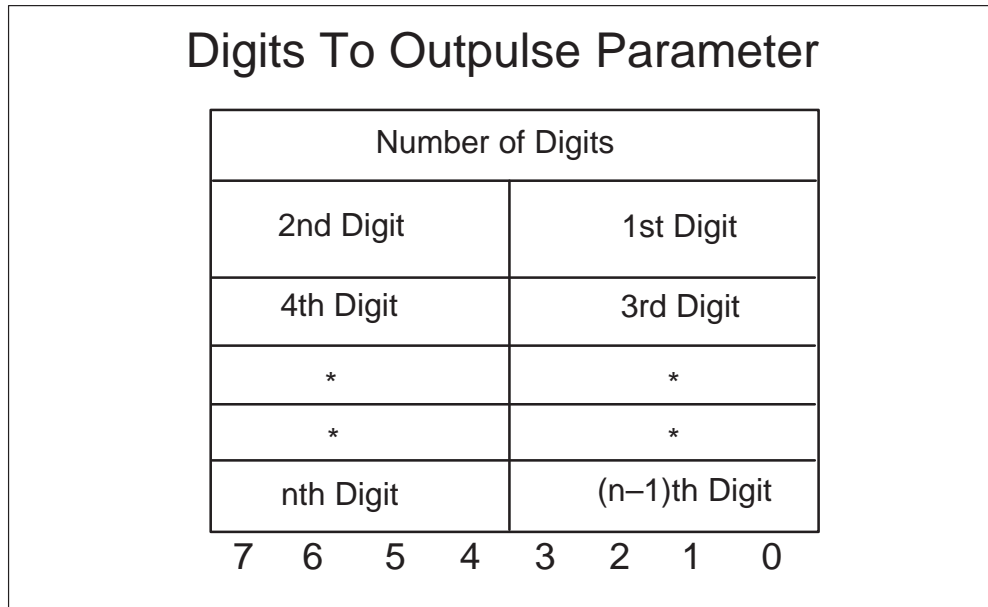
Parameter length: 1 byte

Parameter contents:

- The DIGITS OUTPUTSED field consists of three bits and is encoded as follows:
 - 0 0 0: Unknown
 - 0 0 1: All streams outpulsed
 - 0 1 0 to 1 1 1: Spare

DigitstoOutputpulse

This parameter contains the digits to be outputpulsed on a PTS trunk agent.



Optional parameter ID: 9

Parameter length: Variable in size

Parameter contents:

- The NUMBER OF DIGITS field consists of one byte and is the location which contains the number of digits that are to be outputpulsed on this port. The maximum number of digits that may be outputpulsed is 23. The valid range for this field is 1 – 23.

- The 1ST DIGIT to NTH DIGIT field consists of four bits for each digit and is the location which contains the n digits.

DTMF digits are encoded as follows:

```

0  0 0 0 0: Filler
1  0 0 0 1: Digit 1
2  0 0 1 0: Digit 2
3  0 0 1 1: Digit 3
4  0 1 0 0: Digit 4
5  0 1 0 1: Digit 5
6  0 1 1 0: Digit 6
7  0 1 1 1: Digit 7

```

8 1 0 0 0: Digit 8
9 1 0 0 1: Digit 9
10 1 0 1 0: Digit 0
11 1 0 1 1: *
12 1 1 0 0: #
13 1 1 0 1: D
14 1 1 1 0: E
15 1 1 1 1: F

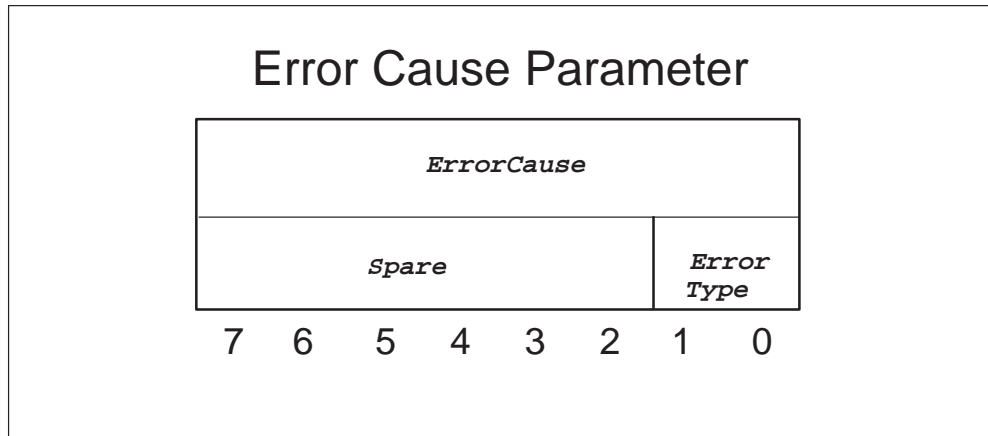
MF digits are encoded as follows:

0 0 0 0 0: Filler
1 0 0 0 1: Digit 1
2 0 0 1 0: Digit 2
3 0 0 1 1: Digit 3
4 0 1 0 0: Digit 4
5 0 1 0 1: Digit 5
6 0 1 1 0: Digit 6
7 0 1 1 1: Digit 7
8 1 0 0 0: Digit 8
9 1 0 0 1: Digit 9
10 1 0 1 0: Digit 0
11 1 0 1 1: KP3 and ST3P
12 1 1 0 0: KPP and STP
13 1 1 0 1: KP and STKP
14 1 1 1 0: KP2 and ST2P
15 1 1 1 1: ST

If multiple “Digits To Outpulse” parameters are contained in a message, then multiple streams of digits are outpulsed on the agent/port—each stream contained in one parameter of type “Digits to Outpulse”.

ErrorCause

This parameter contains the cause of an error that was detected on the PSN.



Optional parameter ID: 10

Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of one byte and is the location which is used to represent the cause of the error. The values include:
 - 0 00000000: Nil Error Cause
 - 1 00000001: Header decode failure
 - 2 00000010: Bad macro tag
 - 3 00000011: Unrecognized primitive
 - 4 00000100: Missing mandatory parameter
 - 5 00000101: Mandatory parameter decode failure
 - 6 00000110: Optional parameter decode failure
 - 7 00000111: Parameter contents out of range
 - 8 00001000: Primitive userclass mismatch
 - 9 00001001: Maximum primitive exceeded
 - 10 00001010: Missing mandatory Signifo parameter
 - 11 00001011: One or more agents in the primitive are not PSN agents
 - 12 00001100: Port not in table PSNROUTE
 - 13 00001101: Agent not supported
 - 14 00001110: Port down due to WARM restart
 - 15 00001111: Primitive invalid for current port state
 - 16 00010000: Unexpected message
 - 17 00010001: STR not available (affects the Monitor primitive)
 - 18 00010010: UTR not available
 - 19 00010011: Conference circuit not available

20 0 0 0 1 0 1 0 0: No IDLE message
21 0 0 0 1 0 1 0 1: Primitive extension block not available
22 0 0 0 1 0 1 1 0: Scratchpad extension block not available
23 0 0 0 1 0 1 1 1: Software Resources unavailable
24 0 0 0 1 1 0 0 0: Message failure
25 0 0 0 1 1 0 0 1: Software error
26 0 0 0 1 1 0 1 0: Not minimum number ports to Bridge
27 0 0 0 1 1 0 1 1: Maximum ports to Bridge exceeded
28 0 0 0 1 1 1 0 0: Bearer capability incompatible
29 0 0 0 1 1 1 0 1: Message index not in table PSNMSGIX
30 0 0 0 1 1 1 1 0: Unsupported signaling type
31 0 0 0 1 1 1 1 1: Duplicate message
32 0 0 1 0 0 0 0 0: Bad agent state
33 0 0 1 0 0 0 0 1: Termination failure
34 0 0 1 0 0 0 1 0: Abnormal Exit
35 0 0 1 0 0 0 1 1: Message not playing
36 0 0 1 0 0 1 0 0: Tone duration unsupported
37 0 0 1 0 0 1 0 1: Prompt failure
38 0 0 1 0 0 1 1 0: Digit collection failure
39 0 0 1 0 0 1 1 1: Q764 protocol problem
40 0 0 1 0 1 0 0 0: Invalid duration gap
41 0 0 1 0 1 0 0 1: Unexpected FC message
42 0 0 1 0 1 0 1 0: Agent not in Table TRKGRP

- The ERROR TYPE field consists of two bits and is the location which contains the type of the error that is detected, and is encoded as follows:
 - 0 0: Non Fatal Error
 - 0 1: Fatal Error

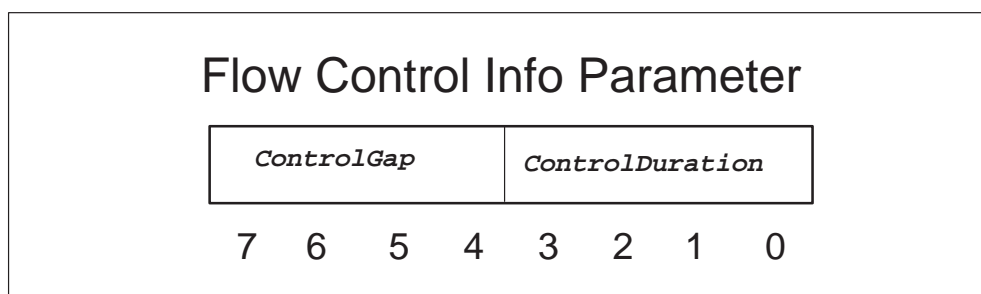
Fatal errors are defined as errors that are so severe that they do not allow normal call processing to proceed on this port.

If the error that is detected is non-fatal, then the PSN does not take any action other than report this error to the SCU. If the error that is detected is fatal, then the PSN reports the error to the SCU, and in addition, takes down the associated port.

FlowControlInfo

This parameter consists of Control Duration, which specifies the maximum amount of time the flow control is effective at the PSN, and Control Gap,

which specifies the maximum rate at which the New Call event notifications may be sent to the SCU while the control is effective.



Optional parameter ID: 11

Parameter length: 1 byte

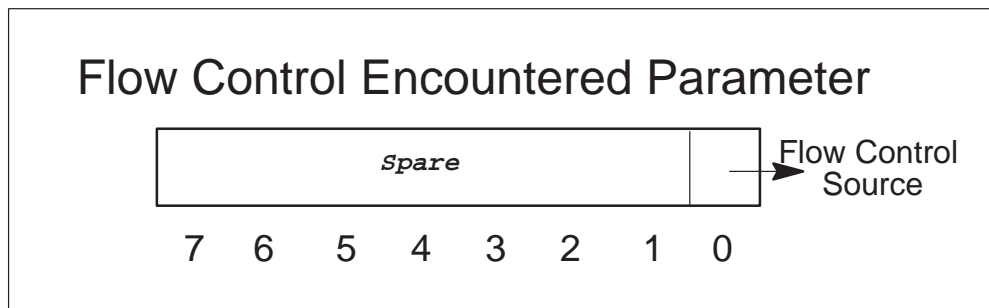
Parameter contents:

- The CONTROL DURATION field consists four bits and is the location which indicates the maximum amount of time that the flow control is effective at the PSN. The Control Duration is encoded as follows:
 - 0 0 0 0: Not Used
 - 1 0 0 1: 1 Second
 - 2 0 0 1 0: 2 Seconds
 - 3 0 0 1 1: 4 Seconds
 - 4 0 1 0 0: 8 Seconds
 - 5 0 1 0 1: 16 Seconds
 - 6 0 1 1 0: 32 Seconds
 - 7 0 1 1 1: 64 Seconds
 - 8 1 0 0 0: 128 Seconds
 - 9 1 0 0 1: 256 Seconds
 - 10 1 0 1 0: 512 Seconds
 - 11 1 0 1 1: 1024 Seconds
 - 12 1 1 0 0: 2048 Seconds
 - 1 1 0 1 to 1 1 1 1: Spare Values
- The CONTROL GAP field consists of four bits and is the location which indicates the maximum rate at which the New Call event notifications may be sent to the SCU while the flow control is effective. The Control Gap is encoded as follows:
 - 0 0 0 0: Remove Gap Control
 - 1 0 0 1: 1 1/10th of a Second
 - 2 0 0 1 0: 3 1/10ths of a Second
 - 3 0 0 1 1: 5 1/10ths of a Second
 - 4 0 1 0 0: 1 Second

- 5 0 1 0 1: 2 Seconds
- 6 0 1 1 0: 5 Seconds
- 7 0 1 1 1: 10 Seconds
- 8 1 0 0 0: 15 Seconds
- 9 1 0 0 1: 30 Seconds
- 10 1 0 1 0: 50 Seconds
- 11 1 0 1 1: 80 Seconds
- 12 1 1 0 0: 120 Seconds
- 13 1 1 0 1: 300 Seconds
- 14 1 1 1 0: 600 Seconds
- 15 1 1 1 1: Stop All Calls

***FlowControlEncountered* parameter**

This parameter is sent in New Call events sent to the SCU while the flow control is active. This parameter informs the SCU that the flow control is active and it identifies the source that initiated the flow control.



Optional parameter ID: 12

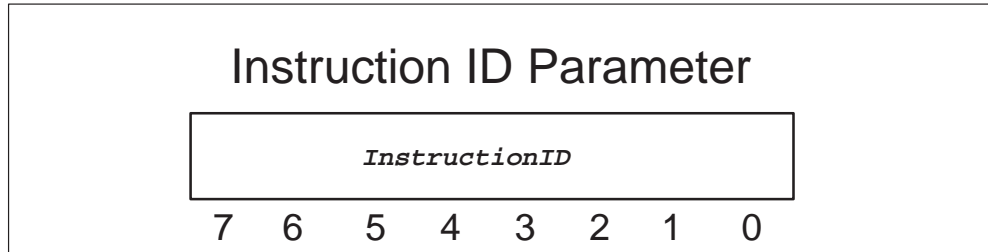
Parameter length: 1 byte

Parameter contents:

- The FLOW CONTROL SOURCE field consists of one bit and is the source that initiated the flow control. It is encoded as follows:
 - 0: Flow control initiated by the SCU
 - 1: Flow control initiated by the PSN

InstructionID

This parameter contains the Identifier of the Primitive that was received from the SCU.



Optional parameter ID: 13

Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION ID field consists of one byte and is the location which is used to represent the primitive instruction. The values include:

```

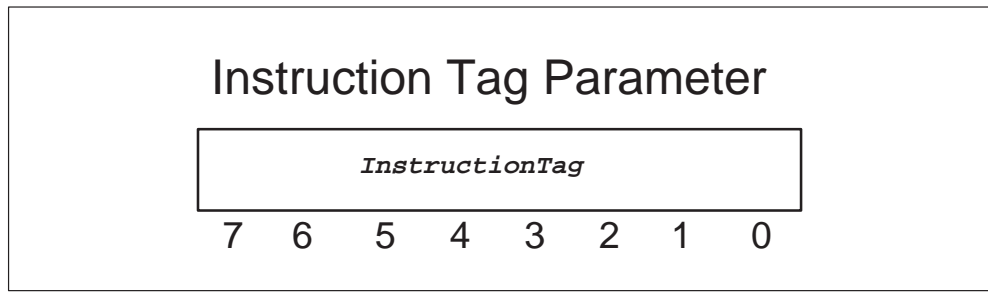
0 00000000: Unknown
1 00000001: Bridge
2 00000010: Collect Digits & Report
3 00000011: Connect
4 00000100: Disconnect
5 00000101: Hold
6 00000110: Monitor
7 00000111: Mute
8 00001000: New Call Accepted
9 00001001: New Call Rejected
10 00001010: Play Message
11 00001011: Play Prompt, Collect Digits & Report
12 00001100: Query Port
13 00001101: Reconnect
14 00001110: Reset Switch
15 00001111: Set Billing Record
16 00010000: Set IP Address
17 00010001: Stop Message
18 00010010: Transmit Siginfo
19 00010011: Heartbeat
20 00010100: Query Time of Day
21 00010101: Error Detected
22 00010110: Port Status
23 00010111: Flow Control
    00011000 to 11111111: Spare Values

```

Instructiontag

This parameter contains the tag associated with the instruction.

- For primitives that are sent from the SCU, the SCU generates an Instruction tag and sends it to the PSN in this parameter. When a response for this primitive is generated by the PSN, the PSN includes this tag in the response to enable the SCU to correlate the response with the primitive request that it had sent earlier.
- For asynchronous event notifications generated by the PSN (for example, On-Hook, Off-Hook and, Signaling Event), the Instruction Tag is hard-coded to #01.
- For instructions that are generated by the PSN that require a reply from the SCU (for example, New Call, Query Port from the Audit Application), the PSN sends a nil Instruction Tag.



Optional parameter ID: 14

Parameter length: 1 byte

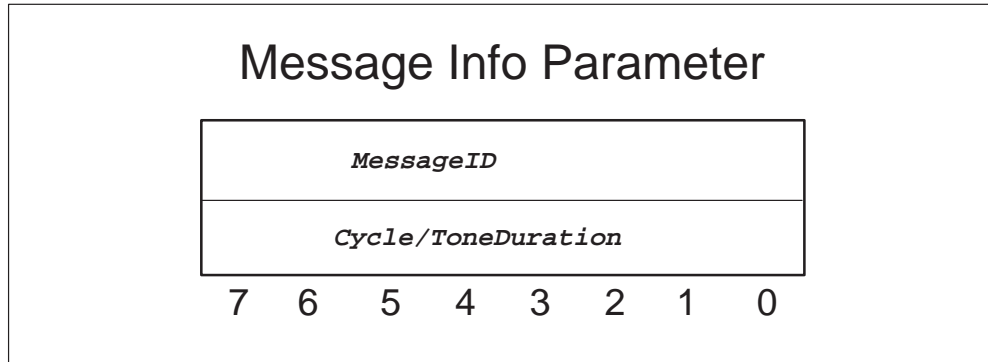
Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which is used to represent the tag of the instruction that was received from or sent to the SCU. The values include:
 - 0 0 0 0 0 0 0 0: Nil Instruction Tag
 - 0 0 0 0 0 0 0 1: Asynchronous Event Notification from the PSN.
 - 0 0 0 0 0 0 0 1 to 1 1 1 1 1 1 1 1: Valid tags for instructions from the SCU.

MessageInfo

This parameter contains the Message ID and the Cycles and Tone Duration information.

The Message ID is used to index into table PSNMSGIX to get either an index into table ANNS (if the message ID corresponds to an announcement) or table TONES (if the message ID corresponds to a tone).



Optional parameter ID: 15

Parameter length: 2 bytes

Parameter contents:

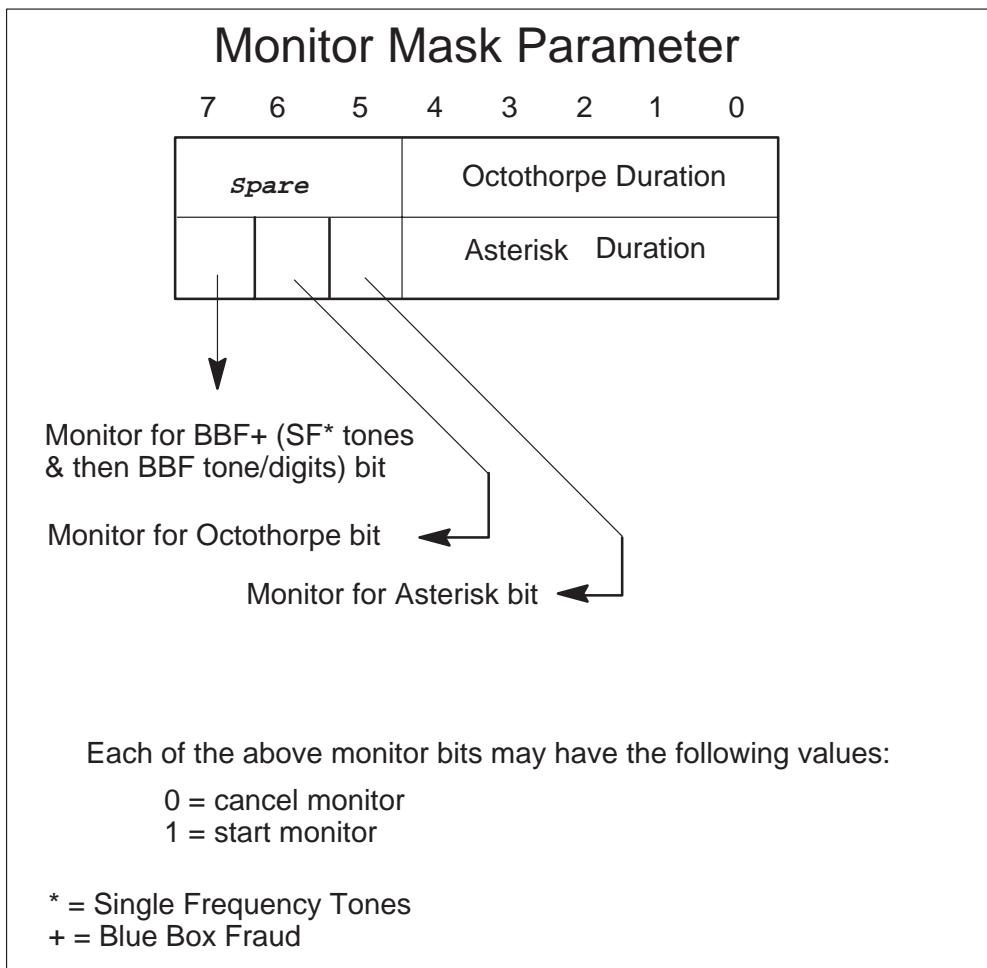
- The MESSAGE ID field consists of one byte. The values include 1 to 255 (a value of 0 is invalid). The value is an index into table PSNMSGIX.
- The CYCLES field consists of one byte: the second byte of the parameter contents is treated as the number of cycles to play the announcement (if the message ID corresponds to an announcement). The values include:
 - 0 0 0 0 0 0 0 0: Play the Announcement indefinitely
 - 0 0 0 0 0 0 0 1: Play the Announcement for 1 cycle
 -
 - 0 0 0 1 1 1 1 0: Play the Announcement for 30 cycles
 - 0 0 0 1 1 1 1 1 to 1 1 1 1 1 1 1 1: Play the Announcement indefinitely
- The TONE DURATION field consists of one byte: the second byte of the parameter contents is treated as the Tone Duration value (if the message ID corresponds to a tone). The values include:
 - 0 0 0 0 0 0 0 0: Play the Tone forever
 - 0 0 0 0 0 0 0 1: Invalid
 - 0 0 0 0 0 0 1 0: Invalid
 - 0 0 0 0 0 0 1 1: Play the Tone for 3 seconds
 - 0 0 0 0 0 1 0 0: Play the Tone for 4 seconds
 -

1 1 1 1 1 1 1 1: Play the Tone for 255 seconds

Note: The values for the cycles or tone duration may be set to any value when the Message Info parameter is sent to the Stop Message primitive. The Stop Message primitive is only concerned with the message ID. For example, to stop the appropriate message on the PSN in the Message Info parameter.

MonitorMask

This parameter is a bitmap of monitor values. Each bit in this bitmap indicates what digit or tone to monitor or not to monitor a for the port. Also, each bit has two values: 0 (indicates cancel the monitor) or 1 (start monitor).



Optional parameter ID: 16

Parameter Size: 2 bytes

Parameter contents:

- The ASTERISK/OCTOTHORPE DURATION field consists of five bits each and is the location which indicates the duration, in 100 milliseconds, for which the Asterisk or Octothorpe must be detected before it is reported as a valid tone or digit to the SCU. The values include 5 to 30.

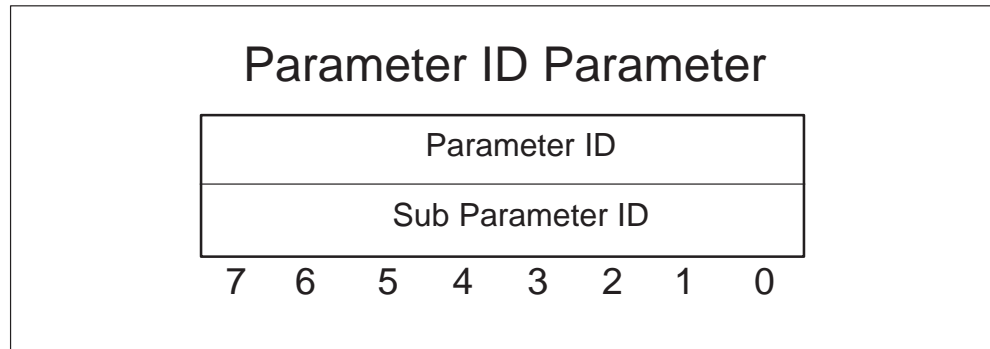
For example, a duration of five for an Asterisk implies that the user has to hold down the Asterisk for 500 msec before it is detected as a valid Asterisk by the PSN and reported to the SCU.

Note: For BBF: the SF Tone Duration, the signaling type (MF/DTMF), the minimum digits to collect before a blue box fraud is declared, and the Partial Dial Timer values are all determined from the BBTKSGRP. Therefore, it is necessary to datafill the “terminating” port (where “terminating” indicates Party B of a Connect or a Reconnect primitive) in table BBTKSGRP.

- The TONE BITMAP field consists of three bits and is a boolean each for BBF, Octothorpe and Asterisk. If the bool is true, then the port is monitored for the appropriate tone or digit. If the bool, is false then the appropriate tone or digit monitoring on that port is canceled.

ParameterID

This parameter contains the Identifier of the Parameter (and the identifier of the parameter if the parameter is composed of sub parameters). It is returned to the SCU in case the decoding of the parameter resulted in an error.



Optional parameter ID: 17

Parameter length: 2 bytes

Parameter contents:

- The PARAMETER ID field consists of : 1 byte – This field is a byte that is used to represent the parameter in error. The values include:

0 00000000: Nil Error Cause
1 00000001: Bearer Capability
2 00000010: Billing Info
3 00000011: Call Reference Identifier
4 00000100: Control Info
5 00000101: Destination Trunk Group
6 00000110: Digit Collection
7 00000111: Digits Collected
8 00001000: Digits Outpulsed
9 00001001: Digits To Outputse
10 00001010: Error Cause
11 00001011: Flow Control Information
12 00001100: Flow Control Encountered
13 00001101: Instruction ID
14 00001110: Instruction Tag
15 00001111: Message Info
16 00010000: Monitor Mask
17 00010001: Parameter ID
18 00010010: Port Count
19 00010011: Port Info
20 00010100: Port Service Information
21 00010101: Port Status
22 00010110: Reset Reason
23 00010111: Session ID
24 00011000: SigInfo Mask
25 00011001: Signaling Info
26 00011010: Switch ID
27 00011011: Time of Day
28 00011100: Tone Detected
 00011101 to 10000001: Spare Values
0 10000010: UCS Point In Call
1 10000011: UCS STS
 10000100 to 11111111: Spare Values

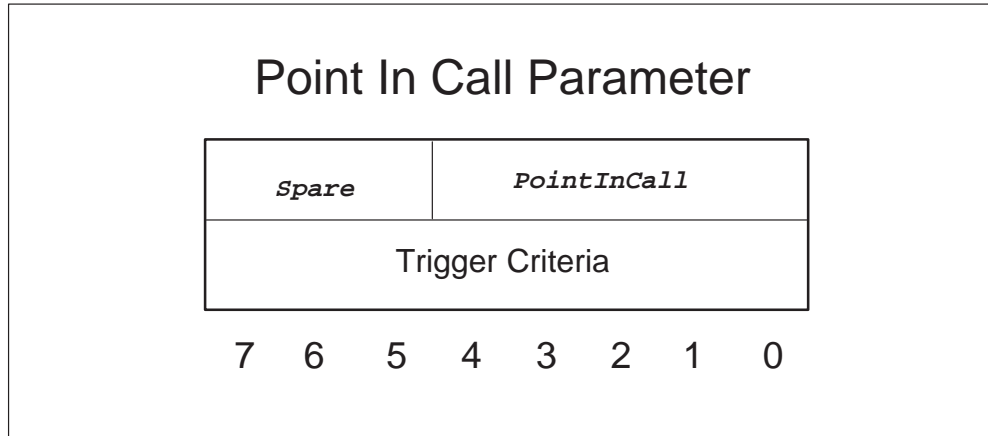
- The SUB PARAMETER ID field consists of one byte and is the location which is used to represent the parameter in error.

The values include (for PTS agents):

0 00000000: Unknown
1 00000001: Message Type
2 00000010: Digits Outpulsed
3 00000011: Digits to Outputse
4 00000100: PTS Off-hook
5 00000101: PTS On-hook
 00000110 to 11111111: Spare

PointInCall

This parameter contains the point in the call when all criteria were met and the PSN gives the SCU control over the call.



Optional parameter Tag: 130

Parameter length: 2 byte

Parameter contents:

- The POINT IN CALL field consists of five bits and is the location which contains the point in the call when the PSN determines that the call is a service call and needs to be controlled by the SCU. It is encoded as follows:

0	0 0 0 0 0:	Not Used
1	0 0 0 0 1:	Orig Null
2	0 0 0 1 0:	Authorize Orig Attempt
3	0 0 0 1 1:	Collect Information
4	0 0 1 0 0:	Analyze Information*
5	0 0 1 0 1:	Select Route
6	0 0 1 1 0:	Authorize Call Setup
7	0 0 1 1 1:	Send Call
8	0 1 0 0 0:	Orig Alerting
9	0 1 0 0 1:	Orig Active
10	0 1 0 1 0:	Orig Suspended
11	0 1 0 1 1:	Term Null
12	0 1 1 0 0:	Authorize Termination
13	0 1 1 0 1:	Select Facility
14	0 1 1 1 0:	Present Call
15	0 1 1 1 1:	Term Alerting
16	1 0 0 0 0:	Term Active
17	1 0 0 0 1:	Term Suspended

1 0 0 1 0 to 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

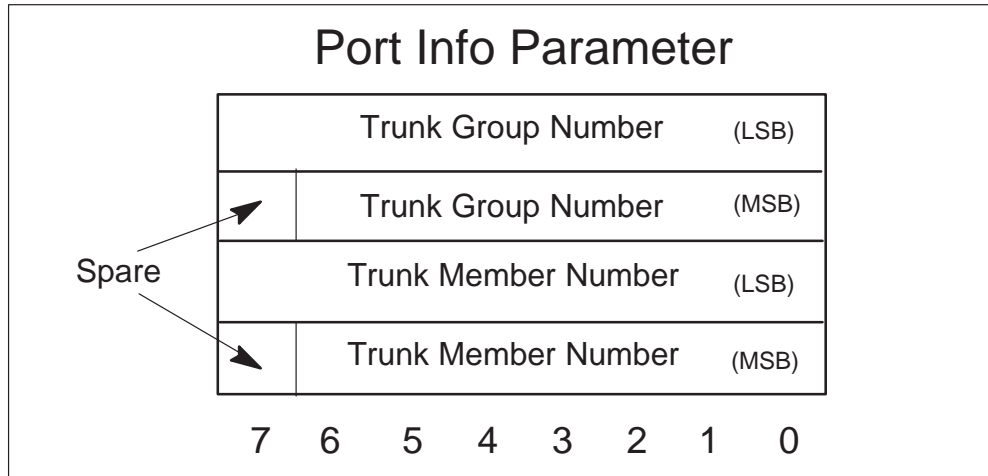
- The TRIGGER CRITERIA field consists of eight bits and encoded as follows:

0 0 0 0 0 0 0 0: Feature Activator
1 0 0 0 0 0 0 1: Vertical Service Code
2 0 0 0 0 0 1 0: Customized Access
3 0 0 0 0 0 1 1: Customized Intercom*
4 0 0 0 0 1 0 0: NPA
5 0 0 0 0 1 0 1: NPA_NXX
6 0 0 0 0 1 1 0: NXX
7 0 0 0 0 1 1 1: NXX_XXXX
8 0 0 0 1 0 0 0: NPA_NXXXXXX*
9 0 0 0 1 0 0 1: Country_Code_NPA_NXXXXXX*
10 0 0 1 0 0 0 0: Offhook Immed
11 0 0 1 1 0 0 0: Net Busy
12 0 0 1 1 0 1 1: Orig Called Party Busy
13 0 0 1 1 1 0 1: Orig No Answer
14 0 0 1 0 0 0 0: Orig Feature Activator
15 0 1 1 0 0 0 0: Channel Setup PRI CLID
16 0 1 1 0 0 0 1: Channel Setup PRI Addr
17 0 1 1 0 0 1 0: Channel Setup PRI N00
18 0 1 1 0 0 1 1: Channel Setup PRI Intl
19 0 1 1 0 0 1 0 0: Specific Digit String Info*
20 0 1 1 0 0 1 0 1: Specific Digit String ANI*
21 0 1 1 0 0 1 1 0: Specific Digit String N00*
22 0 1 1 0 0 1 1 1: Specific Digit String CIC
23 0 1 1 0 1 0 0 0: Shared Interoffice CIC
24 0 1 1 0 1 0 0 1: Shared Interoffice Info
25 0 1 1 0 1 0 1 0: Shared Interoffice ANI
26 0 1 1 0 1 0 1 1: Shared Interoffice Addr
27 0 1 1 0 1 1 0 0: Shared Interoffice N00
28 0 1 1 0 1 1 0 1: Shared Interoffice Intl
0 1 1 0 1 1 1 0 to 1 1 1 1 1 1 1 1: Unused (Spare)

The values marked with an “*” are the only ones that are currently supported.

PortInfo

This parameter contains the port information for an agent. The port info consists of the external Trunk Group Number and the Trunk Member Number.



Optional parameter ID: 19

Parameter length: 4 bytes

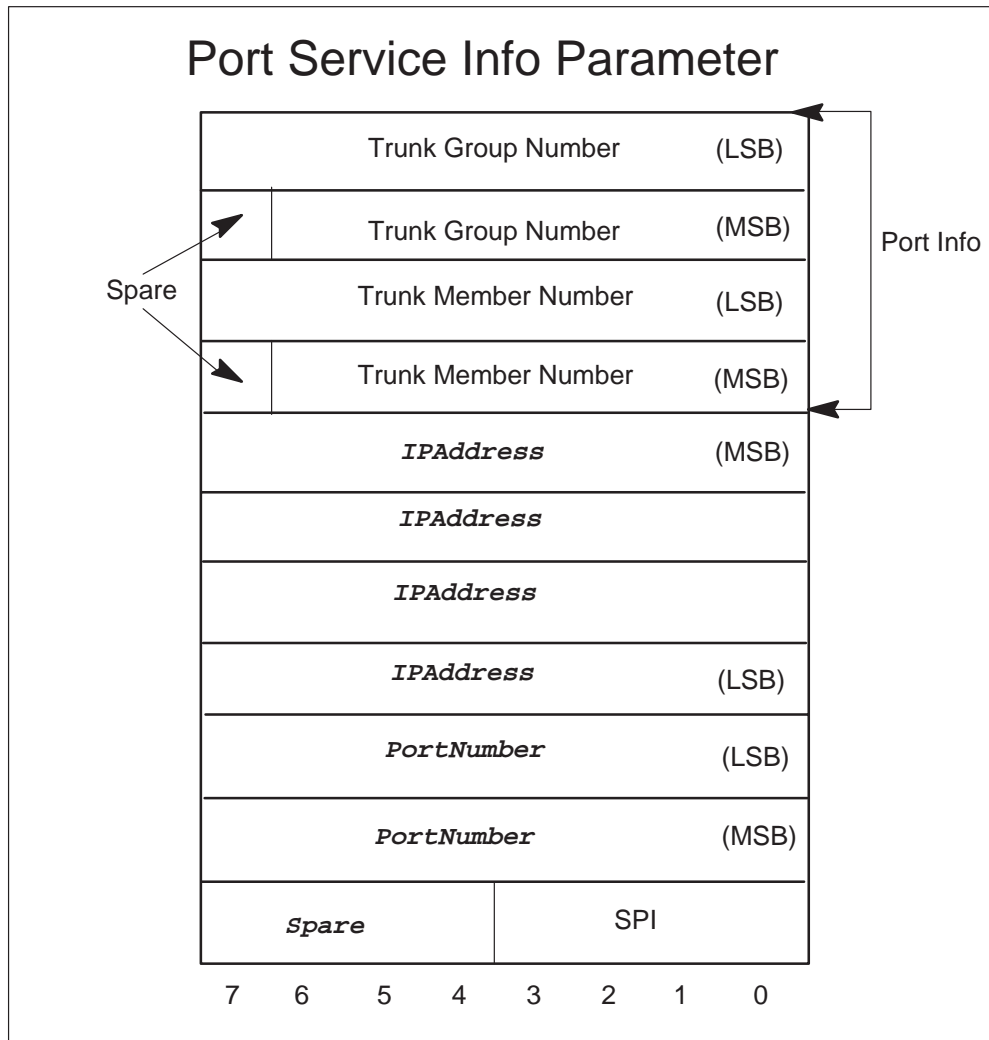
Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the location which contains the external Trunk Group Number. The range is from 0 to 9,999. A value of 32,767 indicates a NIL_TRUNK_GROUP.
- The TRUNK MEMBER NUMBER field consists of 15 bits and is the location which contains the Trunk Member Number. The range is from 0 to 9,999. A value of 32,767 indicates a NIL_TRUNK_MEMBER.

PortServiceInfo

This parameter contains the IP address and SPI information. The SCU returns the IP address and SPI to the PSN. The IP address is the return address used to send the event notification messages back to the SCU (to the

right address and port). The SPI number is the service programming interface version for the specified agent.



Optional parameter ID: 20

Parameter length: 11 bytes

Parameter contents:

- The PORT INFO field consists of four bytes and is the location in which the IP address and the SPI are updated. The external Trunk Group Number and the Trunk Member Number are specified in the *PortInfo* parameter. The range is from 0–9,999 for both.

A value of 32,767 for the Trunk Group Number indicates a NIL_TRUNK_GROUP.

A value of 32,767 for Trunk Member Number indicates a `NIL_TRUNK_MEMBER`.

If the Trunk Group Number is set to `NIL_TRUNK_GROUP` and the Trunk Member Number is set to `NIL_TRUNK_MEMBER`, then the returned IP address and SPI information corresponds to the SCU Arbitrator address. The Arbitrator address is the initial point of contact to send all the New Call event notification messages.

Information in the following table illustrates how the combination of the Trunk Group Number and the Trunk Member Number in the `PortInfo` parameter is interpreted.

Trunk Group Number	Trunk Member Number	What is the IP Address Info used for
<code>nil_trunk_group</code>	<code>nil_trunk_member</code>	Arbitrator's IP address if recvd. in the Set_IP_Address primitive. INVALID if recvd. with other primitives.
Non <code>nil_trunk_group</code>	<code>nil_trunk_member</code>	IP Address for destination trunk group in Connect primitive only.
<code>nil_trunk_group</code>	Non <code>nil_trunk_member</code>	INVALID
Non <code>nil_trunk_group</code>	Non <code>nil_trunk_member</code>	IP Address for the port specified by the trunk group/member.

- The IP ADDRESS field consists of four bytes and is the location which contains the IP address sent by the SCU and which is eventually used to return the event notification messages.

This is a table of four bytes. For example, an IP address of 47.122.64.153 is stored as four bytes: byte one is 47, byte two is 122, byte three is 64, and byte four is 153. The `nil_ip_address` is defined as 0.0.0.0.

- The PORT NUMBER field consists of two bytes and is the location which contains the transport layer port number associated with the IP address. The `nil_port_number` is defined as 0.

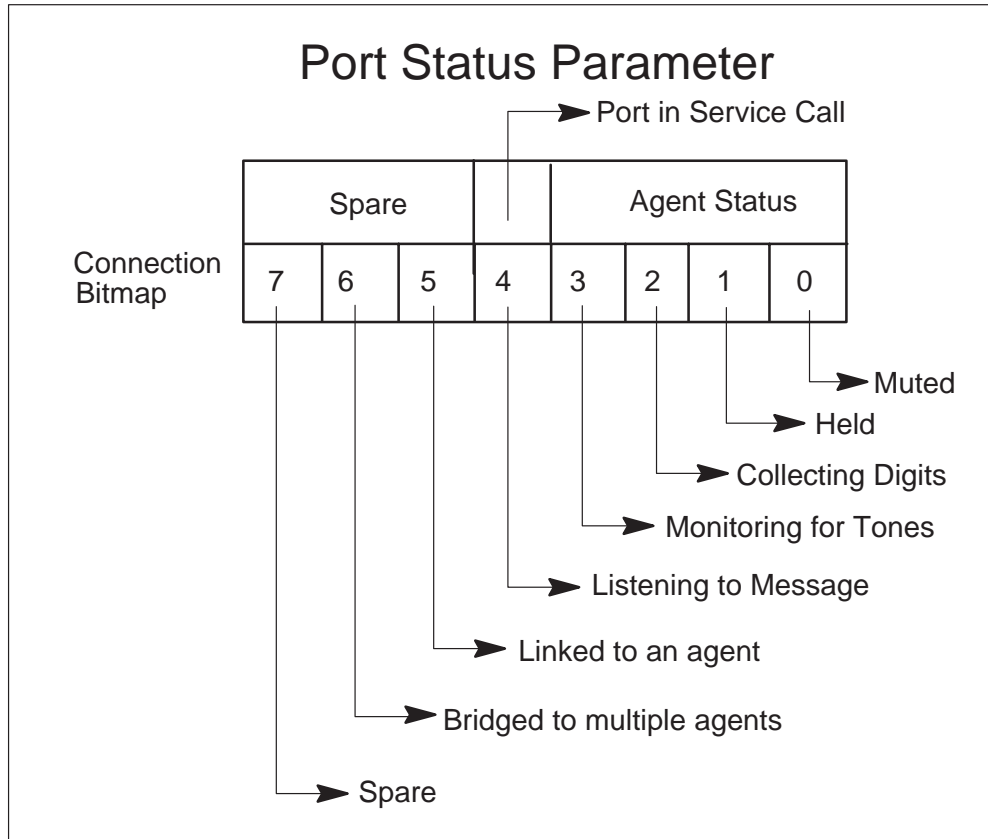
Information in the following table illustrates how the combination of IP Address and Port Number in the IP Address Info parameter are interpreted for the different primitives.

IP Address	Port Number	How is the IP Address Info used for in RESET SWITCH	How is the IP Address Info used in all the other Primitives
nil_ip_address	nil_port_number	Idle all ports that are currently serviced by the SCU (System Reset)	INVALID
Non nil_ip_address	nil_port_number	Idle all ports with the given IP Address and any port number (Shelf Reset)	INVALID
nil_ip_address	Non nil_port_number	INVALID	INVALID
Non nil_ip_address	Non nil_port_number	Idle ports with the appropriate IP Address Info (Service Reset)	<i>IPAddressInfo</i> parameter applies to a specific port – update the port's IP Address Info.

- The SPI field consists of four bits and is the location which contains the SPI version for the specified agent. The range is from 1 to 15.

PortStatus

This parameter contains the status of the port and agent. It also contains information which indicates if the port is currently being controlled by the SCU.



Optional parameter ID: 21

Parameter length: 2 bytes

Parameter contents:

- The AGENT STATUS field consists of four bits and is the location which contains the agent status of the port. This parameter is encoded as follows:

0 0000: Idle
 1 0001: Seized
 2 0010: Answered
 3 0011: ManBusy
 4 0100: Lockout
 5 0101: System Busy

6 0 0 1 1 0: PM Busy
7 0 0 1 1 1: Unknown
1 0 0 0 to 1 1 1 1: Spare Values

- The PORT IN SERVICE CALL field consists of one bit and is the location which indicates if the port is currently a part of a service call, or if the port is currently being serviced by the SCU.

0: Port not a part of a service call
1: Port is a part of a service call.

- The CONNECTION BITMAP field consists of six bits and is a bitmap where each bit indicates what the connection status of the agent is. The bits are used to represent the connection states as specified below:

Bit 0: Muted

Value: 0 = port is unmuted
Value: 1 = port is muted

Bit 1: Held

Value: 0 = port is not held
Value: 1 = port is held

Bit 2: Collecting Digits

Value: 0 = port is not collecting digits
Value: 1 = port is in the process of collecting digits

Bit 3: Monitoring for Tones

Value: 0 = port is not monitoring for tones
Value: 1 = port is monitoring for tones

Bit 4: Listening to Message

Value: 0 = port is not listening to a message
Value: 1 = port is listening to message (announcement and tones)

Bit 5: Linked to an agent

Value: 0 = port is not linked to another agent
Value: 1 = port is linked to another agent

Bit 6: Bridged to multiple agents

Value: 0 = port is not bridged to two or more agents
Value: 1 = port is bridged to two or more agents

ResetReason

This parameter contains the reason why a restart was performed on the PSN. It indicates to the SCU the type of reset that is required at the SCU.



Optional parameter ID: 22

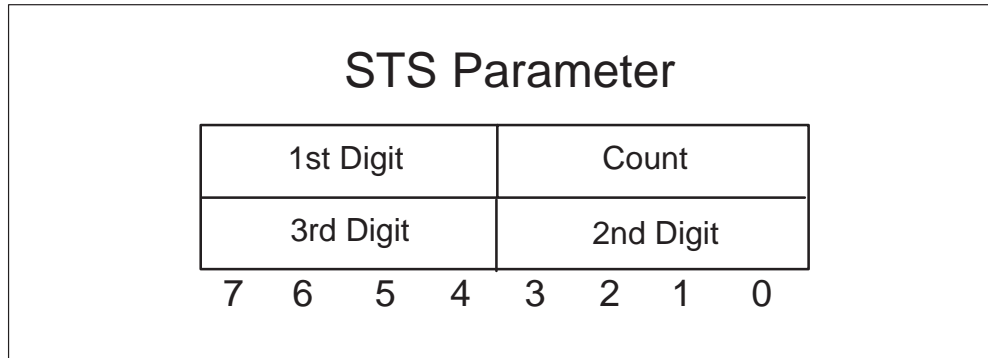
Parameter length: 1 byte

Parameter contents:

- The RESET REASON field consists of four bits and is the location which indicates the reason why a restart was performed on the PSN. The values include:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Warm Restart performed on the PSN
 - 2 0 0 1 0: Cold Restart performed on the PSN
 - 3 0 0 1 1: Reload Restart performed on the PSN
 - 4 0 1 0 0: PSN has just come into service
 - 5 0 1 0 1: Arbitrator Heartbeat has failed
 - 0 1 1 0 to 1 1 1 1: Spare Values

ServingTranslationScheme

This parameter contains the serving translation scheme (STS) of the call. The switch sends this parameter to the SCU in the New Call event notification message.



Optional parameter Tag: 131

Parameter length: 2 bytes

Parameter contents:

- The COUNT field consists of four bits and is the location which contains the number of digits in the STS. The values include from 1 to 3.
- The DIGITS field consists of one, two, and three digits with each digit consisting of four bits. The three digits contained in this field are in the TBCD format and are as follows:

```

0 0 0 0: Filler
1 0 0 0 1: Digit 1
2 0 0 1 0: Digit 2
3 0 0 1 1: Digit 3
4 0 1 0 0: Digit 4
5 0 1 0 1: Digit 5
6 0 1 1 0: Digit 6
7 0 1 1 1: Digit 7
8 1 0 0 0: Digit 8
9 1 0 0 1: Digit 9
10 1 0 1 0: Digit 0
11 1 0 1 1: *
12 1 1 0 0: #
13 1 1 0 1: D
14 1 1 1 0: E
15 1 1 1 1: F

```


Depending upon the primitive or event notification in which this parameter is sent or received, and the signaling type of the associated port, its contents are different. In general, the contents contain parameters that are encoded in the standard format.

For PTS agents, there are no standards used. For SS7 agents, the parameters are encoded based on the TR444 and the GR394. For PRI agents, refer to Chapter “PRI messages” for further details.

SignalingInfo is included in the following primitives and event notification messages.

- **Connect** primitive from the SCU (receive use):
 - The following parameters are sent in Signaling Info on a PTS agent. At least one stream of digits (a **DigitstoOutputpulse** parameter) must be sent with the capability of sending up to three streams. These parameters are encoded as described earlier in “Bearer Capability” and “Digits To Outputpulse.”

<i>BearerCapability</i>	(Optional)
<i>DigitstoOutputpulse</i>	(Mandatory)
<i>DigitstoOutputpulse</i>	(Optional)
<i>DigitstoOutputpulse</i>	(Optional)

- The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)
Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)
RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)
XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)
Originating Line Information	(Optional)
Supplementary Line Info	(Optional)

Transit Network Selection	(Optional)
---------------------------	------------

- The following SETUP message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **Disconnect** primitive from the SCU(receive use):

- The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)

- The following DISC message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)

- The following REL message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- **Transmit_Siginfo** primitive from the SCU(receive use):

- The following parameters are sent in the SigInfo parameter on a PTS agent. At least one stream of digits (a Digits to Output parameter) must be sent with the capability of sending up to three streams. The signaling parameter is encoded as described earlier in “Digits To Output”.

Message Type	(Mandatory)
Digits to Outpulse	(Mandatory)
Digits to Outpulse	(Optional)
Digits to Outpulse	(Optional)

- The following information is sent (on a PTS agent) for a *PTS On-hook*. There is no signaling information present when the message type is *PTS On-hook*:

Message Type	(Mandatory)
--------------	-------------

- The following information is sent (on a PTS agent) for a *PTS Off-hook*. There is no signaling information present when the message type is *PTS Off-hook*:

Message Type	(Mandatory)
--------------	-------------

The **Transmit_siginfo** is used to send the IAM, ANM, CPG, FAA, FAR, FRJ, PAM, SUS, REL, and RES for an SS7 port.

The **Transmit_siginfo** is used to send the CONNECT, CALLPROC, FACILITY, RLC, REL, and DISC messages for a PRI port.

For the definition of the SS7 and PRI messages, refer to the “Signaling Event” bullet in this section.

- **off_hook** event notification to the SCU(send use):

- The following ANM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Optional)
Call Reference	(Optional)
Carrier Selection	(Optional)
Internetwork Specific ANM	(Optional)
Intranetwork Specific ANM	(Optional)
Network Specific ANM	(Optional)
Operator Information	(Optional)
US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following CONNECT message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User to User Information	(Optional)

- **On_Hook** event notification to the SCU (send use):
 - The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)
User to User Indicator	(Optional)
User to User Information	(Optional)
 - The following RSC message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
--------------	-------------
 - The following DISC message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)
User to User Information	(Optional)
 - The following REL message parameters (on a PRI agent) are encoded as in Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)
- **New_Call** event notification to the SCU(send use):
 - The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)
Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)

RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)
XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)
Originating Line Information	(Optional)
Supplementary Line Info	(Optional)
Transit Network Selection	(Optional)

- The following SETUP message parameters (on a PRI agent) are encoded as in Chapter “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **signaling_Event** event notification to the SCU:

- The following parameters are sent in the SigInfo parameter on a PTS agent. These parameters are encoded as described in “Digits Outpulsed”.

Digits Outpulsed Indicator	(Mandatory)
----------------------------	-------------

- The following ACM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Mandatory)
Call Reference	(Optional)
US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following COT message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
--------------	-------------

Continuity Indicators (Mandatory)

- The following CPG message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Event Information	(Mandatory)
Backward Call Indicator	(Mandatory)
Party Information	(Optional)
Redirection Number	(Optional)
Supply end-to-end Inf. Req.	(Optional)
Supply end-to-end Inf. Resp.	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following FAA message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)

- The following FAR message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)
Called Party Address	(Optional)
Calling Party Number	(Optional)
Charge Adjustment	(Optional)
Charge Number	(Optional)
Callid (GD)	(Optional)
Operator Information	(Optional)
Originating Line Info	(Optional)

- The following FRJ message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Cause Indicator	(Mandatory)
Call Reference	(Optional)

- The following PAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
RPM Message	(Mandatory)

- The following RES message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend/Resume Indicators	(Mandatory)

- The following SUS message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend/Resume Indicators	(Mandatory)

- The following PROGRESS message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Progress Indicator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- The following ALERT message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User-User Information	(Optional)
Progress Indicator	(Optional)

- The following CALL PROCEEDING message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)

- The following FACILITY message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

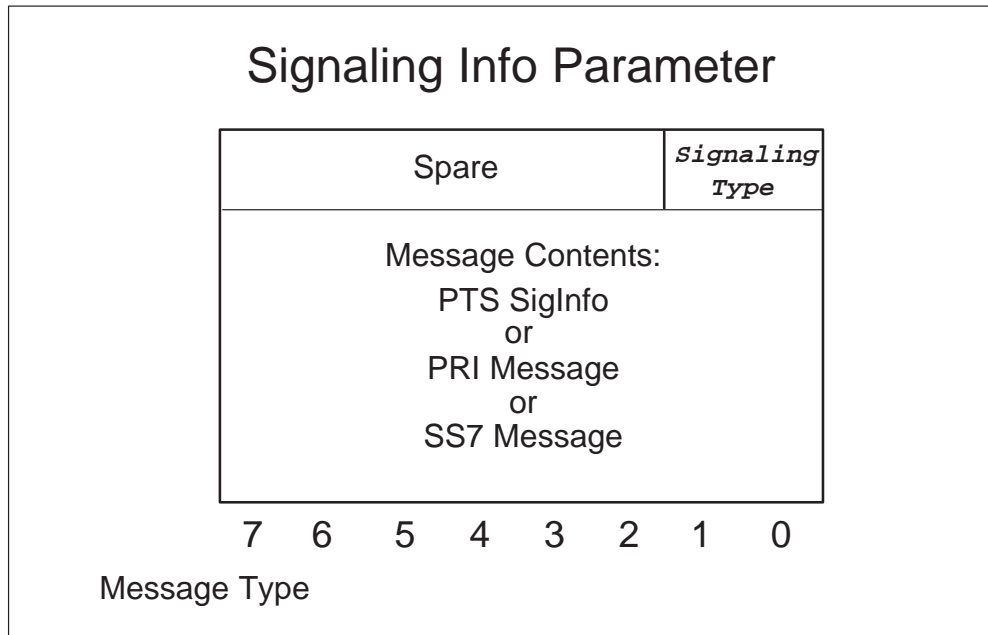
Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Called Party Number	(Optional)
Called Party Subaddress	(Optional)

- The following REL Complete (RLC) message parameters (on a PRI agent) are encoded as indicated in the Chapter “PRI messages”.

Message Type	(Mandatory)
--------------	-------------

User to User Information

(Optional)



Optional parameter ID: 25

Parameter length: maximum of 412 bytes

Parameter contents:

- The Signaling TYPE field consists of two bits and is the location which contains the type of signaling that the port is using. The values include:
 - 0 0: PTS (4 Wire)
 - 0 1: PTS (2 Wire)
 - 1 0: SS7
 - 1 1: PRI

- The MESSAGE CONTENTS field consists of a maximum of 411 bytes. This field is the location of the message contents which contains the parameters in the standard format. The contents are encoded depending upon the type of signaling used.

Primitive/Event Notification	PTS Messages	SS7 Messages	PRI Messages
Connect	<i>Digits_To_Outpulse,</i> <i>Digits_To_Outpulse with Bearer_Capability</i>	IAM	SETUP
Disconnect	—	REL	DISC, REL
Transmit_Siginfo	<i>Digits_To_Outpulse,</i> <i>PTS_On-hook,</i> <i>PTS_Off-hook</i>	IAM, ANM, REL, CPG, FAA, FAR, FRJ, PAM, RES, SUS	CONNECT, REL, DISC, ALERT, FACILITY
New_Call	—	IAM	SETUP
Off_Hook	—	ANM	CONNECT
On_Hook	—	REL, RSC	DISC, REL
Signaling_Event	<i>Digits_Outpulsed_Indicator</i>	ACM, CPG, COT, FAA, FAR, FRJ, PAM, RES, SUS	PROGRESS, ALERT, CALL PROCEEDING, FACILITY, RELEASE COMPLETE

— PTS Message contents:

- The PTS MESSAGE TYPE field consists of one byte and is the location which contains the message type for a PTS message. The values include:
 - 0 00000000: Unknown
 - 1 00000001: Digits to Outputpulse
 - 2 00000010: Digits to Outputpulse with Bearer Capability
 - 3 00000011: PTS Off-Hook
 - 4 00000100: PTS On-Hook
 - 5 00000101: Digits Outputpulsed
 - 00000110 to 11111111: Spare
- The PTS SIGINFO MESSAGE AREA field consists of a maximum of 410 bytes and is the location which contains the the SigInfo message but depends on the PTS message type. If PTS message type is:
 - Digits to Outputpulse, then the PTS SigInfo message area contains from 1 to 3 Digits to Outputpulse parameters. Refer to section “Digits To Outputpulse,” for further information.

DigitstoOutpulse with Bearer Capability, then the PTS SigInfo message area contains the *BearerCapability* parameter first in the area, “Bearer Capability,” with the following bytes containing from 1 to 3 *DigitstoOutpulse* parameters, “Digits To Outpulse”.

PTS Off-hook, then there is no information in the PTS SigInfo message area. In other words, there is no signaling information associated with a PTS Off-hook.

PTS On-hook, then there is no information in the PTS SigInfo message area. In other words, there is no signaling information associated with a PTS On-hook.

Digits Outpulsed, then PTS SigInfo message area contains the Digits Outpulsed parameter. Refer to “Digits Outpulsed”.

— SS7 Message contents:

The TR444 and the GR394 SS7 with ISDN support are used to define the contents of the SS7 message contents.

— PRI Message contents:

The SS7 and PRI parameter contents will not be redefined in this document.

SigInfo_Mask

This parameter lets the SCU control which of the optional SS7 and PRI messages are reported to the SCU in the Signaling Event event notification message.

For example, if the port is an SS7 port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ACM (Address Complete)
- COT (Continuity)
- CPG (Call Progress)
- FAA (Facility Activation), FAR (Facility Request), FRJ (Facility Reject)
- PAM (Pass Along Message)
- RES (Resume) and SUS (Suspend)

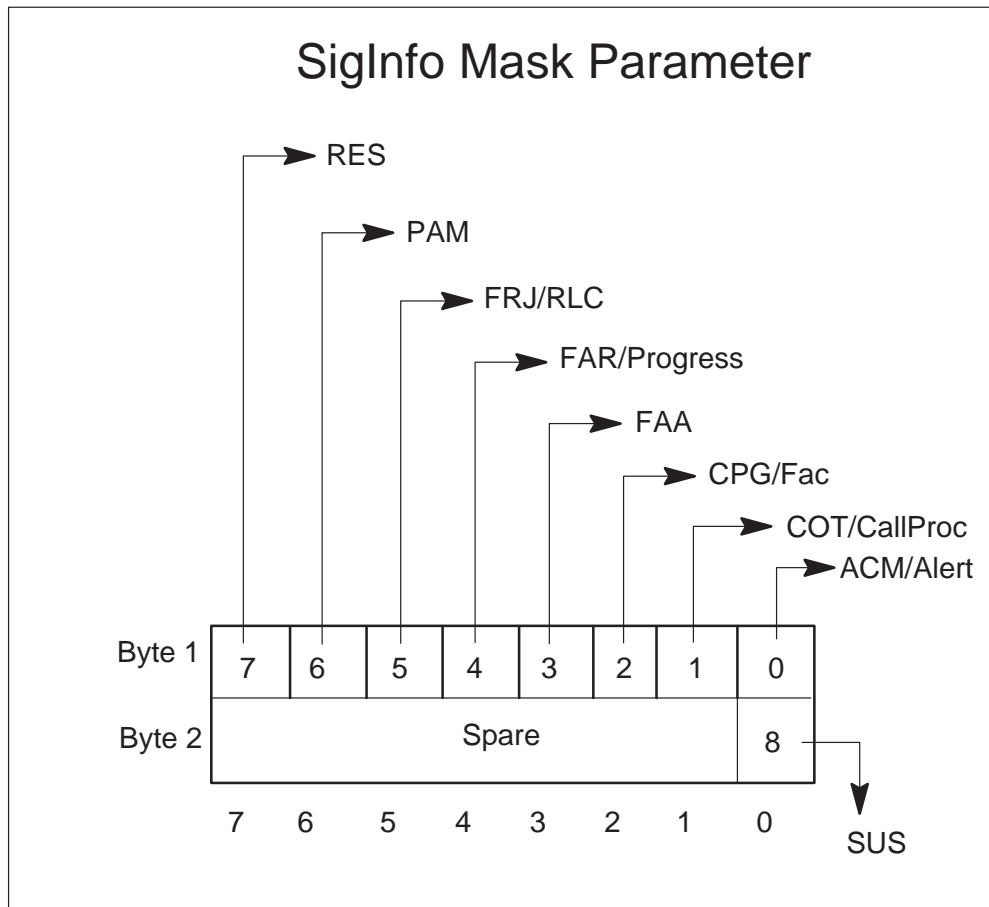
If the port is a PRI port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ALERT (Alerting Message)

- CALLPROC (Call Proceeding)
- FAC (Facility)
- RLC (Release Complete)
- PROG (Progress)

The following messages are mandatory and are always sent to the SCU. In addition, the receipt of these messages cannot be turned off by the SCU.

- ANM (Answer), REL (Release), and RSC (Reset Circuit) for SS7 ports
- CONNECT and REL (Release) and DISC (Disconnect) for PRI ports



Optional parameter ID: 24

Parameter length: 2 bytes

Parameter contents:

- The SIGINFO MASK BITMAP field consists of nine bits and is a bitmap in which each bit indicates what the connection status of the agent is. The bits are used to represent the connection states as specified by the following:

Bit 0: ACM/Alert

Value: 0 = do not send ACM (for SS7 port) or Alert (for PRI port) message to the SCU.

Value 1 = send the ACM (for SS7 port) or Alert (for PRI port) to the SCU.

Bit 1: COT/CallProc

Value: 0 = do not send COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU.

Value: 1 = send the COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU.

Bit 2: CPG/FAC

Value: 0 = do not send CPG (for SS7 port) or FAC (for PRI port) message to the SCU.

Value: 1 = send the CPG (for SS7 port) or FAC (for PRI port) to the SCU.

Bit 3: FAA

Value: 0 = do not send FAA (for SS7 port) message to the SCU.

Value: 1 = send the FAA (for SS7 port) to the SCU.

Bit 4: FAR/Progress

Value: 0 = do not send FAR (for SS7 port) or Progress (for PRI port) message to the SCU.

Value: 1 = send the FAR (for SS7 port) or Progress (for PRI port) to the SCU.

Bit 5: FRJ/RLC

Value: 0 = do not send FRJ (for SS7 port) or RLC (for PRI port) message to the SCU.

Value: 1 = send the FRJ (for SS7 port) or RLC (for PRI port) to the SCU.

Bit 6: PAM

Value: 0 = do not send PAM (for the SS7 port) to the SCU.

Value: 1 = send the PAM (recvd. on the SS7 port) to the SCU.

Bit 7: RES

Value: 0 = do not send RES (for the SS7 port) to the SCU.

Value: 1 = send the RES (recvd. on the SS7 port) to the SCU.

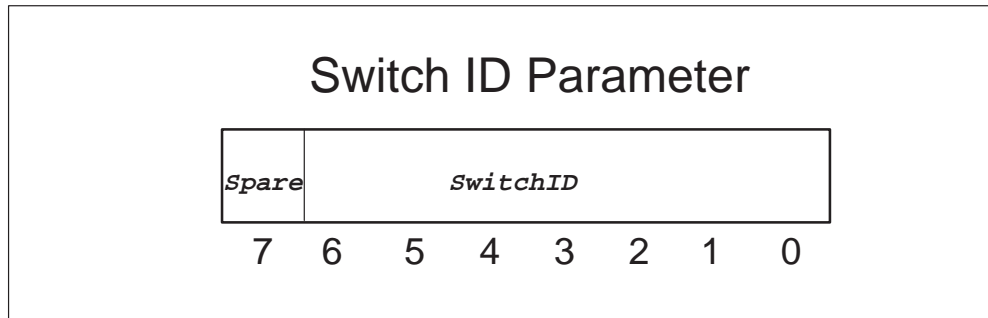
Bit 8: SUS

Value: 0 = do not send SUS (for the SS7 port) to the SCU.
Value: 1 = send the SUS (recvd. on the SS7 port) to the SCU.

Bit 9 – 15: Spare.

Switch_ID

This parameter contains the switch ID of the PSN.



Optional parameter ID: 26

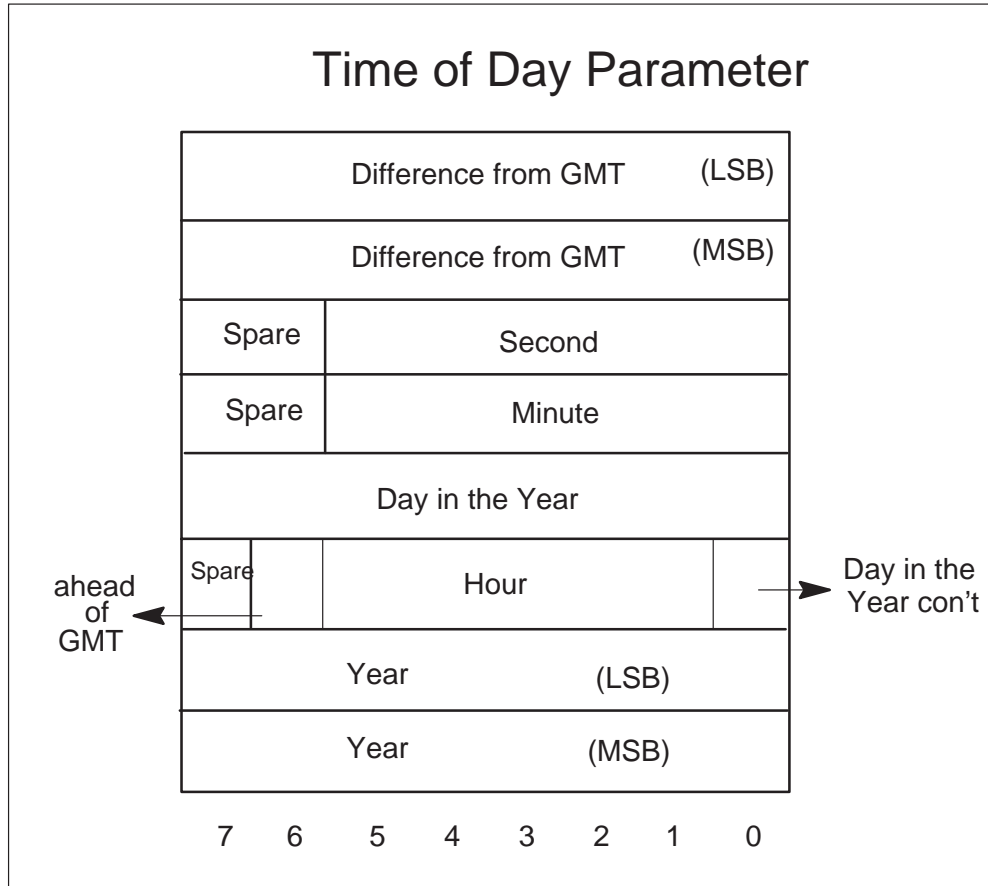
Parameter length: 1 byte

Parameter contents:

- The SWITCH ID field consists of seven bits and is the location which contains the switch ID. The range is from 0 to 127.

Time_of_day

This parameter will contain the time of the day and will be returned in the Current Time of Day event notification to the SCU.



Optional parameter ID: 27

Parameter length: 8 bytes

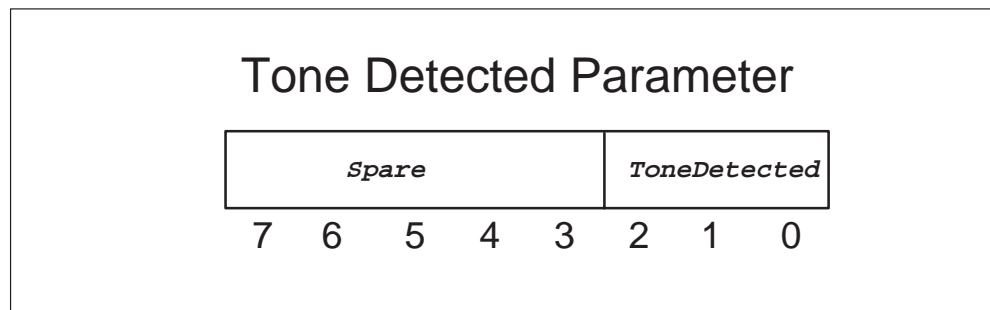
Parameter contents:

- The YEAR field consists two bytes and is the location which contains the year.
- The DIFFERENCE FROM GMT field consists of two bytes and is the location which contains the time difference between the time specified by the “hour, minute and second” fields above, and the Greenwich Mean Time.
- The DAY IN THE YEAR field consists of nine bits and is the location which contains the day of the year. The values include: 1 to 365.

- The MINUTE field consists of six bits and is the location which contains the minute. The values include: 0 to 59. The additional values are spares.
- AHEAD OF GMT: 1 bits – If true, this field indicates that the switch time is AHEAD of the Greenwich Mean Time (GMT). The switch time is BEHIND the GMT if this field is set to false.
- HOUR: 5 bits – This field contains the hour. The values include: 0 to 23. The rest are spare values.
- SECOND: 6 bits – This field contains the minute. The values include: 0 to 59. The additional values are spares.

Tone_Detected

This parameter will contain the tone that was detected on a given port and agent by the PSN.



Optional parameter ID: 28

Parameter length: 1 byte

Parameter contents:

- The TONE OR DIGIT DETECTED consists of three bits and is the location which contains the tone or the digit that was detected, and is encoded as follows:
 - 0 0 0: tone / digit monitoring aborted
 - 1 0 0: asterisk detected
 - 2 0 1 0: octothorpe detected
 - 3 0 1 1: SF tone detected
 - 4 1 0 0: BBF tone detected
 - 5 1 0 1: BBF digits detected
 - 6 1 1 0: BBF not possible on this port
 - 1 1 1: Unused (Spare)

UCS PSN LOPER messages and parameters version 1

This chapter contains a detailed description of the UCS PSN messages and parameters described in the Low Overhead Protocol Encoding Rule (LOPER) message format.

LOPER format

Each of the messages in the LOPER follow the same basic format and rules.

The following rules apply to LOPER:

- A message is byte aligned, meaning that each parameter starts on a new byte.
- The first byte in a message is considered to be the 0th index of the message, the second byte in the message is considered to be the first index of the message, and so on.
- Two bytes are used in describing the length of a message parameter. Two bytes are also used to contain the byte location in the message that the optional parameters begin in the message.
- A LOPER message can either contain one primitive, one event, or one macro which can contain up to five primitives.
- The SS7 signaling message of the SigInfo Contents in a optional *SigInfo* Parameter to be transmitted on a SS7 agent is not be validated. Since the SS7 signaling message is to be built by the SCU in the TR444 BellCore standard, it is assumed that the SS7 signaling message has been correctly built and is sent onto the SS7 network with out validation. Any error in the SS7 signaling message is caught by the network and handled accordingly by the SS7 network.

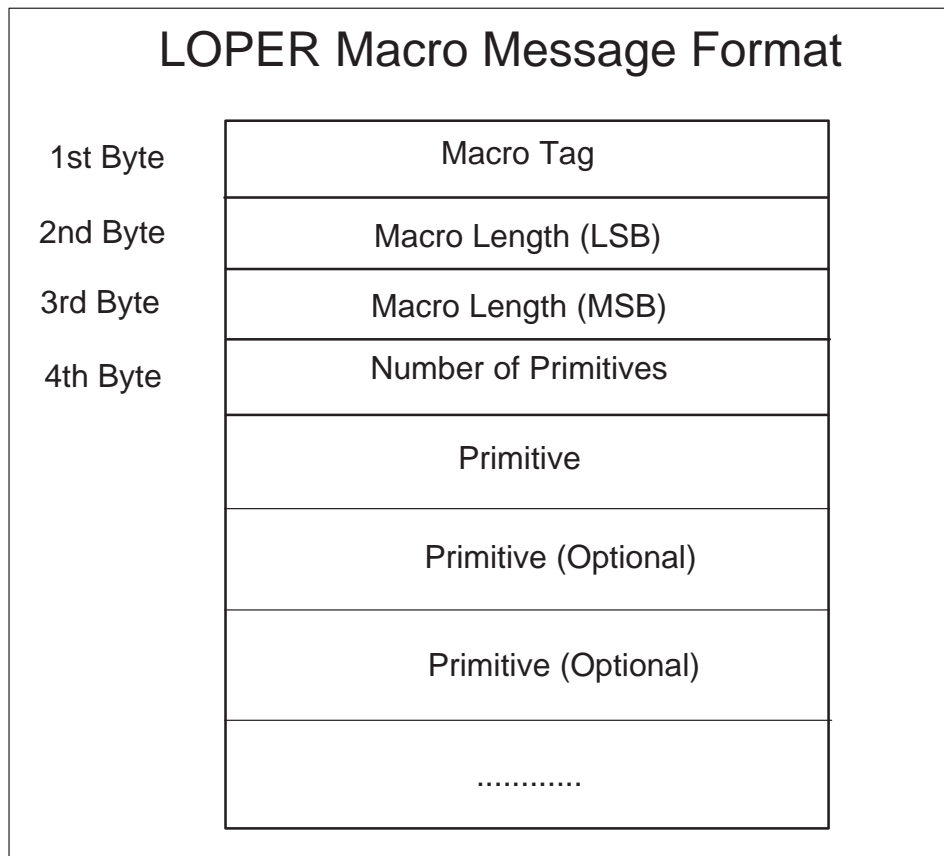
Messages sent from the SCU to the PSN

This section contains information on messages sent from the SCU to the PSN.

Macros

The first byte of a macro designates that the message is a macro. The second and third bytes indicate the length of the macro in the number of bytes that comprise the length of the message from the macro Tag to the end of the last primitive in the macro. The fourth byte identifies how many primitives there are in the Primitive Area. The Primitive Area contains the individual primitives. Refer to the following diagram.

If there is an error in the decoding of one of the primitives found in the macro, that primitive is dropped and the decoding process is ceased because the rest of the macro is considered to be corrupted. Any primitive decoded before the error is still processed.

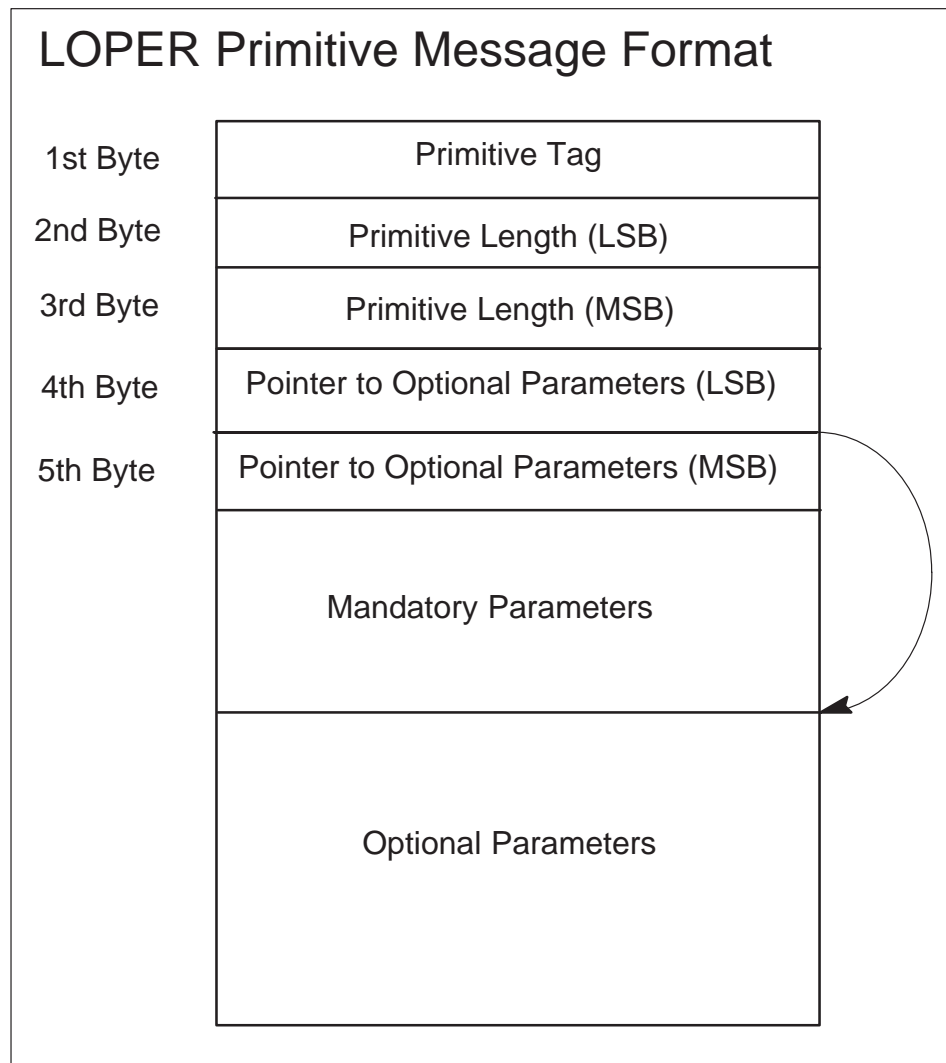


- The MACRO TAG field consists of one byte and is the location which is used to identify the primitive in the message. The values include:

0 0 0 0 0 0 0 0 : Macro Tag

Primitives

The first byte is the Primitive Tag and identifies the primitive. The second and third bytes indicate the length of the primitive, beginning with the primitive tag to the end of the mandatory parameters or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters. The contents of the mandatory parameter depend on the primitive and event, and the customer using LOPER. Refer to the following diagram:



- The PRIMITIVE TAG field consists of one byte and is the location which is used to identify the primitive in the message. The values include:

```
0 00000001: Bridge
1 00000010: Collect_Digits
2 00000011: Connect
3 00000100: Disconnect
4 00000101: Hold
5 00000110: Monitor
6 00000111: Mute
7 00001000: RESERVED FOR INTERNAL PSN USE
8 00001001: New_Call_Accepted
9 00001010: New_Call_Rejected
10 00001011: Play_Message
11 00001100: PPCD
12 00001101: Query_Port
13 00001110: Reconnect
14 00001111: Set_Billing_Record
15 00010000: Stop_Message
16 00010001: Transmit_Siginfo
17 00010010: Error_Detected
18 00010011: Heartbeat
19 00010100: Port_Status
20 00010101: Query_TOD
21 00010110: Reset_Switch
22 00010111: Set_IP_Address
23 00011000: Flow_Control
    00011001 to 11111111: Spare
```

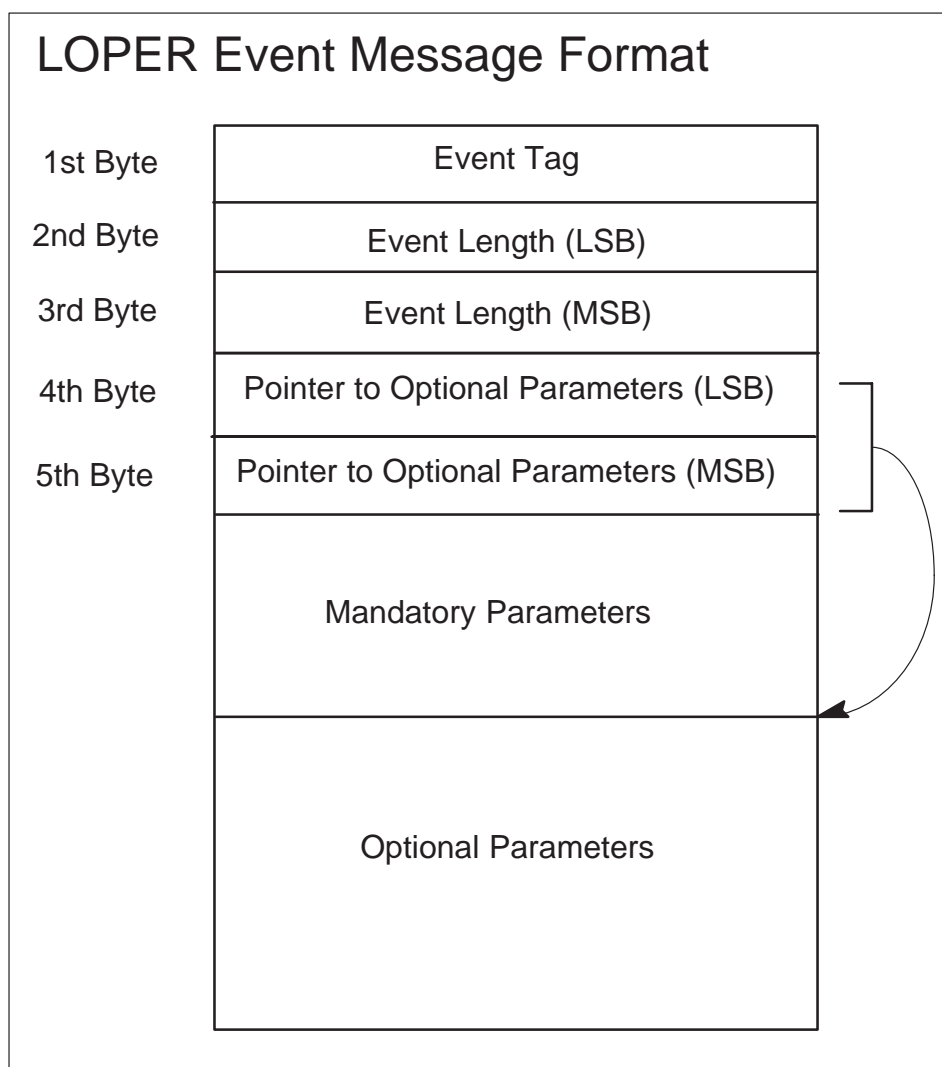
Messages sent from the PSN to the SCU

This section contains information on messages sent from the PSN to the SCU.

Events

The first byte is the Event Tag and identifies the event. The second and third bytes indicate the length of the event, beginning with the Event Tag to the end of the mandatory parameters, or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters.

The contents of the mandatory parameter depend on the primitive and event, and the customer using LOPER. Refer to the following diagram.



- The EVENT TAG field consists of one byte and is the location which is used to identify the event in the message. The values include:

```

0 00000000: Current_TOD
1 00000001: Digit_Collected
2 00000010: Error_Detected
3 00000011: In_Service
4 00000100: Instruction_Completed
5 00000101: Message_Played
6 00000110: New_Call
7 00000111: Off_Hook

```

```
8 00001000: On_Hook
9 00001001: Port_Status
10 00001010: Query_Port
11 00001011: Route_Not_Available
12 00001100: Route_Selected
13 00001101: Signaling_Event
14 00001110: Tone_Detected
    00001111 to 11111111: Spare
```

Parameters

LOPER divides parameters into two classes, mandatory and optional. Mandatory parameters contain the information required by the primitive and event in order to be processed. Optional parameters contain the information not required by the primitive and event but add extra information for processing.

Mandatory parameters

Mandatory parameters in LOPER are customized for each primitive and event. The sixth byte in the primitive and event is always the beginning of the mandatory parameters for primitives and events. There is no need for a mandatory parameter tag. Mandatory parameters are identified by the primitive and event ID and can only be found in their corresponding primitive and event, such as a Connect mandatory parameter can only be found in a **connect** primitive. Also, there is no need for a length indicator for the mandatory parameters. For example, since the pointer to the optional parameters designates the end of the mandatory parameters, and if there are no optional parameters, then the length of the primitive and event is the end of the mandatory parameters.

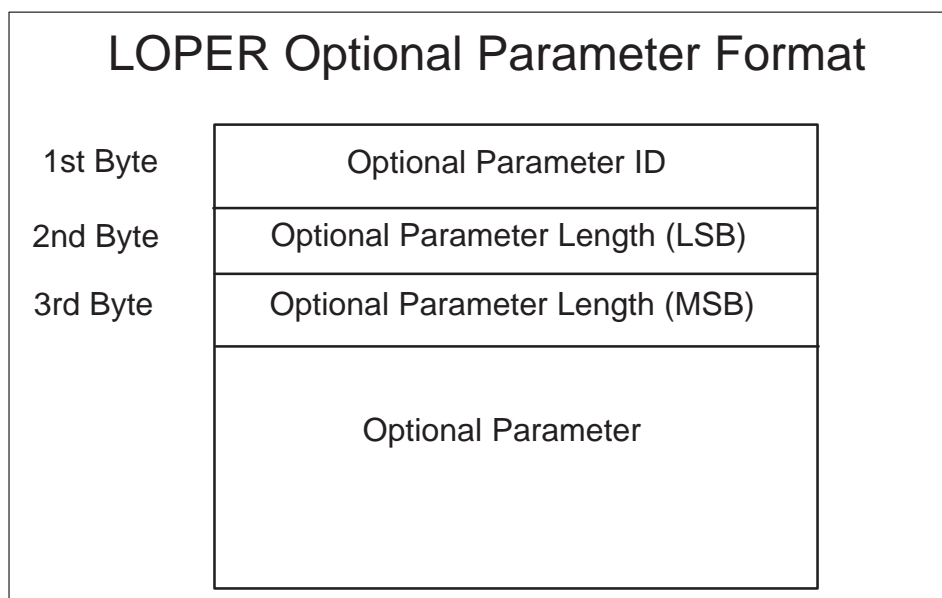
If there is an error in the decoding of the mandatory parameters, the entire message is dropped, the OM PSN_ERPS:DECODEFL is incremented, logs PSN202 and PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause HEADER_DECODE_FAILURE_EC. If there is an error in the validating of the mandatory parameters values, the entire message is dropped, the OM PSN_ERPS:MANDPDEF is incremented, the logs PSN202 AND PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause PARAM_CONTENTS_OUT_OF_ -RANGE_EC.

Optional parameters

The first byte of an optional parameter is the optional parameter tag, which identifies the optional parameter. The second and third bytes indicate the length of the optional parameter in bytes, starting with the beginning of the optional parameter ID, to the end of the optional parameter.

The contents of the optional parameter depends on the specific parameter and the customer using LOPER. The contents of the optional parameters are defined in PSN PARMS. Refer to the following diagram.

If a LOPER message is received with an unknown optional parameter ID, the unknown optional parameter is skipped and the decoding of the rest of the optional parameters in the message continues. If there is an error in the decoding of a known optional parameter, the optional parameter is dropped and the decoding process is ceased. As a result, the rest of the message is considered to be corrupted, the OM PSN_ERPS:OPPRMDEF is incremented, the logs PSN202 and PSN212 are produced, and the primitive, along with the optional parameters, are decoded before the error is processed.



- The OPTIONAL PARAMETER ID field consists of one byte and is the location which is used to represent the parameter. The values include:

```

0 00000000: Unknown
1 00000001: BearerCapability
2 00000010: BillingInfo
3 00000011: CallReferenceID
4 00000100: ControlInfo
5 00000101: Destinationtrunkgroup
6 00000110: DigitsCollection
7 00000111: DigitsCollected
8 00001000: DigitsOutpulsed
9 00001001: DigitstoOutputpulse
10 00001010: ErrorCause
11 00001011: FlowControlInfo

```

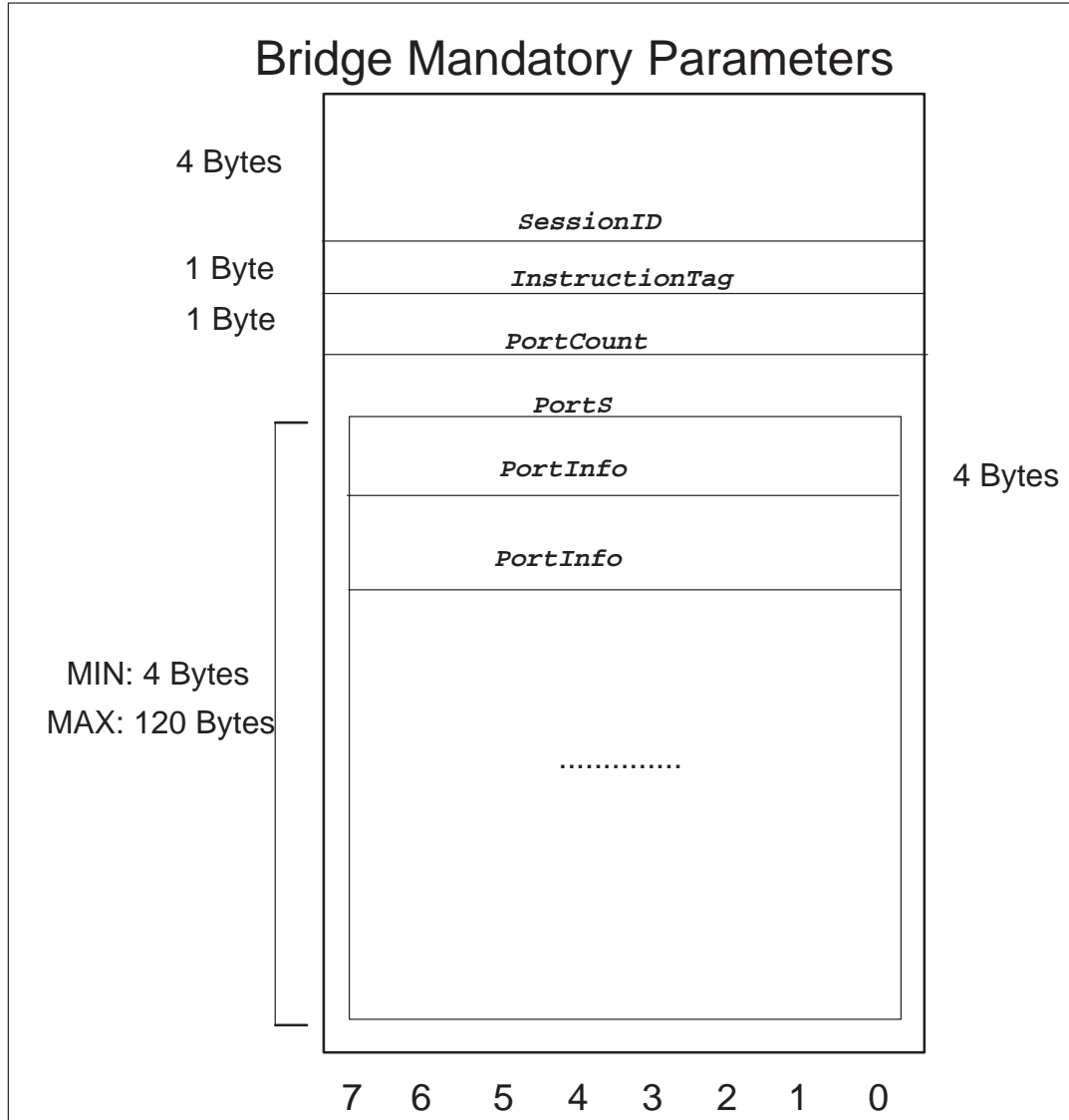
12 00001100: *FlowControlEncountered*
13 00001101: *InstructionID*
14 00001110: *Instructiontag*
15 00001111: *MessageInfo*
16 00010000: *MonitorMask*
17 00010001: *ParameterID*
18 00010010: *PortCount*
19 00010011: *PortInfo*
20 00010100: *PortServiceInfo*
21 00010101: *PortStatus*
22 00010110: *ResetReason*
23 00010111: *SessionID*
24 00011000: *SigInfo_Mask*
25 00011001: *SignalingInfo*
26 00011010: *Switch_ID*
27 00011011: *Time_of_day*
28 00011100: *Tone_Detected*
29 00011101 to 10000001: *SpareValues*
130 10000010: UCS *PointInCall*
131 10000011: UCS *STS*
10000100 to 11111111: *SpareValues*

PSN LOPER mandatory parameters for primitives

The following mandatory parameters are used to build primitives. Their description includes the length of the parameter as well as the type of each field in the parameter.

Bridge mandatory parameters

The mandatory parameters for a **Bridge** Primitive are as follows:



Parameter length: a minimum of 10 bytes and a maximum of 126 bytes

Parameter contents:

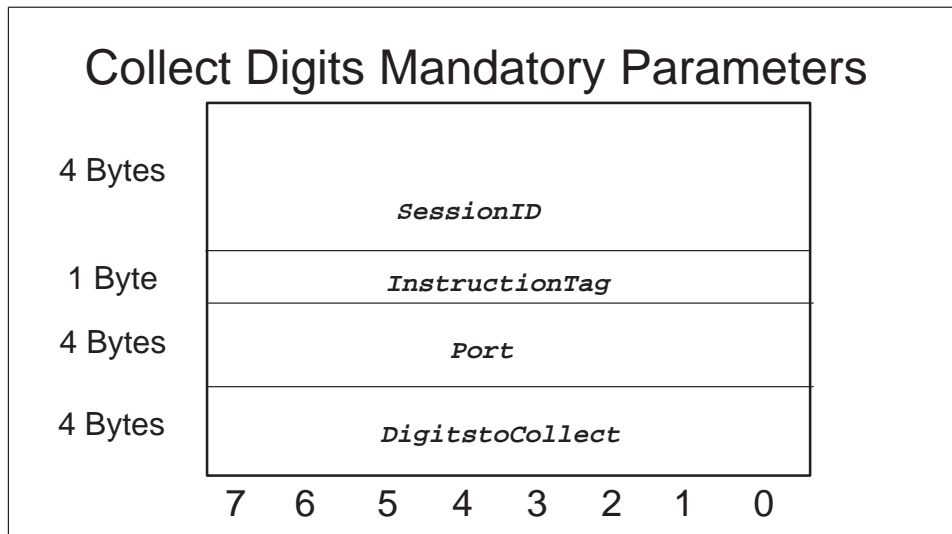
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *Instructiontag* parameter.
- The PORT COUNT field consists of one byte and is the location which includes the TYPE: Count of 1 to 30.
- The PORTS field consists of a minimum of four bytes and a maximum of 120 bytes. Also, the PORTS field consists of a minimum of one port to a maximum of 30 ports.

The Port TYPE: *PortInfo* parameter.

Collect Digits mandatory parameters

The mandatory parameters for a `Collect_Digits` primitive.



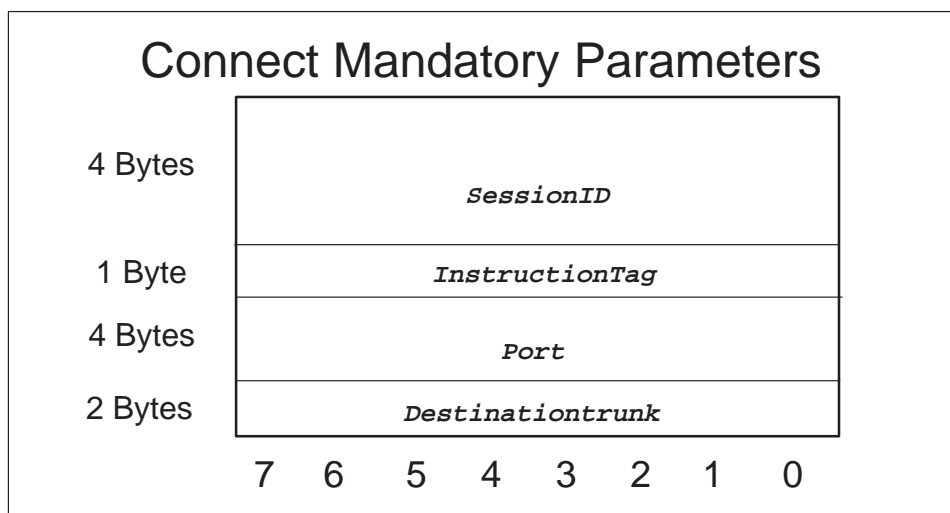
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitsCollection* parameter.

Connect mandatory parameters

The mandatory parameters for a **Connect** primitive are as follows:



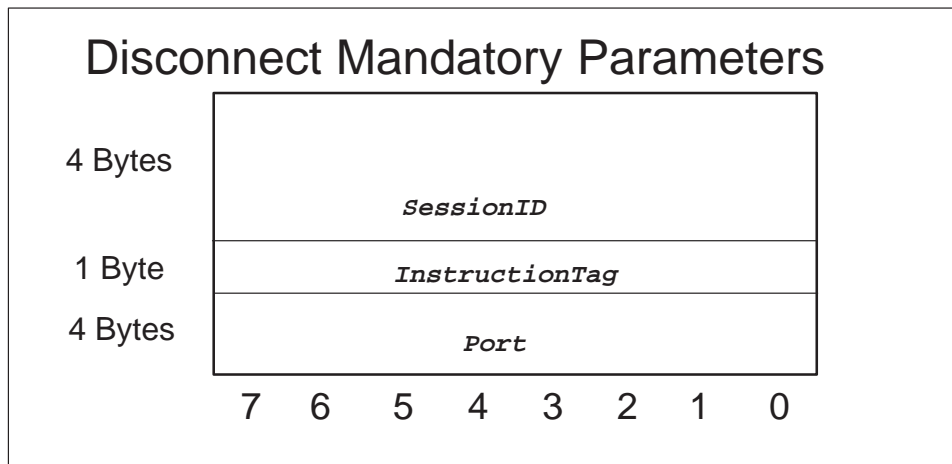
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DESTINATION TRUNK field consists of two bytes and is the location which includes the TYPE: *DestinationTrunkGroup* parameter.

Disconnect mandatory parameters

The mandatory parameters for a **Disconnect** primitive are as follows:



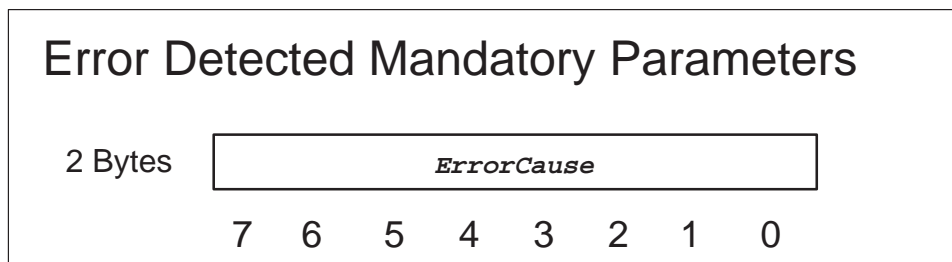
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Error Detected mandatory parameters

The mandatory parameters for a **Error_Detected** primitive are as follows:



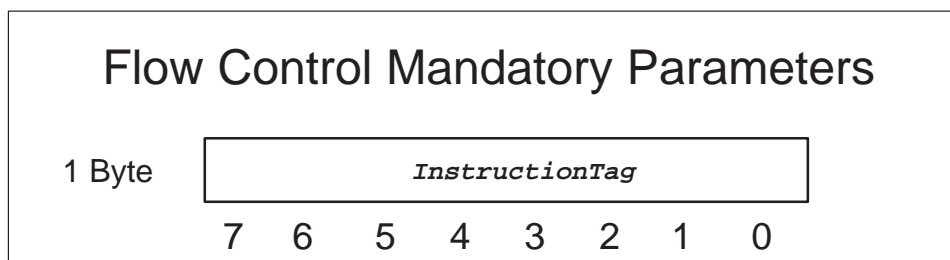
Parameter length: 2 bytes

Parameter contents:

- ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

Flow Control mandatory parameters

The mandatory parameters for a `Flow_Control` primitive are as follows:



Parameter length: 1 byte

Parameter contents:

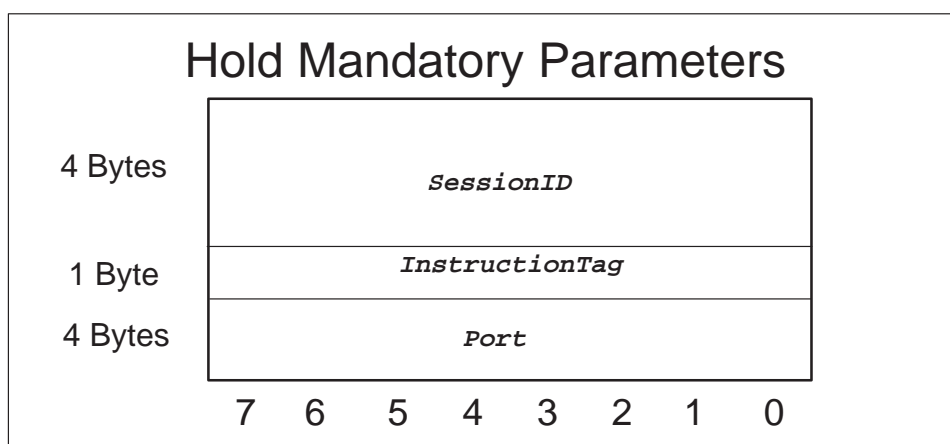
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Heartbeat mandatory parameters

The `Heartbeat` primitive does not contain any information in the primitive. A `Heartbeat` primitive message contains only what is mandatory for a LOPER primitive which is the primitive tag, the primitive length, and the optional pointer, totaling five bytes.

Hold mandatory parameters

The mandatory parameters for a `Hold` primitive are as follows:



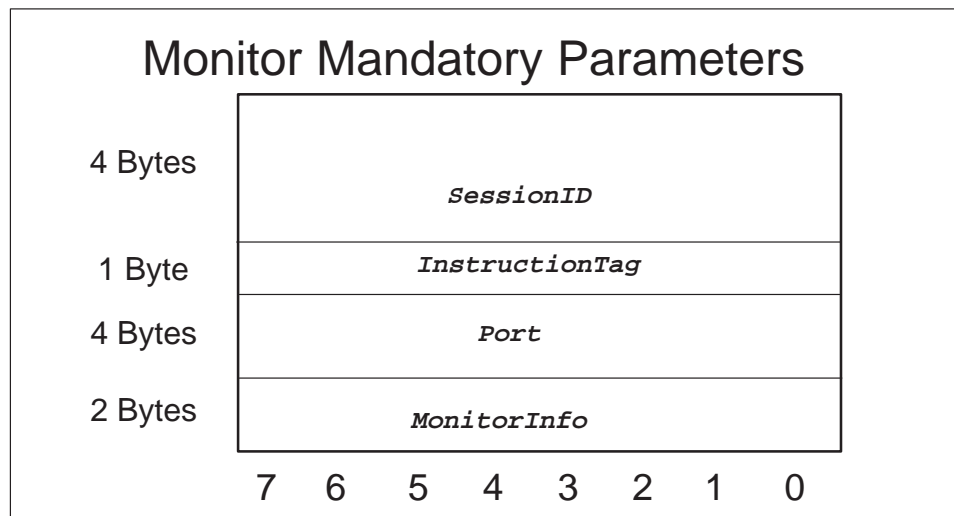
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Monitor mandatory parameters

The mandatory parameters for a **Monitor** primitive are as follows:



Parameter length: 11 bytes

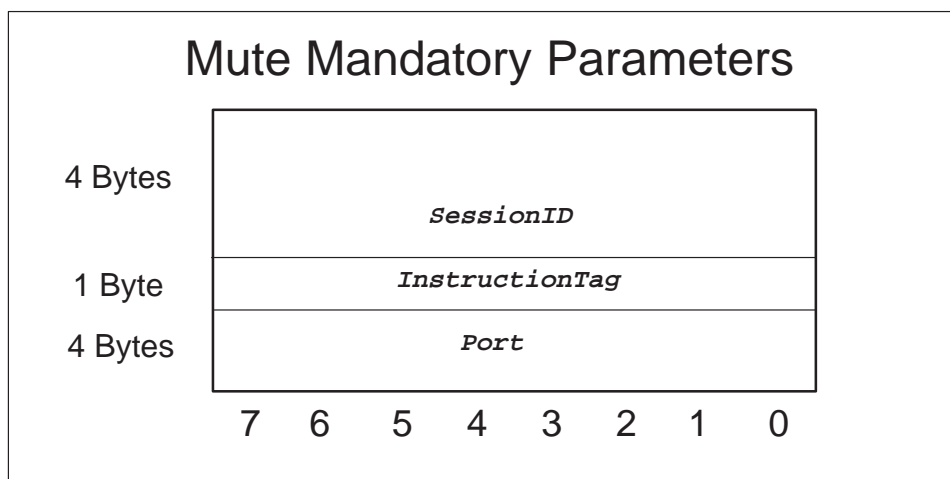
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

- The MONITOR INFORMATION field consists of two bytes and is the location which includes the TYPE: *MonitorMask* parameter.

Mute mandatory parameters

The mandatory parameters for a **Mute** primitive are as follows:



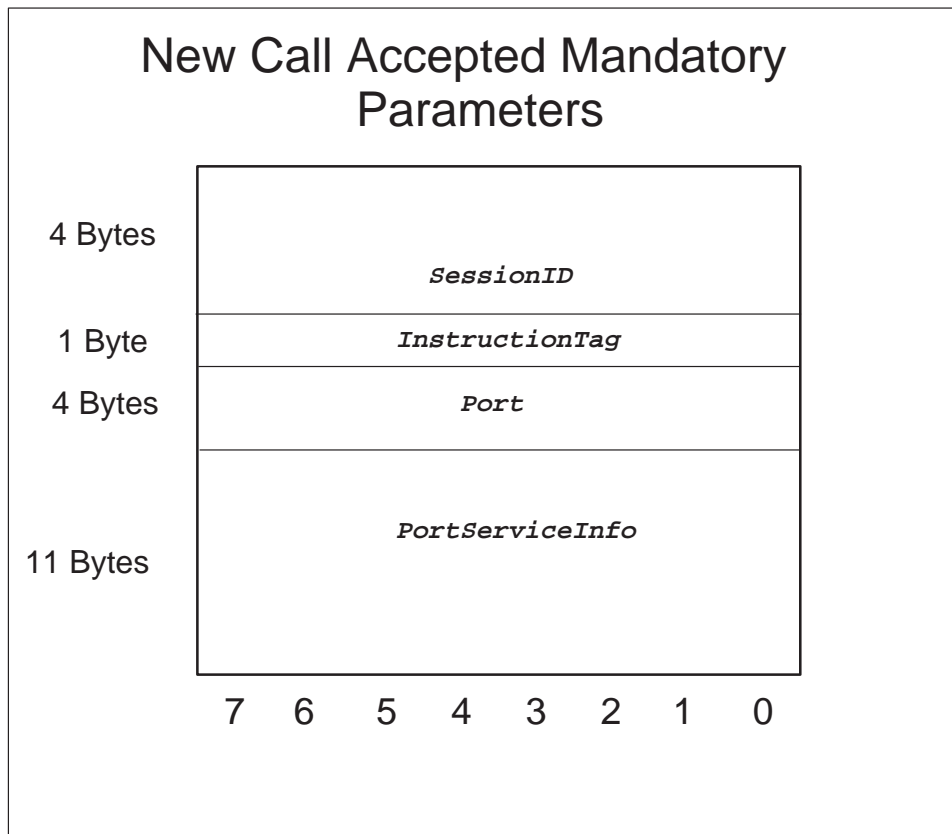
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call accepted mandatory parameters

The mandatory parameters for a **New_Call_Accepted** primitive are as follows:



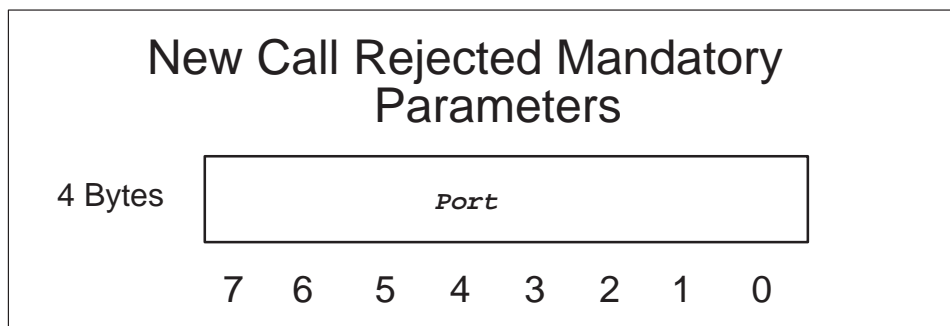
Parameter length: 20 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT SERVICE INFO field consists of 11 bytes and is the location which includes the TYPE: *PortServiceInfo* parameter.

New call rejected mandatory parameters

The mandatory parameters for a *New_Call_Rejected* primitive are as follows:



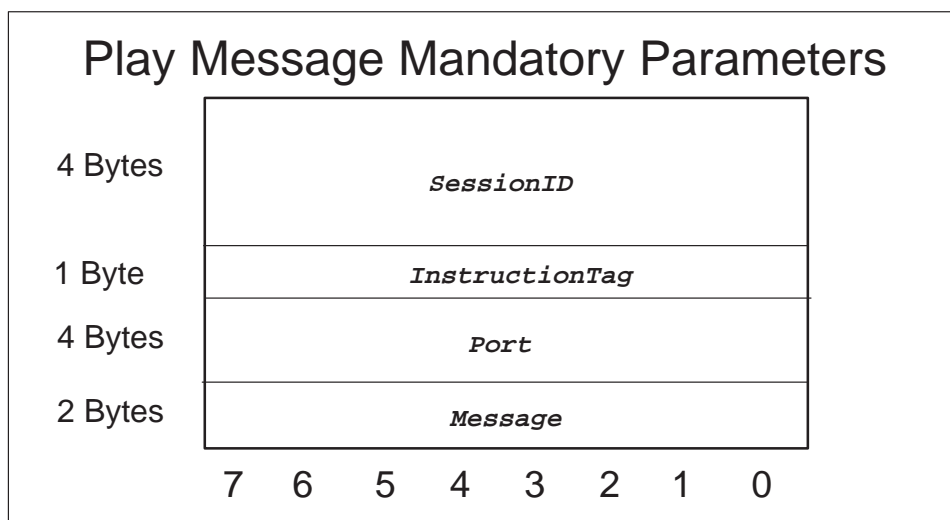
Parameter length: 4 bytes

Parameter contents:

- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Play message mandatory parameters

The mandatory parameters for a `Play_Message` primitive are as follows:



Parameter length: 11 bytes

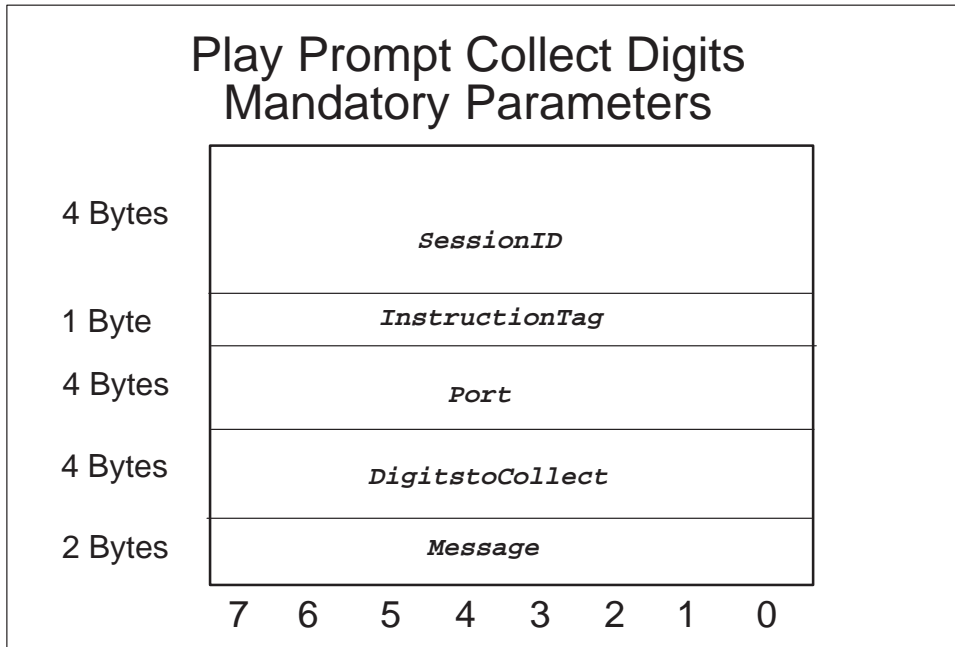
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Play prompt collect digits mandatory parameters

The mandatory parameters for a *Play_Prompt_Collect_Digits* primitive are as follows:



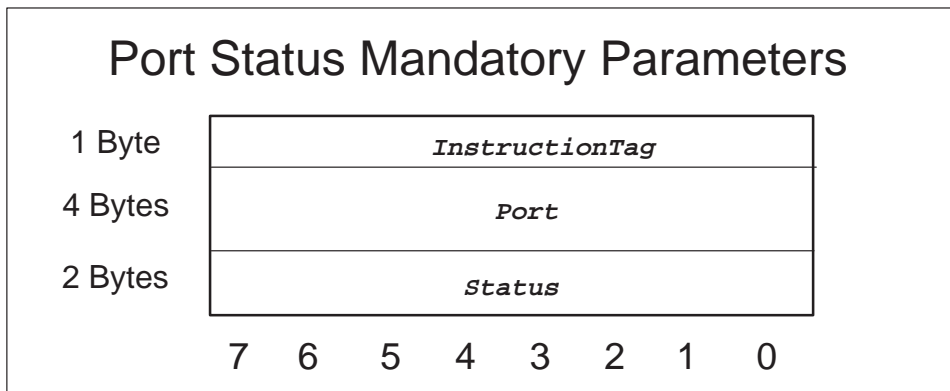
Parameter length: 15 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitsCollection* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` primitive are as follows:



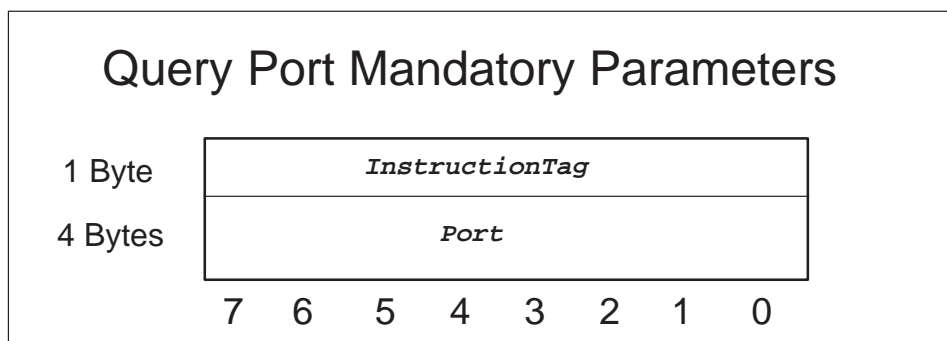
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` primitive are as follows:



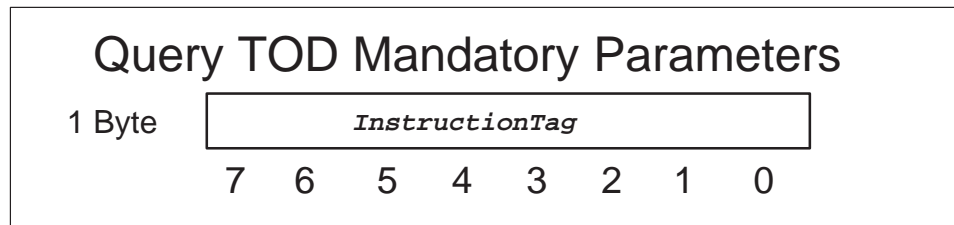
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Query time of day (TOD) mandatory parameters

The mandatory parameters for a *Query_Time_of_Day* primitive are as follows:



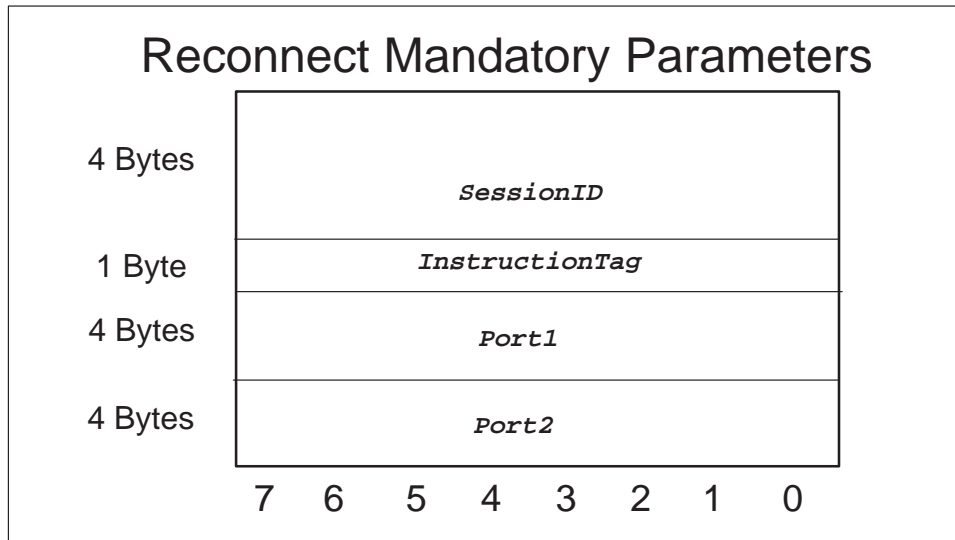
Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Reconnect mandatory parameters

The mandatory parameters for a **Reconnect** primitive are as follows:



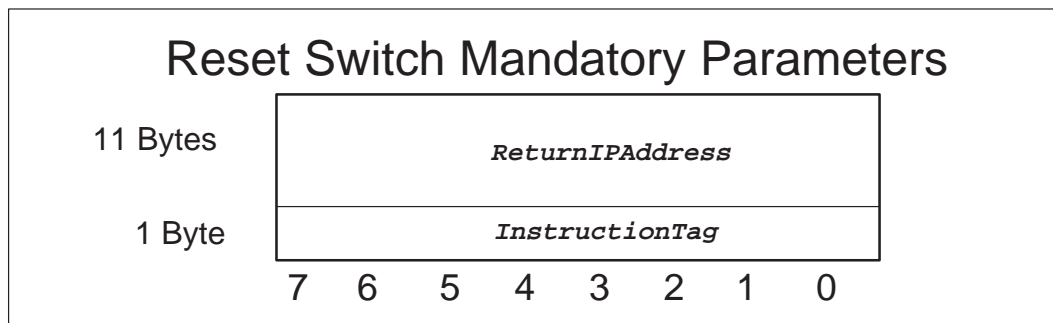
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT 1 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT 2 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Reset switch mandatory parameters

The mandatory parameters for a `Reset_Switch` primitive are as follows:



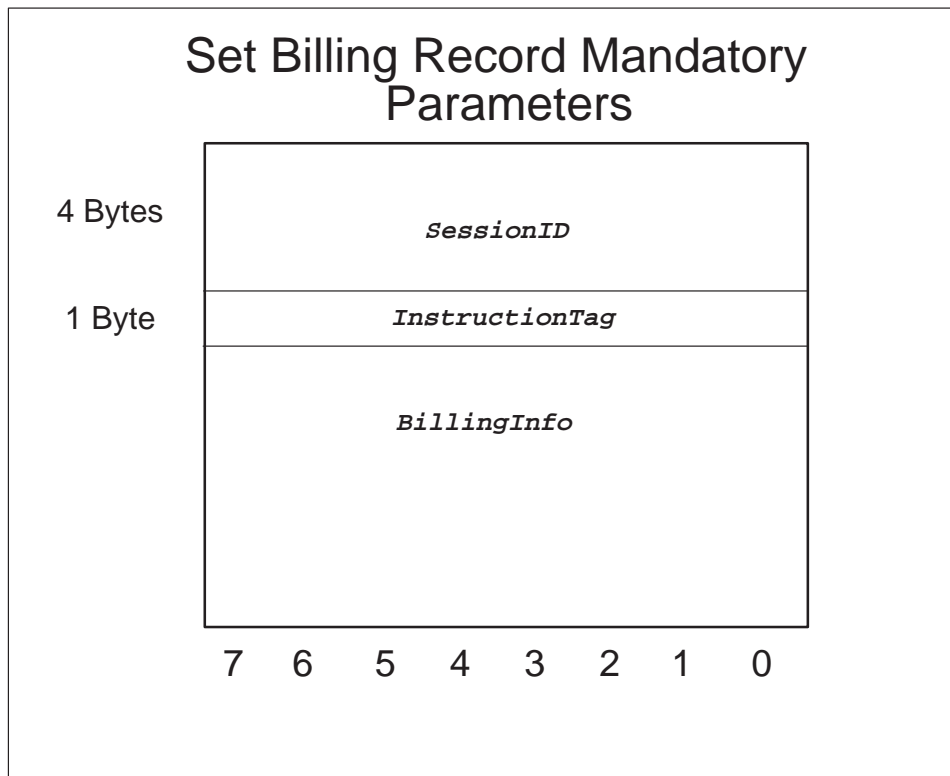
Parameter length: 12 bytes

Parameter contents:

- The RETURN IP ADDRESS field consists of 11 bytes and is the location which includes the TYPE: *PortService* parameter.
- Note:** There is one mandatory PSI parameter called the Return IP Address parameter. It is used to send the **Instruction_Completed**. There are 0 to 20 optional PSI parameters called the IP Address to Reset parameters which are used to do the actual resetting of calls.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Set billing record mandatory parameters

The mandatory parameters for a `Set_Billing_Record` primitive are as follows:



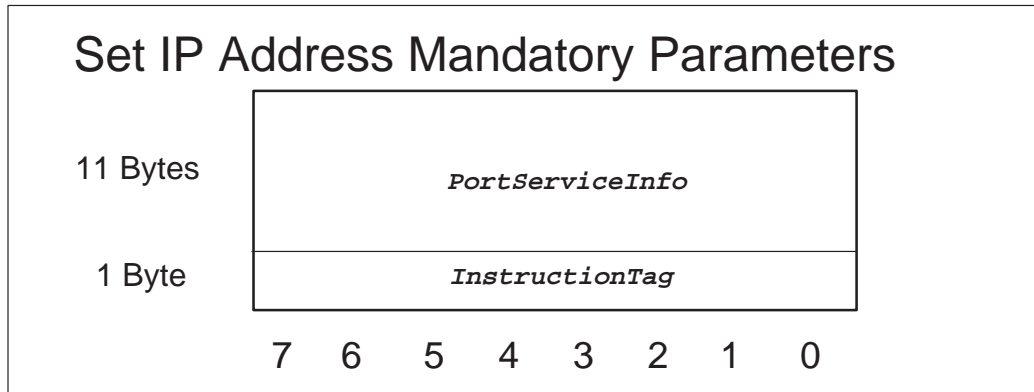
Parameter length: variable in size

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The BILLING INFORMATION is variable in size and is the location which includes the TYPE: *BillingInfo* parameter.

Set IP address mandatory parameters

The mandatory parameters for a `Set_IP_Address` primitive are as follows:



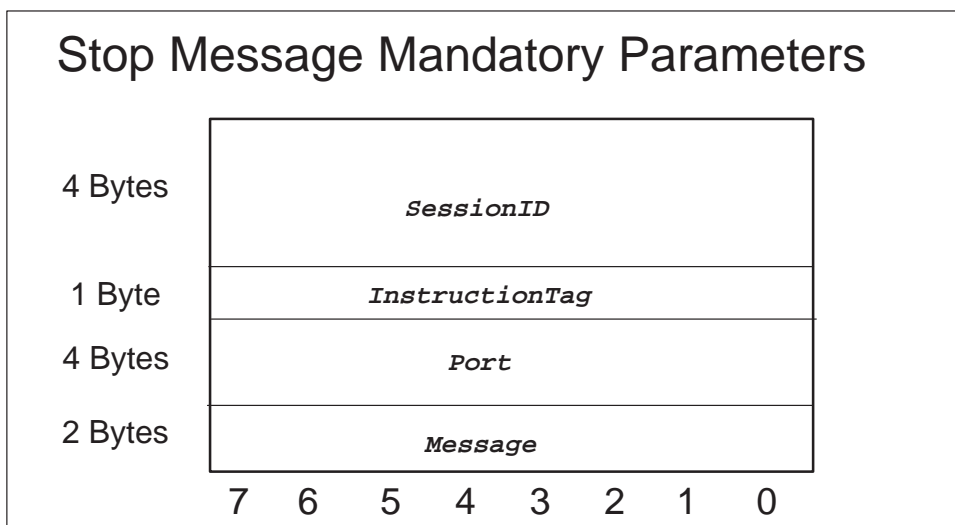
Parameter length: 12 bytes

Parameter contents:

- The PORT SERVICE INFO field consists of 11 bytes and is the location which includes the TYPE: `PortServiceInfo` parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: `InstructionTagID` parameter.

Stop message mandatory parameters

The mandatory parameters for a `Stop_Message` primitive are as follows:



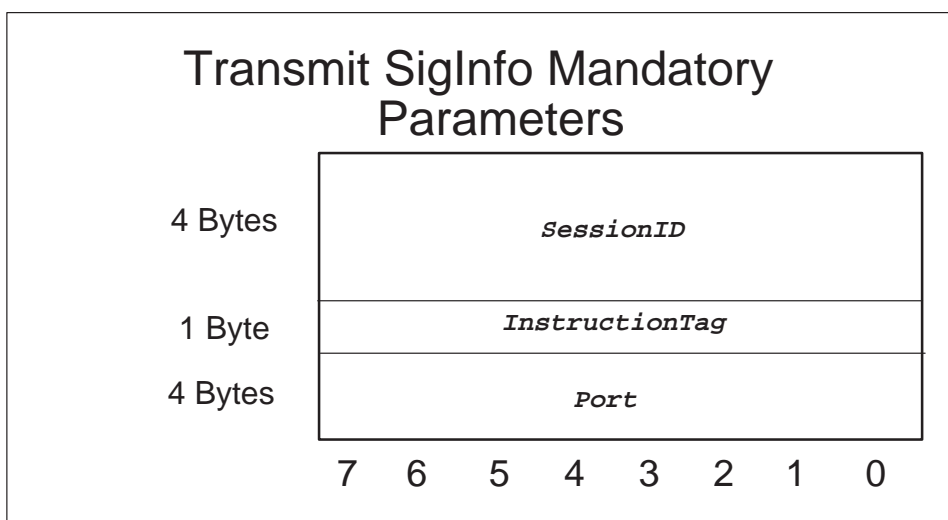
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Transmit siginfo mandatory parameters

The mandatory parameters for a `Transmit_Siginfo` primitive are as follows:



Parameter length: 9 bytes

Parameter contents:

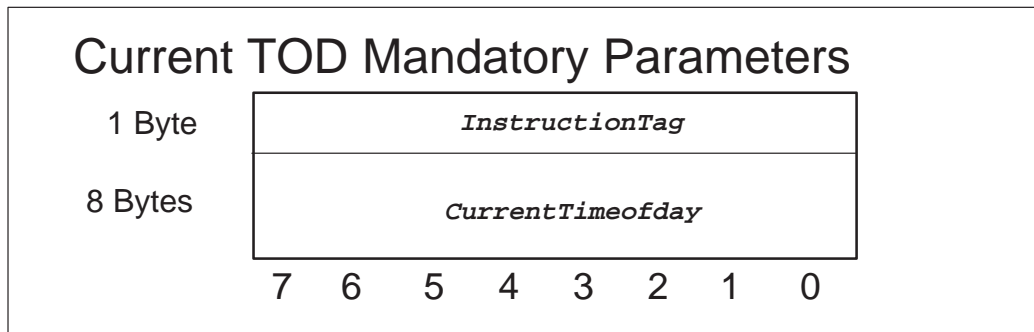
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

PSN LOPER mandatory parameters for events

The following mandatory parameters are used to build event notifications. Their description includes the parameter length as well as the type of each field in the parameter.

Current time of day (TOD) mandatory parameters

The mandatory parameters for a `Current_Time_of_Day` event are as follows:



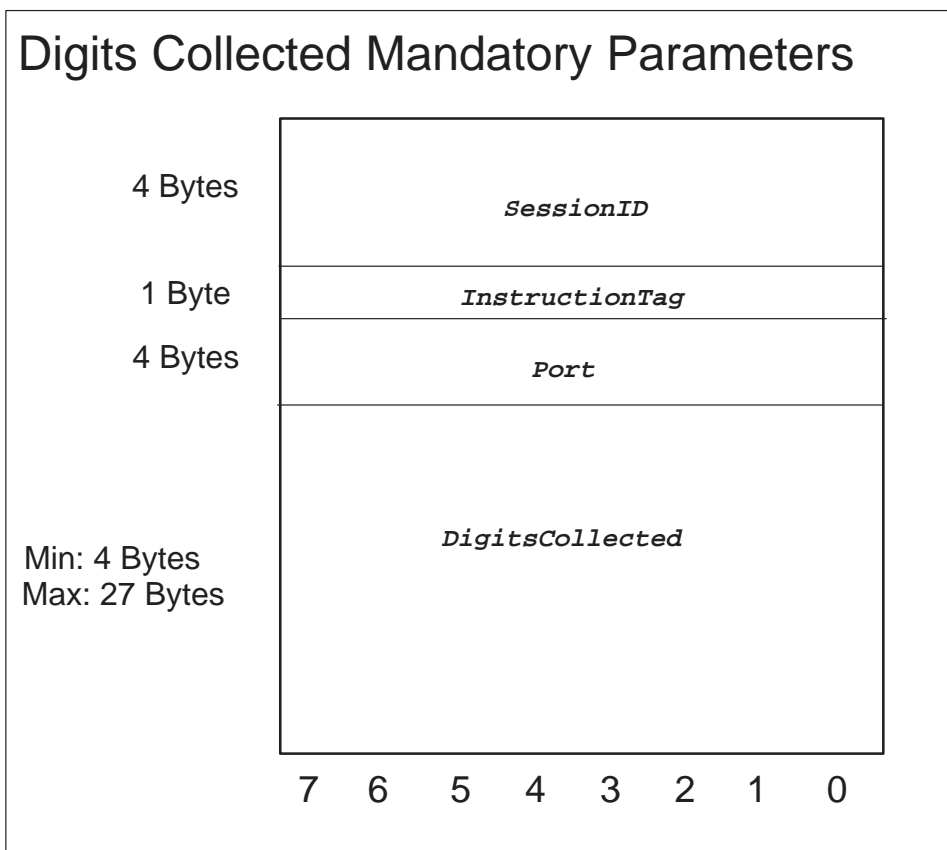
Parameter length: 9 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The CURRENT TIME OF DAY field consists of eight byte and is the location which includes the TYPE: *TimeOfDay* parameter.

Digits collected mandatory parameters

The mandatory parameters for a `Digits_Collected` event are as follows:



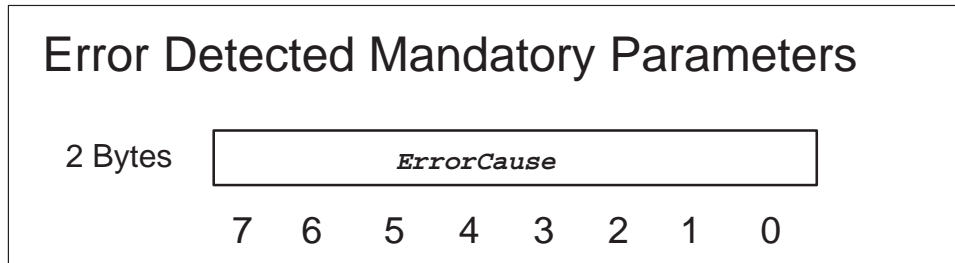
Parameter length: a minimum of 13 bytes and a maximum of 36 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS COLLECTED field consists of a minimum of four bytes and a maximum of 27 bytes and is the location which includes the TYPE: *DigitsCollected* parameter.

Error detected mandatory parameters

The mandatory parameters for a **Error_Detected** event are as follows:



Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

In service mandatory parameters

The mandatory parameters for a **In_Service** Event are as follows:



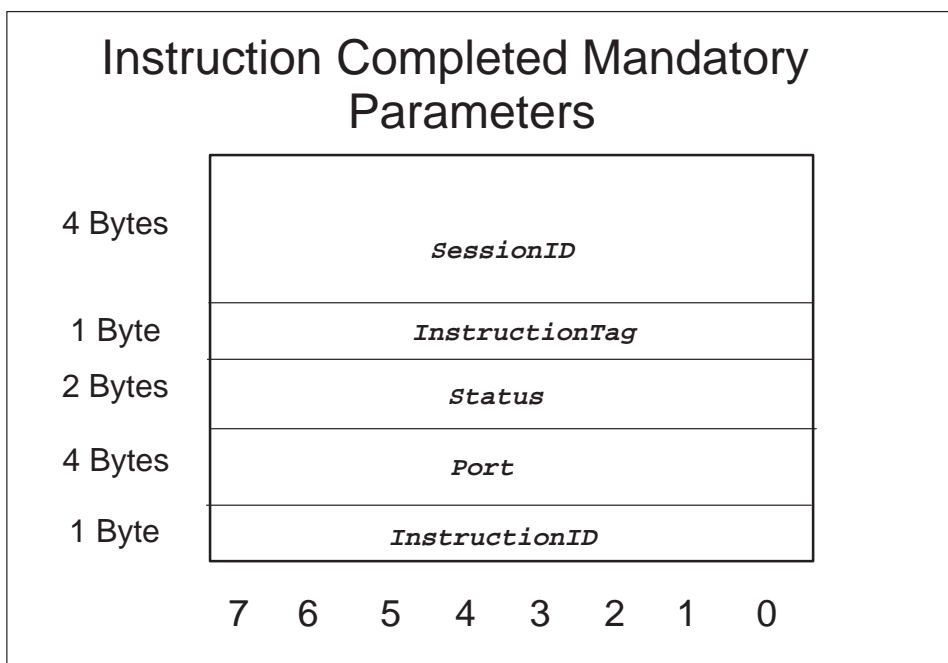
Parameter length: 1 byte

Parameter contents:

- The RESET REASON field consists of one byte and is the location which includes the TYPE: *ResetReason* Parameter

Instruction completed mandatory parameters

The mandatory parameters for a `Instruction_Completed` event are as follows:



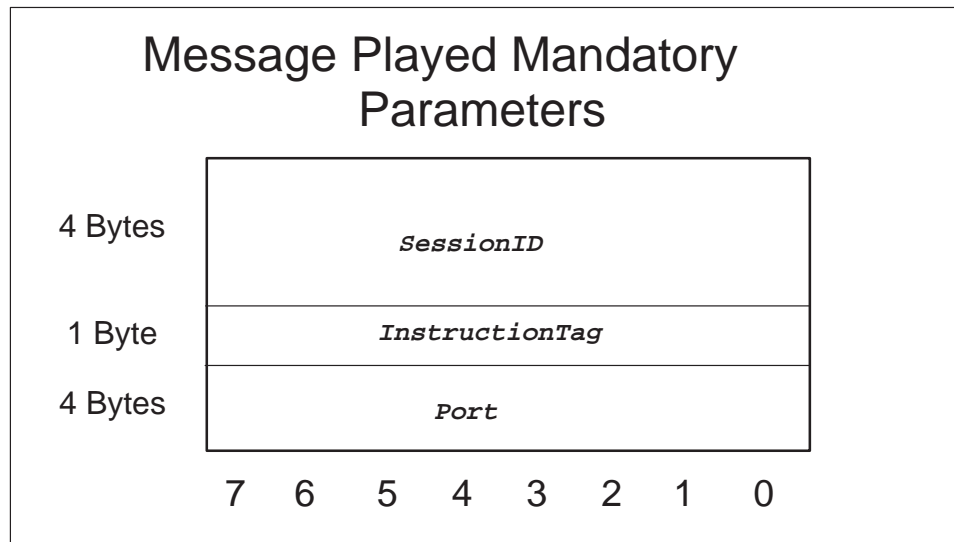
Parameter length: 12 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The INSTRUCTION ID field consists of one byte and is the location which includes the TYPE: *InstructionID* parameter.

Message played mandatory parameters

The mandatory parameters for a **Message_Played** event are as follows:



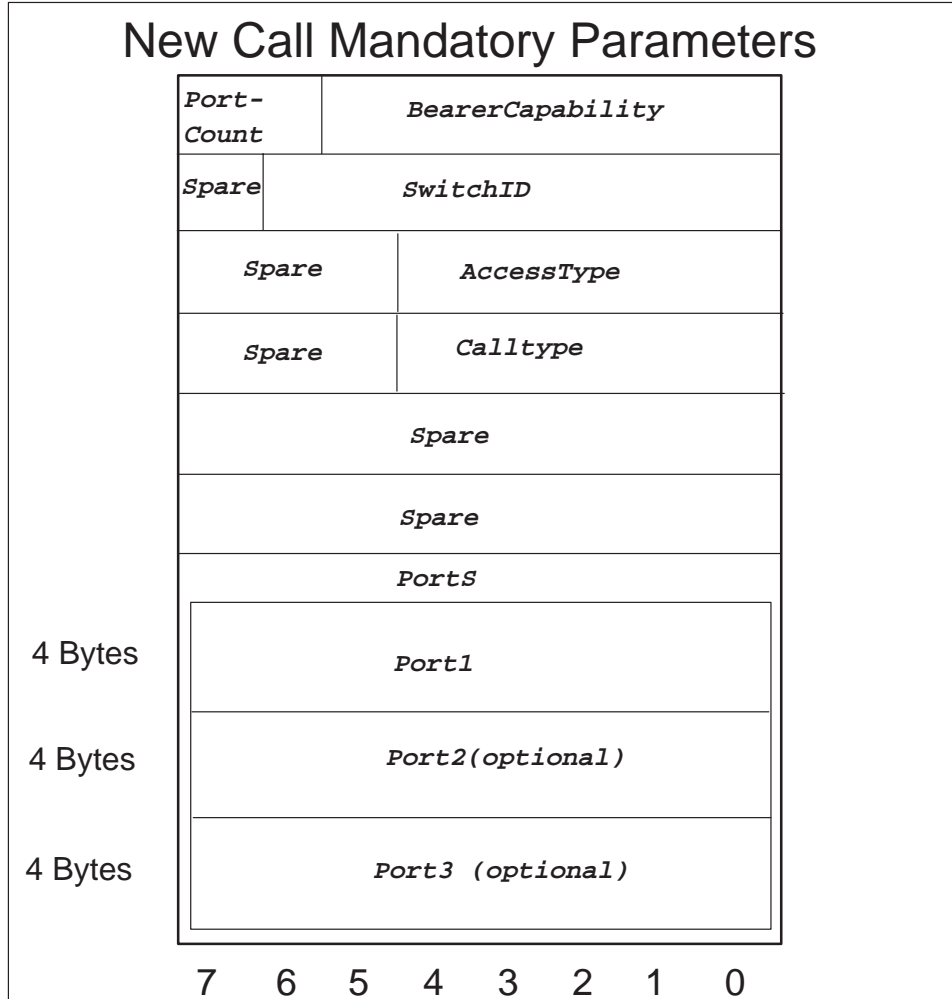
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call mandatory parameters

The mandatory parameters for a *New_Call* event are as follows:



Parameter length: a minimum of 10 bytes and a maximum of 18 bytes

Parameter contents:

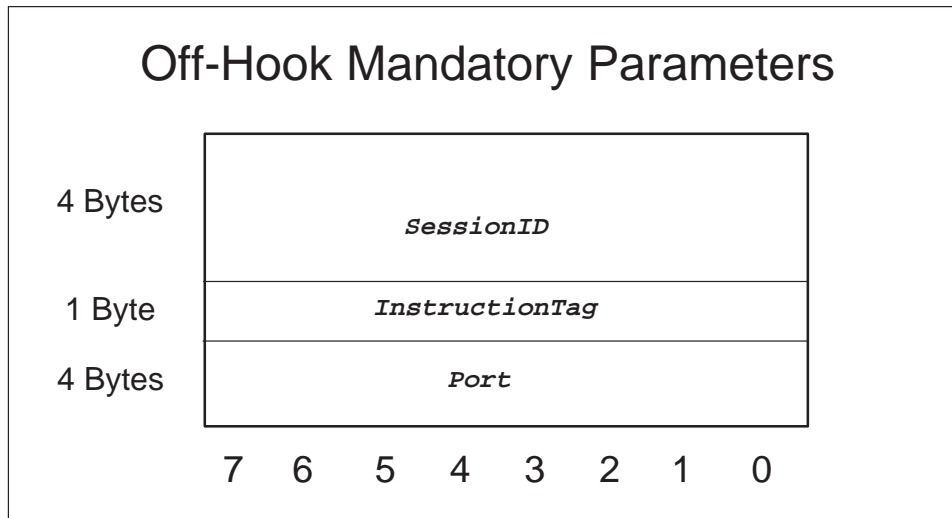
- The BEARER CAPABILITY field consists of six bits and is the location which includes the TYPE: *BearerCapability* parameter.
- The PORT COUNT field consists of two bits. The values include: 0 to 3.
- The SWITCH ID field consists of seven bits and is the location which includes the TYPE: *switchID* parameter.
- The ACCESS TYPE field consists of five bits and is the location which includes the TYPE: *AccessType* parameter.

- The CALL TYPE field consists of five bits and is the location which includes the TYPE: *CallType* Parameter
- PORTS: Ports consists of a minimum of 1 port to a maximum of 3 ports.

Port TYPE: *PortInfo* parameter.

Off-hook mandatory parameters

The mandatory parameters for an *off_Hook* event are as follows:



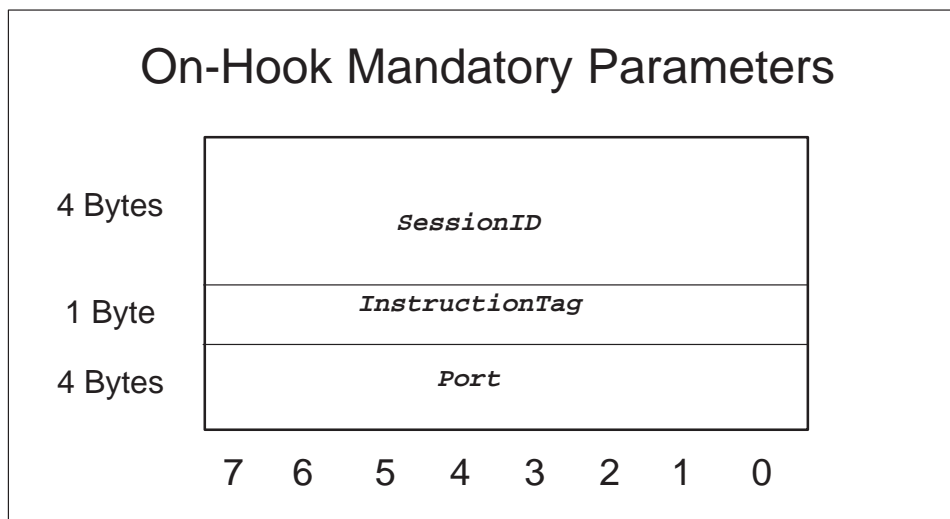
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

On-hook mandatory parameters

The mandatory parameters for an `On_Hook` event are as follows:



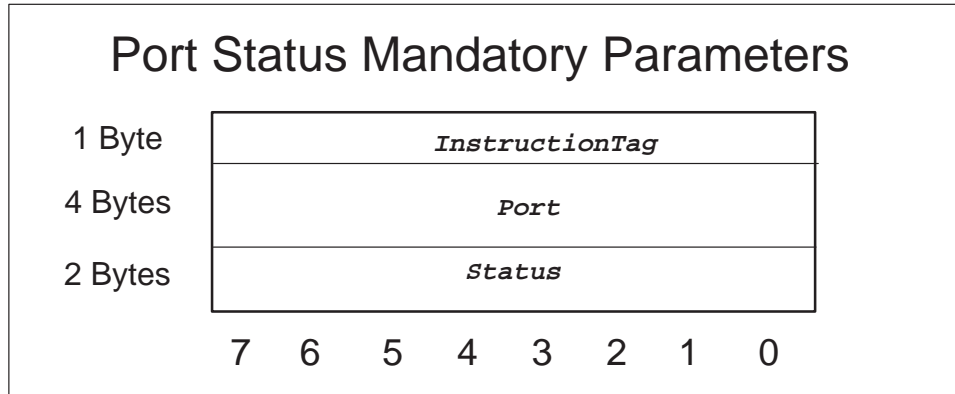
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` event are as follows:



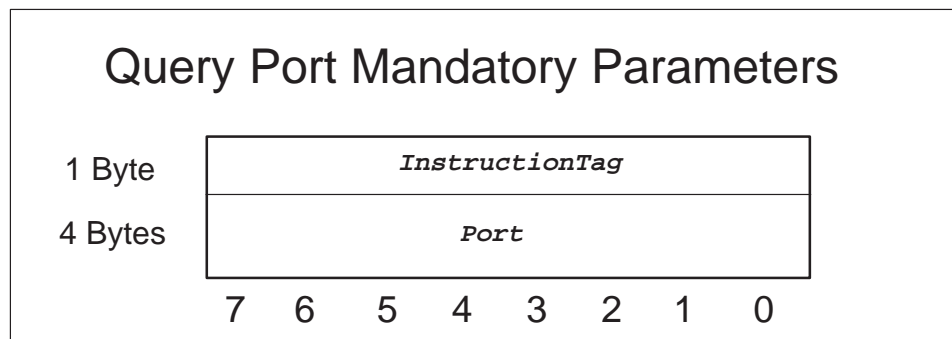
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` event are as follows:



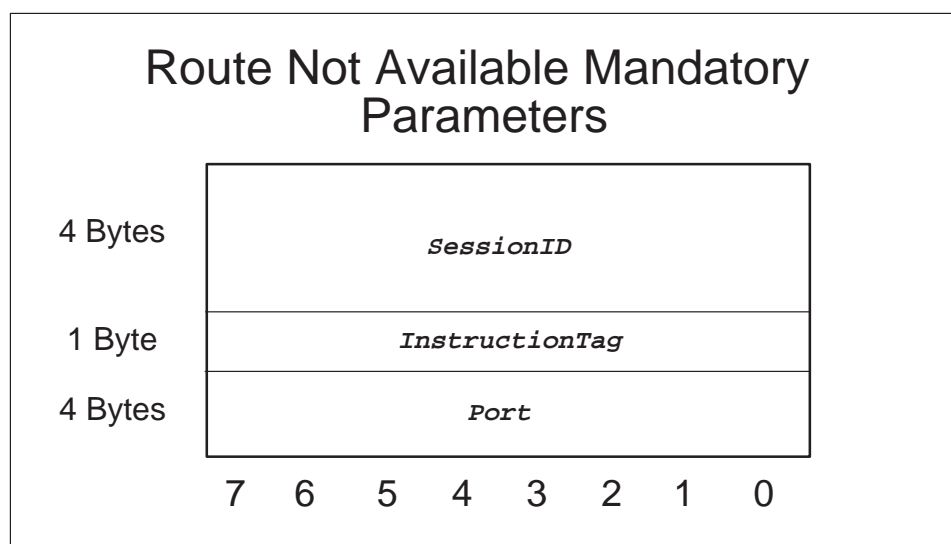
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route not available mandatory parameters

The mandatory parameters for an `Route_Not_Available` event are as follows:



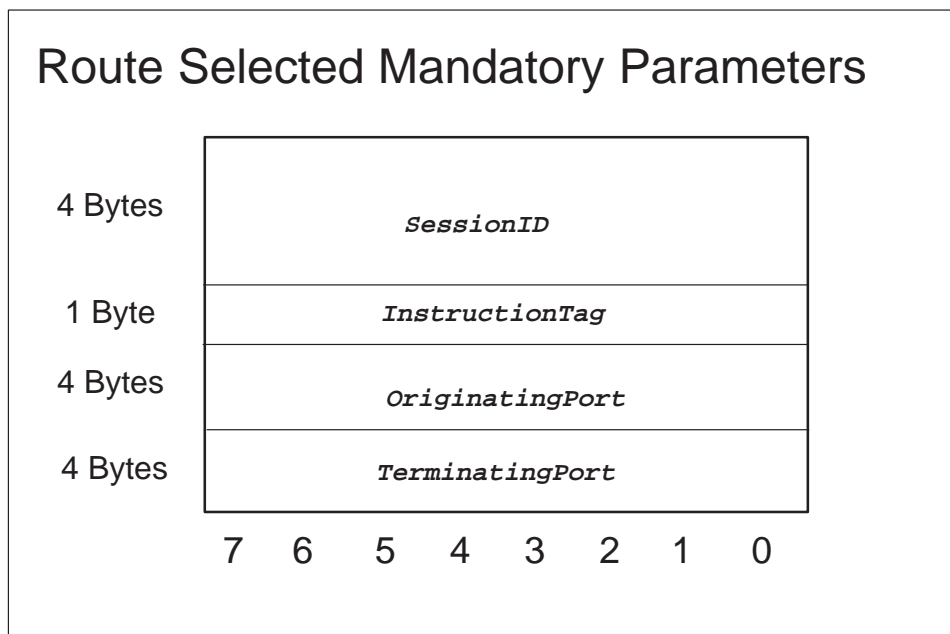
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route selected mandatory parameters

The mandatory parameters for an `Route_selected` event are as follows:



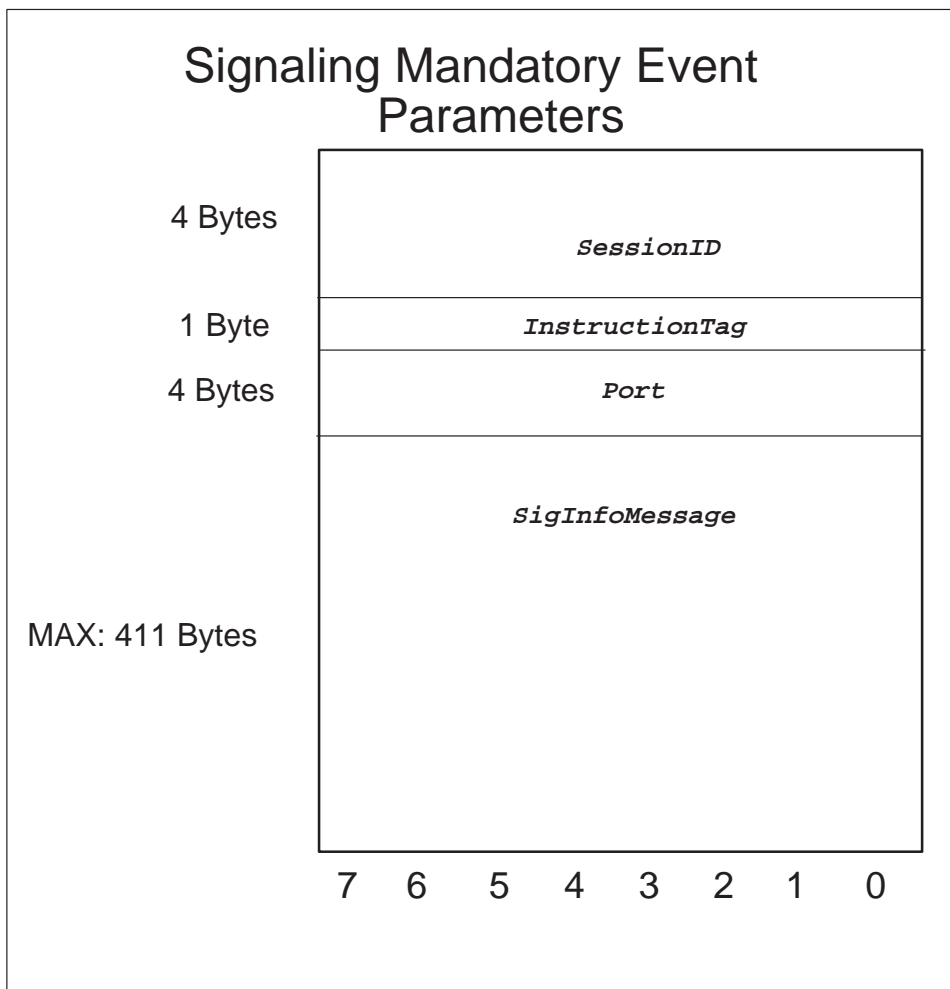
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The ORIGINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TERMINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Signaling event mandatory parameters

The mandatory parameters for a **signaling** event are as follows:



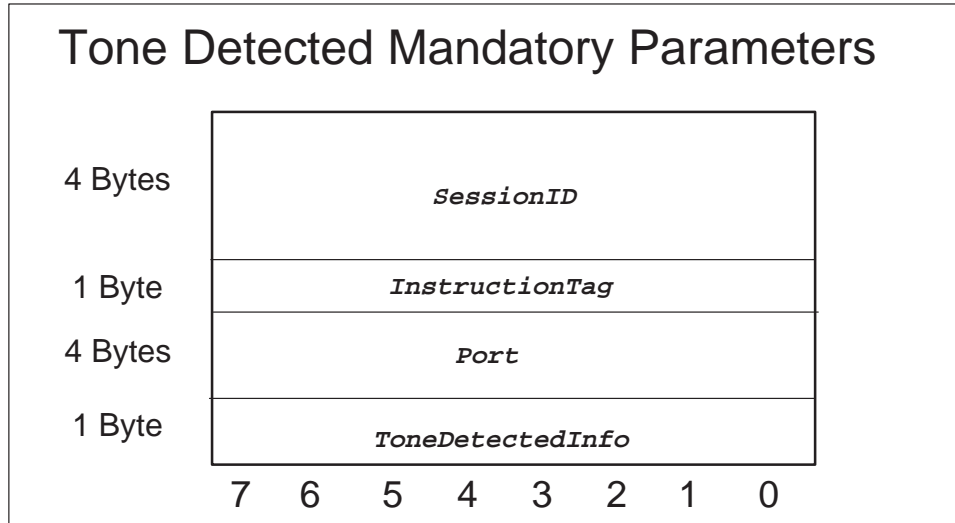
Parameter length: maximum of 420 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of one bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The SIGINFO MSG field consists of a maximum of 411 bytes and is the location which includes the TYPE: *SignalingInfo* parameter.

Tone detected mandatory parameters

The mandatory parameters for a `Tone_Detected` event are as follows:



Parameter length: 10 bytes

Parameter contents:

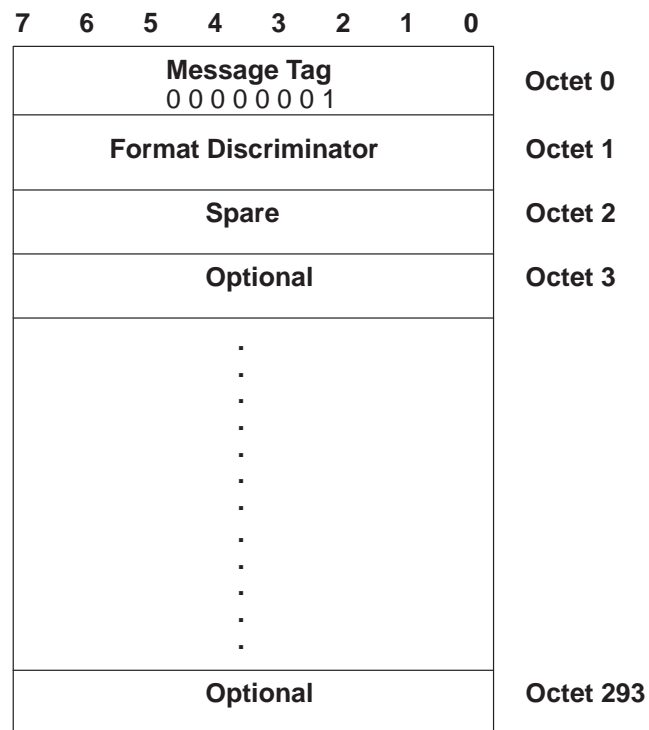
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TONE DETECTED INFORMATION field consists of one byte and is the location which includes the TYPE: *ToneDetected* parameter.

PRI messages

Alerting message

The following diagram shows the layout of the Alerting message.

Alerting message



Alert mandatory parameters

message tag (Octet 0)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 1

%% 1– Alert message tag

Format discriminator (Octet 1)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

SCP_C_FD_V1

Spare (Octet 2)

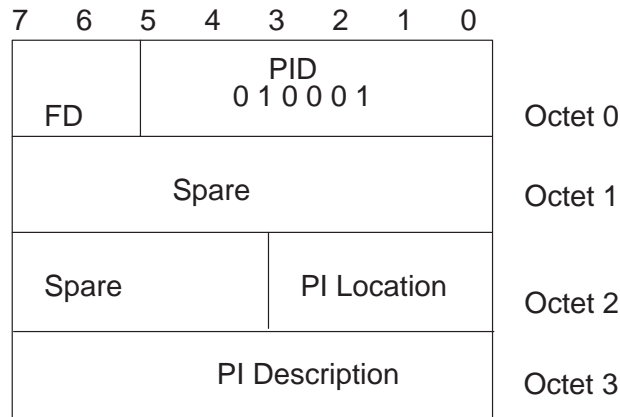
The value of this octet is 0.

Alert optional parameters (Octet 3–293)

The Alert message support the following optional parameters: *ProgressIndicator* and *User to User Information*.

Note: Each new optional parameter must start a new word. If the previous optional parameter did not fill out the word, meaning that there is a byte missing to fill out the word, it then pads the reset of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the word.

Progress indicator



PID (Octet 0)

Bits

5 4 3 2 1 0

0 1 0 0 0 1

% 17 PI_PID

Format discriminator (Octet 0)

Bits

7 6

0 0

OPTIONAL_PARM_FD_V1

PI location (Octet 2)

Bits

3 2 1 0

0 0 0 1	% 1	LOCATION_USER
0 0 1 0	% 2	LOCATION_PVT_NETWORK_SERV_LOCAL
0 0 1 1	% 3	LOCATION_PUB_NETWORK_SERV_LOCAL
0 1 0 0	% 4	LOCATION_TRANSIT_NETWORK
0 1 0 1	% 5	LOCATION_PVT_NETWORK_SERV_REMOTE
0 1 1 0	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE
0 1 1 1	% 7	LOCATION_INTERNATIONAL_NETWORK
1 0 0 0	% 8	LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1	% 9	LOCATION_LOCAL_INTERFACE_
		CONTR_BY_THIS_SIG_LINK

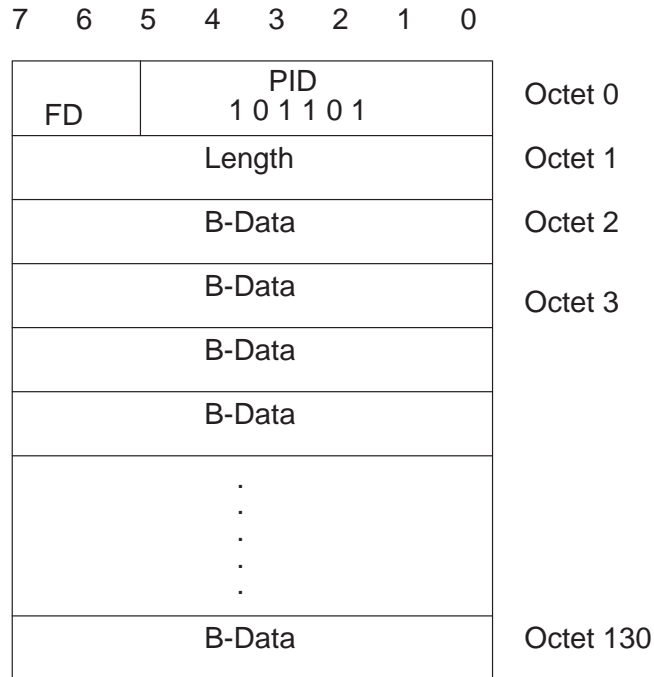
PI description (Octet 3)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 1	% 1	PROG_IND_NOT_END_TO_END_ISDN
0 0 0 0 0 0 1 0	% 2	PROG_IND_IN_BAND_INFO_AVAIL
0 0 0 0 0 0 1 1	% 3	PROG_IND_DEST_NOT_RESP
0 0 0 0 0 1 0 0	% 4	Not Used
0 0 0 0 0 1 0 1	% 5	Not Used
0 0 0 0 0 1 1 0	% 6	Not Used
0 0 0 0 0 1 1 1	% 7	PROG_IND_DEST_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 0	% 8	PROG_IND_ORIG_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 1	% 9	PROG_IND_CALL_RETURNED_TO_ISDN
0 0 0 0 1 0 1 0	% 10	Not Used
0 0 0 0 1 0 1 1	% 11	PROG_IND_DELAY_RESP_CALLED_INTERFACE
0 0 0 0 1 1 0 0	% 12	PROG_IND_OVLP_NOT_END_TO_END_ISDN

User to user information



PID (Octet 0)

Bits	
5 4 3 2 1 0	
1 0 1 1 0 1	% 45 UUI_PID

Format discriminator (Octet 0)

Bits	
7 6	
0 0	Optional_Parm_FD_V1

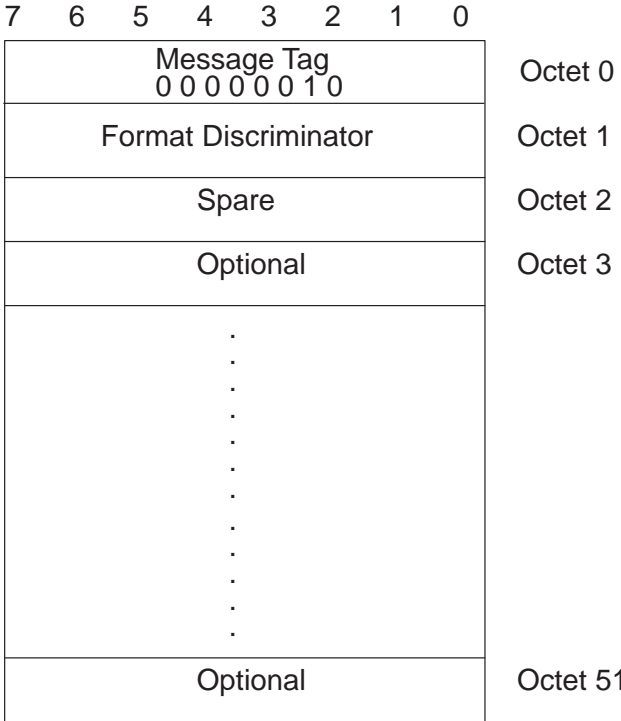
Parameter length (Octet 1)

This parameter is defined as one byte, but ranges from 1 to 129.

B-Data (Octet 2–130)

The maximum size of B-Data is 129 bytes long.

Call proceeding



Call proceeding mandatory parameters

message tag(Octet 0)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 1 0 %% 2-Call_Proceeding message
tag

Format discriminator (Octet 1)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 SCP_C_FD_V1

Spare (Octet 2)

The value of this octet is 0.

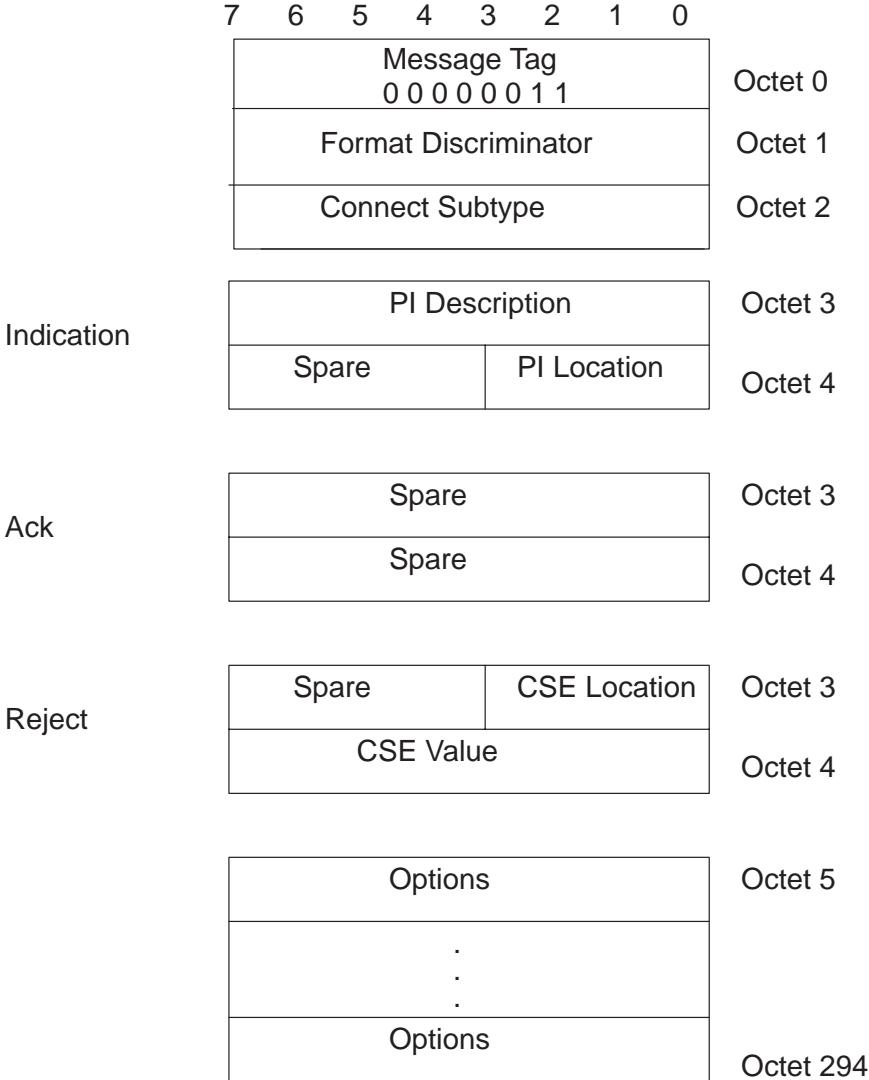
Call proceeding optional parameters

There are no optional parameters currently supported in the `Call_Proceeding` message type by this feature.

Connect message

The following diagram shows the layout of the Connect message.

Connect message



**Connect mandatory message parameters
message tag (Octet 0)**

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 1 1

%% 3 – Connect message tag

Format discriminator (Octet 1)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 SCP_C_FD_V1

Connect subtype (Octet 2)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0 INDICATION
0 0 0 0 0 0 0 1 ACK
0 0 0 0 0 0 1 0 REJECT

Depending on the Connect subtype, Octet 3 and 4 look like the following:

PI description (Octet 3)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 1 % 1 PROG_IND_NOT_END_TO_END_ISDN
0 0 0 0 0 0 1 0 % 2 PROG_IND_IN_BAND_INFO_AVAIL
0 0 0 0 0 0 1 1 % 3 PROG_IND_DEST_NOT_RESP
0 0 0 0 0 1 0 0 % 4 Not Used
0 0 0 0 0 1 0 1 % 5 Not Used
0 0 0 0 0 1 1 0 % 6 Not Used
0 0 0 0 0 1 1 1 % 7 PROG_IND_DEST_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 0 % 8 PROG_IND_ORIG_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 1 % 9 PROG_IND_CALL_RETURNED_TO_ISDN
0 0 0 0 1 0 1 0 % 10 Not Used
0 0 0 0 1 0 1 1 % 11 PROG_IND_DELAY_RESP_CALLED_INTERFACE
0 0 0 0 1 1 0 0 % 12 PROG_IND_OVLP_NOT_END_TO_END_ISDN

PI location (Octet 4)

Bits
3 2 1 0

0 0 0 1 % 1 LOCATION_USER
0 0 1 0 % 2 LOCATION_PVT_NETWORK_SERV_LOCAL
0 0 1 1 % 3 LOCATION_PUB_NETWORK_SERV_LOCAL
0 1 0 0 % 4 LOCATION_TRANSIT_NETWORK
0 1 0 1 % 5 LOCATION_PVT_NETWORK_SERV_REMOTE

0 1 1 0	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE
0 1 1 1	% 7	LOCATION_INTERNATIONAL_NETWORK
1 0 0 0	% 8	LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1	% 9	LOCATION_LOCAL_INTERFACE_ CONTR_BY_THIS_SIG_LINK

Ack

Octet 6 and 7 are spares

Reject**CSE location (Octet 3)**

Bits

3 2 1 0

0 0 0 1	% 1	LOCATION_USER
0 0 1 0	% 2	LOCATION_PVT_NETWORK_SERV_LOCAL
0 0 1 1	% 3	LOCATION_PUB_NETWORK_SERV_LOCAL
0 1 0 0	% 4	LOCATION_TRANSIT_NETWORK
0 1 0 1	% 5	LOCATION_PVT_NETWORK_SERV_REMOTE
0 1 1 0	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE
0 1 1 1	% 7	LOCATION_INTERNATIONAL_NETWORK
1 0 0 0	% 8	LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1	% 9	LOCATION_LOCAL_INTERFACE_ CONTRA_BY_THIS_SIG_LINK

CSE value (Octet 4)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 1	% 1	ISDN_UNASSIGNED_NUMBER_CSE
0 0 0 0 0 0 1 0	% 2	ISDN_USER_BUSY
0 0 0 0 0 0 1 1	% #3	ISDN_INCOMPLETE_NUMBER
0 0 0 0 0 1 0 0	% #4	ISDN_NORMAL_UNSPECIFIED
0 0 0 0 0 1 0 1	% #5	ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
0 0 0 0 0 1 1 0	% #6	ISDN_SWITCH_EQUIPMENT_CONGESTION
0 0 0 0 0 1 1 1	% #7	ISDN_RESOURCE_UNAVAILABLE
0 0 0 0 1 0 0 0	% #8	ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
0 0 0 0 1 0 0 1	% #9	ISDN_DEST_MISSING
0 0 0 0 1 0 1 0	% #A	ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE

15-10 PRI messages

00001011	% #B	ISDN_PROTOCOL_ERROR_UNSPECIFIED
00001100	% #C	ISDN_NO_ROUTE_TO_SPECIFIED_TRANSIT_NETWORK
00001101	% #D	ISDN_NORMAL_CALL_CLEARING
00001110	% #E	ISDN_NO_USER_RESPONDING
00001111	% #F	ISDN_CALL_REJECTED
00010000	% #10	ISDN_NUMBER_CHANGED
00010001	% #11	ISDN_FACILITY_REJECTED
00010010	% #12	ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13	ISDN_USER_INFORMATION_DISCARDED
00010100	% #14	ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15	ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110	% #16	ISDN_INCOMING_CALLS_BARRED
00010111	% #17	ISDN_BC_NOT_AUTHORIZED
00011000	% #18	ISDN_BC_NOT_IMPLEMENTED
00011001	% #19	ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A	ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B	ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C	ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D	ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E	ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F	ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20	ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21	ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22	ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23	ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24	ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25	ISDN_DST_OUT_OF_SERV
00100110	% #26	ISDN_TEMP_FAIL
00100111	% #27	ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28	– Not Used
00101001	% #29	ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A	ISDN_PREEMPT
00101011	% #2B	ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C	ISDN_PREEMPT_REUSE
00101101	% #2D	ISDN_VACANT_CODE
00101110	% #2E	ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F	ISDN_PRFX_1_NOT_DIALED
00110000	% #30	ISDN_SERV_OPT_VIOL
00110001	% #31	– Not Used
00110010	% #32	ISDN_NORTE_TO_DSTN
00110011	% #33	ISDN_NO_ANS_FROM_USER

00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A – Not Used
00111011	% #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100	% #3C ISDN_CALL_IDENTITY_IN_USE
00111101	% #3D ISDN_NOT_CALL_SUSPENDED
00111110	% #3E ISDN_INCOMPATIBLE_DESTINATION
00111111	% #3F ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000	% #40 ISDN_OUTGOING_CALLS_BARRED
01000001	% #41 – Not Used
01000010	% #42 – Not Used
01000011	% #43 – Not Used
01000100	% #44 – Not Used
01000101	% #45 – Not Used
01000110	% #46 – Not Used
01000111	% #47 – Not Used
01001000	% #48 – Not Used
01001001	% #49 – Not Used
01001010	% #4A ISDN_USER_NOT_MEMBER_OF_CUG

Connect optional parameters

The `Connect` message supports the following optional parameters :
User to User Information.

Note: Each new optional parameter must start a new word. If the previous optional parameter did not fill out the word, (there is a byte missing to fill out the word), it then pads the rest of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the word.

User to user information

7	6	5	4	3	2	1	0																	
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%; border: 1px solid black; text-align: center;">FD</td> <td style="border: 1px solid black; text-align: center;"> PID 1 0 1 1 0 1 </td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">Length</td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">B-Data</td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">B-Data</td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">B-Data</td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">B-Data</td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;"> </td> </tr> <tr> <td colspan="2" style="border: 1px solid black; text-align: center;">B-Data</td> </tr> </table>								FD	PID 1 0 1 1 0 1	Length		B-Data		B-Data		B-Data		B-Data			B-Data		Octet 0 Octet 1 Octet 2 Octet 3 Octet 130
FD	PID 1 0 1 1 0 1																							
Length																								
B-Data																								
B-Data																								
B-Data																								
B-Data																								
. . . .																								
B-Data																								

PID (Octet 0)

Bits	
5 4 3 2 1 0	
1 0 1 1 0 1	% 45 UUI_PID

Format discriminator (Octet 0)

Bits	
7 6	
0 0	Optional_Parm_FD_V1

Parameter length (Octet 1)

This parameter is defined as one byte, but ranges from 1 to 129.

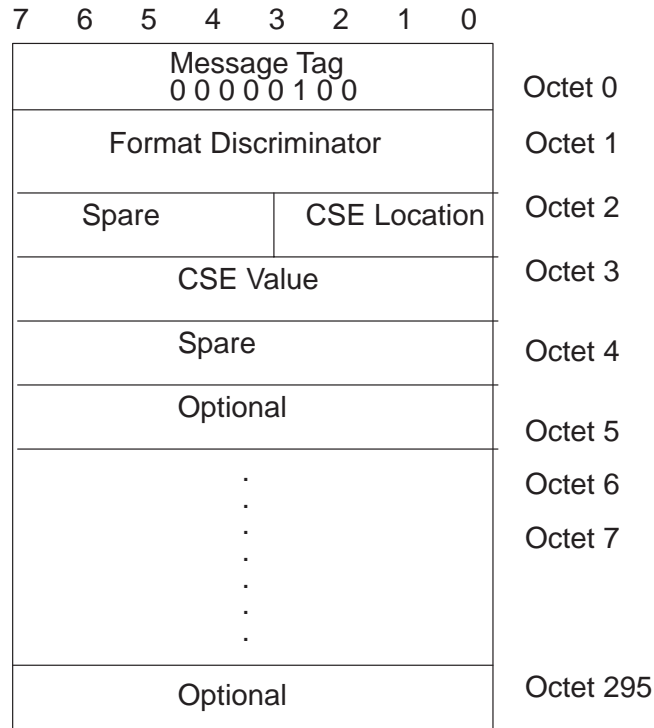
B-Data (Octet 2–130)

The maximum size of B-Data is 129 bytes long.

Disconnect message

The following diagram shows the layout of the Disconnect message.

Release Message



Disconnect mandatory message

message tag(Octet 0)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 1 0 0

%% 4 – Disconnect message tag

Format discriminator (Octet 1)

Bits

7 6

0 0

SCP_C_FD_V1

CSE location (Octet 2)

Bits

3 2 1 0

0001	% 1	LOCATION_USER
0010	% 2	LOCATION_PVT_NETWORK_SERV_LOCAL
0011	% 3	LOCATION_PUB_NETWORK_SERV_LOCAL
0100	% 4	LOCATION_TRANSIT_NETWORK
0101	% 5	LOCATION_PVT_NETWORK_SERV_REMOTE
0110	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE
0111	% 7	LOCATION_INTERNATIONAL_NETWORK
1000	% 8	LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1001	% 9	LOCATION_LOCAL_INTERFACE_ CONTRA_BY_THIS_SIG_LINK

CSE value (Octet 3)

Bits		
7 6 5 4 3 2 1 0		
00000001	% 1	ISDN_UNASSIGNED_NUMBER_CSE
00000010	% 2	ISDN_USER_BUSY
00000011	% #3	ISDN_INCOMPLETE_NUMBER
00000100	% #4	ISDN_NORMAL_UNSPECIFIED
00000101	% #5	ISDN_NO_CHANNEL_OR_CIRCUIT_ AVAILABLE
00000110	% #6	ISDN_SWITCH_EQUIPMENT_CONGESTION
00000111	% #7	ISDN_RESOURCE_UNAVAILABLE
00001000	% #8	ISDN_SERVICE_OR_OPTION_NOT_ AVAILABLE
00001001	% #9	ISDN_DEST_MISSING
00001010	% #A	ISDN_MESSAGE_NOT_COMPATIBLE_WITH_ STATE
00001011	% #B	ISDN_PROTOCOL_ERROR_UNSPECIFIED
00001100	% #C	ISDN_NO_ROUTE_TO_SPECIFIED_ TRANSIT_NETWORK
00001101	% #D	ISDN_NORMAL_CALL_CLEARING
00001110	% #E	ISDN_NO_USER_RESPONDING
00001111	% #F	ISDN_CALL_REJECTED
00010000	% #10	ISDN_NUMBER_CHANGED
00010001	% #11	ISDN_FACILITY_REJECTED
00010010	% #12	ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13	ISDN_USER_INFORMATION_DISCARDED
00010100	% #14	ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15	ISDN_REQUESTED_FACILITY_NOT SUBSCRIBED
00010110	% #16	ISDN_INCOMING_CALLS BARRED
00010111	% #17	ISDN_BC_NOT AUTHORIZED
00011000	% #18	ISDN_BC_NOT IMPLEMENTED

00011001	% #19 ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B
	ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D
	ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20
	ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A – Not Used
00111011	% #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100	% #3C ISDN_CALL_IDENTITY_IN_USE
00111101	% #3D ISDN_NOT_CALL_SUSPENDED
00111110	% #3E ISDN_INCOMPATIBLE_DESTINATION
00111111	% #3F
	ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000	% #40 ISDN_OUTGOING_CALLS_BARRED
01000001	% #41 – Not Used
01000010	% #42 – Not Used

```

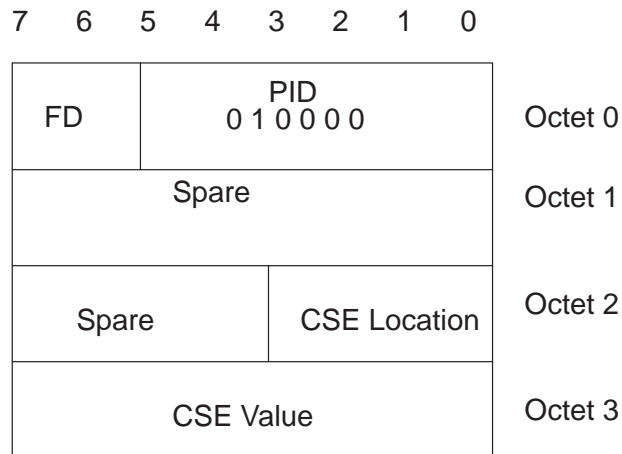
0 1 0 0 0 0 1 1    % #43 – Not Used
0 1 0 0 0 1 0 0    % #44 – Not Used
0 1 0 0 0 1 0 1    % #45 – Not Used
0 1 0 0 0 1 1 0    % #46 – Not Used
0 1 0 0 0 1 1 1    % #47 – Not Used
0 1 0 0 1 0 0 0    % #48 – Not Used
0 1 0 0 1 0 0 1    % #49 – Not Used
0 1 0 0 1 0 1 0    % #4A  ISDN_USER_NOT_MEMBER_OF_CUG
    
```

Disconnect optional parameters

The `Disconnect` message supports the following optional parameters: *Cause* and *UsertoUserInformation*.

Note: Each new optional parameter must start a new word. If the previous optional parameter did not fill out the word, meaning that there is a byte missing to fill out the word, it then pads the rest of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the word.

Cause message



PID (Octet 0)

```

Bits
5 4 3 2 1 0
-----
0 1 0 0 0 0          %16  CSE_PID
    
```

Format discriminator (Octet 0)

Bits	
7 6	
—	
0 1	Optional_Parm_FD_V2

CSE location (Octet 2)

Bits	
3 2 1 0	
—	
0 0 0 1	% 1 LOCATION_USER
0 0 1 0	% 2 LOCATION_PVT_NETWORK_SERV_LOCAL
0 0 1 1	% 3 LOCATION_PUB_NETWORK_SERV_LOCAL
0 1 0 0	% 4 LOCATION_TRANSIT_NETWORK
0 1 0 1	% 5 LOCATION_PVT_NETWORK_SERV_REMOTE
0 1 1 0	% 6 LOCATION_PUB_NETWORK_SERV_REMOTE
0 1 1 1	% 7 LOCATION_INTERNATIONAL_NETWORK
1 0 0 0	% 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1	% 9 LOCATION_LOCAL_INTERFACE_
	CONTRA_BY_THIS_SIG_LINK

CSE value (Octet 3)

Bits	
7 6 5 4 3 2 1 0	
—	
0 0 0 0 0 0 0 1	% 1 ISDN_UNASSIGNED_NUMBER_CSE
0 0 0 0 0 0 1 0	% 2 ISDN_USER_BUSY
0 0 0 0 0 0 1 1	% #3 ISDN_INCOMPLETE_NUMBER
0 0 0 0 0 1 0 0	% #4 ISDN_NORMAL_UNSPECIFIED
0 0 0 0 0 1 0 1	% #5
	ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
0 0 0 0 0 1 1 0	% #6 ISDN_SWITCH_EQUIPMENT_CONGESTION
0 0 0 0 0 1 1 1	% #7 ISDN_RESOURCE_UNAVAILABLE
0 0 0 0 1 0 0 0	% #8
	ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
0 0 0 0 1 0 0 1	% #9 ISDN_DEST_MISSING
0 0 0 0 1 0 1 0	% #A
	ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE
0 0 0 0 1 0 1 1	% #B ISDN_PROTOCOL_ERROR_UNSPECIFIED
0 0 0 0 1 1 0 0	% #C ISDN_NO_ROUTE_TO_SPECIFIED_
	TRANSIT_NETWORK
0 0 0 0 1 1 0 1	% #D ISDN_NORMAL_CALL_CLEARING

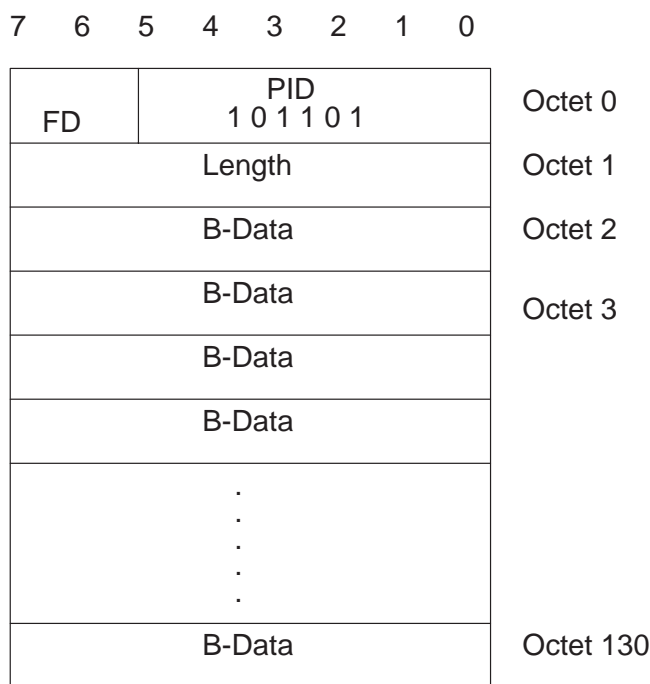
00001110	% #E ISDN_NO_USER_RESPONDING
00001111	% #F ISDN_CALL_REJECTED
00010000	% #10 ISDN_NUMBER_CHANGED
00010001	% #11 ISDN_FACILITY_REJECTED
00010010	% #12 ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13 ISDN_USER_INFORMATION_DISCARDED
00010100	% #14 ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15 ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110	% #16 ISDN_INCOMING_CALLS_BARRED
00010111	% #17 ISDN_BC_NOT_AUTHORIZED
00011000	% #18 ISDN_BC_NOT_IMPLEMENTED
00011001	% #19 ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20 ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED

```

00111000 % #38 ISDN_NON_SELECTED_USER_CLEARING
00111001 % #39 ISDN_NETWORK_OUT_OF_ORDER
00111010 % #3A – Not Used
00111011 % #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100 % #3C ISDN_CALL_IDENTITY_IN_USE
00111101 % #3D ISDN_NOT_CALL_SUSPENDED
00111110 % #3E ISDN_INCOMPATIBLE_DESTINATION
00111111 % #3F
                ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000 % #40 ISDN_OUTGOING_CALLS_BARRED
01000001 % #41 – Not Used
01000010 % #42 – Not Used
01000011 % #43 – Not Used
01000100 % #44 – Not Used
01000101 % #45 – Not Used
01000110 % #46 – Not Used
01000111 % #47 – Not Used
01001000 % #48 – Not Used
01001001 % #49 – Not Used
01001010 % #4A ISDN_USER_NOT_MEMBER_OF_CUG

```

User to user information



PID (Octet 0)

Bits

5 4 3 2 1 0

1 0 1 1 0 1 % 45 UUI_PID

Format discriminator (Octet 0)

Bits
7 6

0 0 Optional_Parm_FD_V1

Parameter length (Octet 1)

This parameter is defined as one byte, but ranges from 1 to 129.

B-Data (Octet 2–130)

The maximum size of B-Data is 129 bytes long.

Note: Each new optional parameter must start a new WORD. If the previous optional parameter did not fill out the WORD, meaning that there is a byte missing to fill out the WORD, it then pads the rest of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the WORD.

Facility (call associated) message (encoded)

Figure 15-2 shows the layout of the encoded version of the Facility (Call Associated) message.

Figure 15-2
Call associated facility message (encoded)

7	6	5	4	3	2	1	0	
Message Type								Octet 0
Format Discriminator								Octet 1
Sequence Number								Octet 2
FAC ID								Octet 3
EXT EXP		First Half		FAC Length				Octet 4
Orig.Net ID								Octet 5
Orig Net ID								Octet 6
Dest Net ID (LSB)								Octet 7
Dest Net ID (MSB)								Octet 8
Reject Option		Orig Type		Orig Encoded				Octet 9
Dest Type Dest Encoded								Octet 10
Orig Length								Octet 11
Dest Length								Octet 12
Optional								Octet 13
.								
.								
.								
.								
.								
Optional								Octet 61

Facility (call associated) mandatory parameters

message tag(Octet 0)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 1 0 0 0

%% 8 – Facility message tag

Format discriminator (Octet 1)

Bits

7 6

0 0

Optional_Parm_FD_V1

Sequence number (Octet 2)

Sequence is 1 byte.

FAC ID (Octet 3)

FAC ID is 1 byte.

EXT EXP (Octet 4)

EXT EXP is 1 bit.

First half (Octet 4)

First half is 1 bit.

FAC length (Octet 4)

FAC length is 6 bits.

Orig net ID (Octets 5 and 6)

Orig net ID is 2 bytes.

Dest net ID (Octets 7 and 8)

Dest net ID is 2 bytes.

Reject option (Octet 9)

Reject option is 1 bit.

Orig type (Octet 9)

Orig type is 4 bits.

Orig encoded (Octet 9)

Orig encoded is 3 bits.

Dest type (Octet 10)

Dest type is 4 bits.

Dest encoded (Octet 10)

Dest encoded is 3 bits.

Orig length (Octet 11)

Orig length is 1 byte.

Dest length (Octet 12)

Dest length is 1 byte.

Facility (call associated) optional parameters

The `Facility` message supports the following optional parameters : *CalledPartyNumber* and *Called PartySubaddress*.

Called party number (CDN)

7	6	5	4	3	2	1	0		
FD		PID 0 0 0 1 0 0							Octet 0
Spare								Octet 1	
CDN TON			CDN Length						Octet 2
CDN PI		CDN SI	CDN NPI						Octet 3
CDN Digits								Octet 4	
.									
.									
.									
.									
.									
CDN Digits								Octet 33	

PID (Octet 0)

0 0 0 1 0 0 % 4 CDN_PID

Format discriminator (Octet 0)

Bits

7 6

0 0 Optional_Parm_FD_V1
0 1 Optional_Parm_FD_V2

CDN length (Octet 2)

The CDN is five bits long, but its internal structure is limited from 1 to 30. The CDN length indicates how many digits are contained in the *CDNDigits* parameter.

CDN TON (Octet 2)

Bits

7 6 5

0 0 0 % 0 TON_UNKNOWN
0 0 1 % 1 TON_INTERNATIONAL
0 1 0 % 2 TON_NATIONAL
0 1 1 % 3 TON_LOCAL
1 0 0 % 4 TON_NETWORK

CDN NPI (Octet 3)

Bits

3 2 1 0

0 0 0 0 % 0 NPI_UNKNOWN
0 0 0 1 % 1 NPI_E164
0 0 1 0 % 2 NPI_PRIVATE

CDN SI (Octet 3)

Bits

5 4

0 1 % 1 SI_USER_PROVIDED
1 0 % 2 SI_NETWORK_PROVIDED
1 1 % 3 SI_USER_PROVIDED_VERIF_AND_PASSED

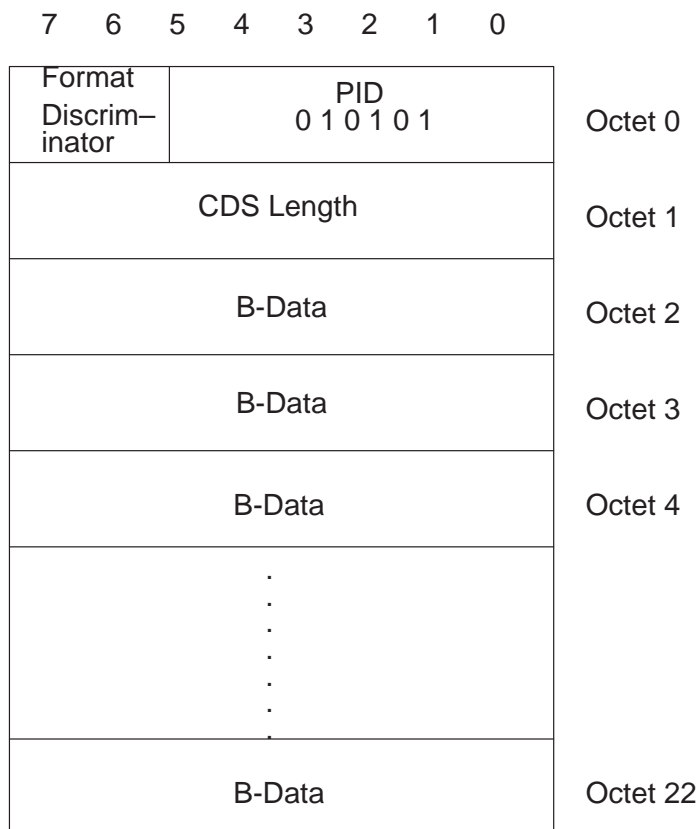
CDN PI (Octet 3)

Bits			
7 6			
—			
0 1	% 1	PI_PRESENTATION_ALLOWED	
1 0	% 2	PI_PRESENTATION_RESTRICTED	
1 1	% 3	PI_NOT_AVAILABLE	

CDN digits (Octet 4 – 33)

The maximum size of CDN Digits is 30 bytes long (or 60 digits long). Each digit in the CDN Digits occupies one nibble.

Called party subaddress(CDS)



PID (Octet 0)

15-28 PRI messages

Bits

5 4 3 2 1 0

0 1 0 1 0 1

% 21 CDS_PID

Format discriminator (Octet 0)

Bits
 7 6

 0 0 Optional_Parm_FD_V1

CDS length (Octet 1)

The CDS is eight bits long and ranges from 1 to 255; however, its internal range structure is defined is from 1 to 21.

B-Data (Octet 2 – Octet 22)

The maximum size of B-Data is 21 bytes long.

Progress message

7	6	5	4	3	2	1	0	
Message Tag 0 0 0 0 1 0 0 1								Octet 0
Format Discriminator								Octet 1
Progress Phase Ind.								Octet 2
PI Location				CSE Location				Octet 3
Notification Description								Octet 4
CSE Value								Octet 5
PI Description								Octet 6
Optional								Octet 7
:								
Optional								Octet 249

Progress mandatory parameters

message tag(Octet 0)

Bits
 7 6 5 4 3 2 1 0

00001001 %% 9 – Progress message tag

Format discriminator (Octet 1)

Bits

7 6 5 4 3 2 1 0

00000000 Optional_Parm_FD_V1

Progress phase indicator (Octet 2)

Bits

7 6 5 4 3 2 1 0

00000000	% 0	PPI_NIL_VALUE
00000001	% 1	PPI_PROGRESS
00000010	% 2	PPI_ADDRESS_INFORMATION
00000011	% 3	PPI_ALERTING
00000100	% 4	PPI_PROCEEDING
00000101	% 5	PPI_PREEMPTEE_CLEARED
00000110	% 6	PPI_SETUP_ACK
00000111	% 7	PPI_CONNECT

CSE location (Octet 3)

Bits

3 2 1 0

0001	% 1	LOCATION_USER
0010	% 2	LOCATION_PVT_NETWORK_SERV_LOCAL
0011	% 3	LOCATION_PUB_NETWORK_SERV_LOCAL
0100	% 4	LOCATION_TRANSIT_NETWORK
0101	% 5	LOCATION_PVT_NETWORK_SERV_REMOTE
0110	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE
0111	% 7	LOCATION_INTERNATIONAL_NETWORK
1000	% 8	LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1001	% 9	LOCATION_LOCAL_INTERFACE_CONTRA_BY_THIS_SIG_LINK

PI location (Octet 3)

Bits

3 2 1 0

0001	% 1	LOCATION_USER
0010	% 2	LOCATION_PVT_NETWORK_SERV_LOCAL
0011	% 3	LOCATION_PUB_NETWORK_SERV_LOCAL
0100	% 4	LOCATION_TRANSIT_NETWORK
0101	% 5	LOCATION_PVT_NETWORK_SERV_REMOTE
0110	% 6	LOCATION_PUB_NETWORK_SERV_REMOTE

```

0 1 1 1   % 7 LOCATION_INTERNATIONAL_NETWORK
1 0 0 0   % 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1   % 9 LOCATION_LOCAL_INTERFACE_
           CONTR_BY_THIS_SIG_LINK

```

Notification description (Octet 4)

Bits

7 6 5 4 3 2 1 0

```

-----
0 0 0 0 0 0 0 0   % 0  NOTIF_NIL_VALUE
0 0 0 0 0 0 0 1   % 1  NOTIF_USER_SUSPENDED
0 0 0 0 0 0 1 0   % 2  NOTIF_USER_RESUMED
0 0 0 0 0 0 1 1   % 3  NOTIF_BEARER_SERVICE_CHANGE
0 0 0 0 0 1 0 0   % 4  NOTIF_CALL_INFORMATION_EVENT
0 0 0 0 1 1 0 0   % 12 NOTIF_RETRIEVE_HELD_CALL
0 0 0 1 0 0 0 1   % 17 NOTIF_CALL_ON_HOLD
0 0 0 1 0 1 1 0   % 22 NOTIF_CALL_WAITING
0 0 0 1 1 1 1 0   % 30 NOTIF_REMOTE_HOLD
0 0 0 1 1 1 1 1   % 31 NOTIF_REMOTE_RELEASED
0 1 0 0 0 0 0 0   % 32 NOTIF_CALL_IS_FORWARDED
0 1 0 0 0 0 0 1   % 33 NOTIF_FORWARDING_ACTIVATED

```

CSE value (Octet 5)

Bits

7 6 5 4 3 2 1 0

```

-----
0 0 0 0 0 0 0 1   % 1  ISDN_UNASSIGNED_NUMBER_CSE
0 0 0 0 0 0 1 0   % 2  ISDN_USER_BUSY
0 0 0 0 0 0 1 1   % #3 ISDN_INCOMPLETE_NUMBER
0 0 0 0 0 1 0 0   % #4 ISDN_NORMAL_UNSPECIFIED
0 0 0 0 0 1 0 1   % #5
                       ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
0 0 0 0 0 1 1 0   % #6 ISDN_SWITCH_EQUIPMENT_CONGESTION
0 0 0 0 0 1 1 1   % #7 ISDN_RESOURCE_UNAVAILABLE
0 0 0 0 1 0 0 0   % #8
                       ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
0 0 0 0 1 0 0 1   % #9 ISDN_DEST_MISSING
0 0 0 0 1 0 1 0   % #A
                       ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE
0 0 0 0 1 0 1 1   % #B ISDN_PROTOCOL_ERROR_UNSPECIFIED
0 0 0 0 1 1 0 0   % #C ISDN_NO_ROUTE_TO_SPECIFIED_
                       TRANSIT_NETWORK
0 0 0 0 1 1 0 1   % #D ISDN_NORMAL_CALL_CLEARING
0 0 0 0 1 1 1 0   % #E ISDN_NO_USER_RESPONDING
0 0 0 0 1 1 1 1   % #F ISDN_CALL_REJECTED
0 0 0 1 0 0 0 0   % #10 ISDN_NUMBER_CHANGED

```

00010001	% #11 ISDN_FACILITY_REJECTED
00010010	% #12 ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13 ISDN_USER_INFORMATION_DISCARDED
00010100	% #14 ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15 ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110	% #16 ISDN_INCOMING_CALLS_BARRED
00010111	% #17 ISDN_BC_NOT_AUTHORIZED
00011000	% #18 ISDN_BC_NOT_IMPLEMENTED
00011001	% #19 ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20 ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A – Not Used

00111011	% #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100	% #3C ISDN_CALL_IDENTITY_IN_USE
00111101	% #3D ISDN_NOT_CALL_SUSPENDED
00111110	% #3E ISDN_INCOMPATIBLE_DESTINATION
00111111	% #3F ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000	% #40 ISDN_OUTGOING_CALLS_BARRED
01000001	% #41 – Not Used
01000010	% #42 – Not Used
01000011	% #43 – Not Used
01000100	% #44 – Not Used
01000101	% #45 – Not Used
01000110	% #46 – Not Used
01000111	% #47 – Not Used
01001000	% #48 – Not Used
01001001	% #49 – Not Used
01001010	% #4A ISDN_USER_NOT_MEMBER_OF_CUG

PI description (Octet 6)

Bits

7 6 5 4 3 2 1

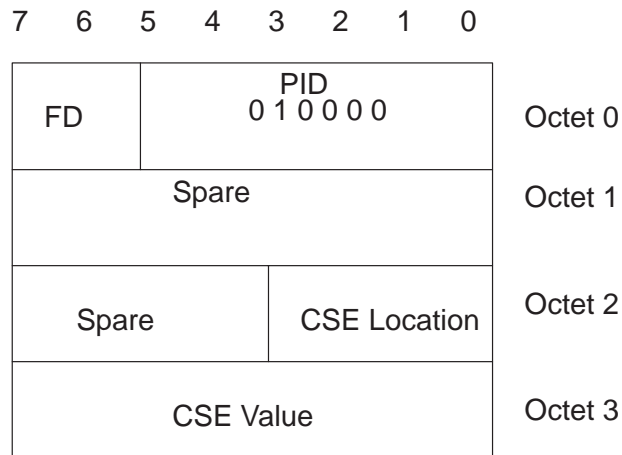
0000001	% 1 PROG_IND_NOT_END_TO_END_ISDN
0000010	% 2 PROG_IND_IN_BAND_INFO_AVAIL
0000011	% 3 PROG_IND_DEST_NOT_RESP
0000100	% 4 Not Used
0000101	% 5 Not Used
0000110	% 6 Not Used
0000111	% 7 PROG_IND_DEST_ADDR_IS_NOT_ISDN
0001000	% 8 PROG_IND_ORIG_ADDR_IS_NOT_ISDN
0001001	% 9 PROG_IND_CALL_RETURNED_TO_ISDN
0001010	% 10 Not Used
0001011	% 11 PROG_IND_DELAY_RESP_CALLED_INTERFACE
0001100	% 12 PROG_IND_OVLP_NOT_END_TO_END_ISDN

Progress optional parameters

The *Progress* message support the following optional parameters : *Cause* and *UserToUserInformation*.

Note: Each new optional parameter must start a new WORD. If the previous optional parameter did not fill out the WORD, meaning that there is a byte missing to fill out the WORD, it then pads the reset of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the WORD.

Cause



PID (Octet 0)

Bits
 5 4 3 2 1 0

 0 1 0 0 0 0 %16 CSE_PID

Format discriminator (Octet 0)

Bits
 7 6

 0 0 Optional_Parm_FD_V1

CSE location (Octet 2)

Bits
 3 2 1 0

 0 0 0 1 % 1 LOCATION_USER
 0 0 1 0 % 2 LOCATION_PVT_NETWORK_SERV_LOCAL
 0 0 1 1 % 3 LOCATION_PUB_NETWORK_SERV_LOCAL
 0 1 0 0 % 4 LOCATION_TRANSIT_NETWORK
 0 1 0 1 % 5 LOCATION_PVT_NETWORK_SERV_REMOTE
 0 1 1 0 % 6 LOCATION_PUB_NETWORK_SERV_REMOTE
 0 1 1 1 % 7 LOCATION_INTERNATIONAL_NETWORK
 1 0 0 0 % 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT
 1 0 0 1 % 9 LOCATION_LOCAL_INTERFACE_

CONTRA_BY_THIS_SIG_LINK

CSE value (Octet 3)

Bits

7 6 5 4 3 2 1 0

00000001	%1	ISDN_UNASSIGNED_NUMBER_CSE
00000010	%2	ISDN_USER_BUSY
00000011	% #3	ISDN_INCOMPLETE_NUMBER
00000100	% #4	ISDN_NORMAL_UNSPECIFIED
00000101	% #5	ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
00000110	% #6	ISDN_SWITCH_EQUIPMENT_CONGESTION
00000111	% #7	ISDN_RESOURCE_UNAVAILABLE
00001000	% #8	ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
00001001	% #9	ISDN_DEST_MISSING
00001010	% #A	ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE
00001011	% #B	ISDN_PROTOCOL_ERROR_UNSPECIFIED
00001100	% #C	ISDN_NO_ROUTE_TO_SPECIFIED_TRANSIT_NETWORK
00001101	% #D	ISDN_NORMAL_CALL_CLEARING
00001110	% #E	ISDN_NO_USER_RESPONDING
00001111	% #F	ISDN_CALL_REJECTED
00010000	% #10	ISDN_NUMBER_CHANGED
00010001	% #11	ISDN_FACILITY_REJECTED
00010010	% #12	ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13	ISDN_USER_INFORMATION_DISCARDED
00010100	% #14	ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15	ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110	% #16	ISDN_INCOMING_CALLS_BARRED
00010111	% #17	ISDN_BC_NOT_AUTHORIZED
00011000	% #18	ISDN_BC_NOT_IMPLEMENTED
00011001	% #19	ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A	ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B	ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C	ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D	ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E	ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F	ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20	

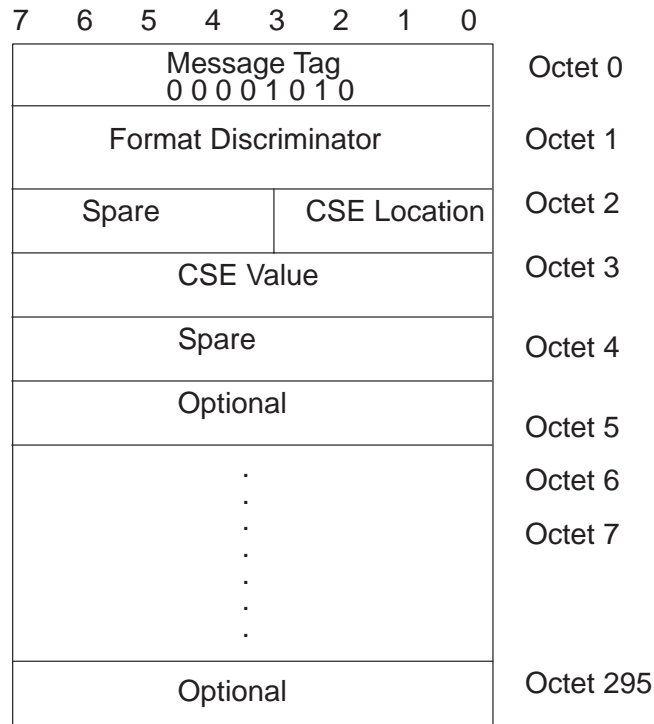
	ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A Not Used
00111011	% #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100	% #3C ISDN_CALL_IDENTITY_IN_USE
00111101	% #3D ISDN_NOT_CALL_SUSPENDED
00111110	% #3E ISDN_INCOMPATIBLE_DESTINATION
00111111	% #3F
	ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000	% #40 ISDN_OUTGOING_CALLS_BARRED
01000001	% #41 – Not Used
01000010	% #42 – Not Used
01000011	% #43 – Not Used
01000100	% #44 – Not Used
01000101	% #45 – Not Used
01000110	% #46 – Not Used
01000111	% #47 – Not Used
01001000	% #48 – Not Used
01001001	% #49 – Not Used
01001010	% #4A ISDN_USER_NOT_MEMBER_OF_CUG

The maximum size of B-Data is 129 bytes long.

Release message

The following figure shows the layout of the Release message.

Release message



Release mandatory message

message tag(Octet 0)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 1 0 1 0

%% 10 – Release message tag

Format discriminator (Octet 1)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

SCP_C_FD_V1

CSE location (Octet 2)

Bits

3 2 1 0

```

0001  % 1 LOCATION_USER
0010  % 2 LOCATION_PVT_NETWORK_SERV_LOCAL
0011  % 3 LOCATION_PUB_NETWORK_SERV_LOCAL
0100  % 4 LOCATION_TRANSIT_NETWORK
0101  % 5 LOCATION_PVT_NETWORK_SERV_REMOTE
0110  % 6 LOCATION_PUB_NETWORK_SERV_REMOTE
0111  % 7 LOCATION_INTERNATIONAL_NETWORK
1000  % 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1001  % 9 LOCATION_LOCAL_INTERFACE_
      CONTRA_BY_THIS_SIG_LINK

```

CSE value (Octet 3)

```

Bits
76543210
-----
00000001  % 1  ISDN_UNASSIGNED_NUMBER_CSE
00000010  % 2  ISDN_USER_BUSY
00000011  % #3  ISDN_INCOMPLETE_NUMBER
00000100  % #4  ISDN_NORMAL_UNSPECIFIED
00000101  % #5
      ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
00000110  % #6  ISDN_SWITCH_EQUIPMENT_CONGESTION
00000111  % #7  ISDN_RESOURCE_UNAVAILABLE
00001000  % #8
      ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
00001001  % #9  ISDN_DEST_MISSING
00001010  % #A
      ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE
00001011  % #B  ISDN_PROTOCOL_ERROR_UNSPECIFIED
00001100  % #C  ISDN_NO_ROUTE_TO_SPECIFIED_
      TRANSIT_NETWORK
00001101  % #D  ISDN_NORMAL_CALL_CLEARING
00001110  % #E  ISDN_NO_USER_RESPONDING
00001111  % #F  ISDN_CALL_REJECTED
00010000  % #10  ISDN_NUMBER_CHANGED
00010001  % #11  ISDN_FACILITY_REJECTED
00010010  % #12  ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011  % #13  ISDN_USER_INFORMATION_DISCARDED
00010100  % #14
      ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101  % #15
      ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110  % #16  ISDN_INCOMING_CALLS_BARRED
00010111  % #17  ISDN_BC_NOT_AUTHORIZED
00011000  % #18  ISDN_BC_NOT_IMPLEMENTED
00011001  % #19  ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED

```

00011010	% #1A ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20 ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A – Not Used
00111011	% #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100	% #3C ISDN_CALL_IDENTITY_IN_USE
00111101	% #3D ISDN_NOT_CALL_SUSPENDED
00111110	% #3E ISDN_INCOMPATIBLE_DESTINATION
00111111	% #3F ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000	% #40 ISDN_OUTGOING_CALLS_BARRED
01000001	% #41 – Not Used
01000010	% #42 – Not Used
01000011	% #43 – Not Used
01000100	% #44 – Not Used

```

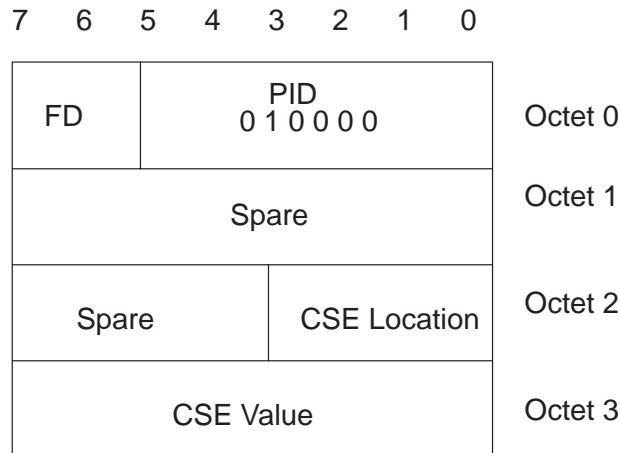
0 1 0 0 0 1 0 1    % #45 – Not Used
0 1 0 0 0 1 1 0    % #46 – Not Used
0 1 0 0 0 1 1 1    % #47 – Not Used
0 1 0 0 1 0 0 0    % #48 – Not Used
0 1 0 0 1 0 0 1    % #49 – Not Used
0 1 0 0 1 0 1 0    % #4A  ISDN_USER_NOT_MEMBER_OF_CUG
    
```

Release optional parameters

The Release message support the following optional parameters : *Cause*, and *UserToUserInformation*.

Note: Each new optional parameter must start a new word. If the previous optional parameter did not fill out the word, meaning that there is a byte missing to fill out the word, it then pads the reset of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the word.

Cause



PID (Octet 0)

```

Bits
5 4 3 2 1 0
-----
0 1 0 0 0 0          %16  CSE_PID
    
```

Format discriminator (Octet 0)

Bits

7 6
—
0 1 Optional_Parm_FD_V2
0 1 Optional_Parm_FD_V2

CSE location (Octet 2)

Bits
3 2 1 0
—
0 0 0 1 % 1 LOCATION_USER
0 0 1 0 % 2 LOCATION_PVT_NETWORK_SERV_LOCAL
0 0 1 1 % 3 LOCATION_PUB_NETWORK_SERV_LOCAL
0 1 0 0 % 4 LOCATION_TRANSIT_NETWORK
0 1 0 1 % 5 LOCATION_PVT_NETWORK_SERV_REMOTE
0 1 1 0 % 6 LOCATION_PUB_NETWORK_SERV_REMOTE
0 1 1 1 % 7 LOCATION_INTERNATIONAL_NETWORK
1 0 0 0 % 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT
1 0 0 1 % 9 LOCATION_LOCAL_INTERFACE_
 CONTRA_BY_THIS_SIG_LINK

CSE value (Octet 3)

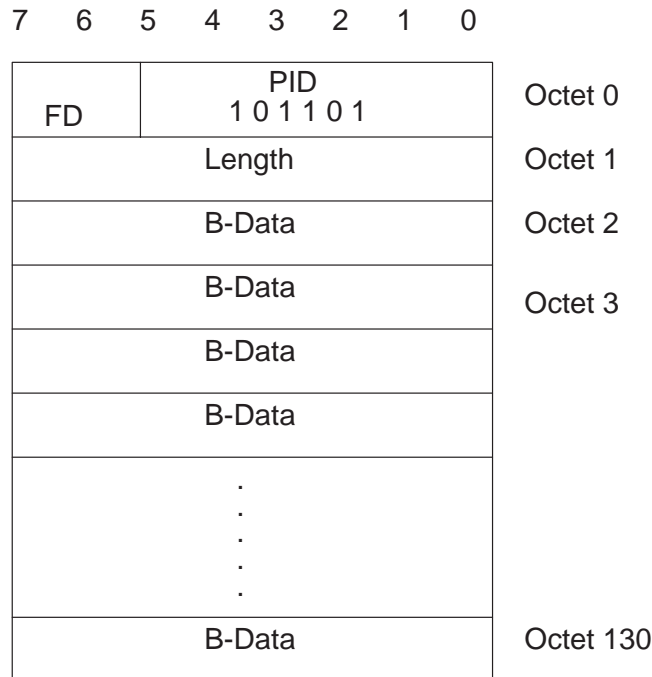
Bits
7 6 5 4 3 2 1 0
—
0 0 0 0 0 0 1 % 1 ISDN_UNASSIGNED_NUMBER_CSE
0 0 0 0 0 0 1 0 % 2 ISDN_USER_BUSY
0 0 0 0 0 0 1 1 % #3 ISDN_INCOMPLETE_NUMBER
0 0 0 0 0 1 0 0 % #4 ISDN_NORMAL_UNSPECIFIED
0 0 0 0 0 1 0 1 % #5
 ISDN_NO_CHANNEL_OR_CIRCUIT_AVAILABLE
0 0 0 0 0 1 1 0 % #6 ISDN_SWITCH_EQUIPMENT_CONGESTION
0 0 0 0 0 1 1 1 % #7 ISDN_RESOURCE_UNAVAILABLE
0 0 0 0 1 0 0 0 % #8
 ISDN_SERVICE_OR_OPTION_NOT_AVAILABLE
0 0 0 0 1 0 0 1 % #9 ISDN_DEST_MISSING
0 0 0 0 1 0 1 0 % #A
 ISDN_MESSAGE_NOT_COMPATIBLE_WITH_STATE
0 0 0 0 1 0 1 1 % #B ISDN_PROTOCOL_ERROR_UNSPECIFIED
0 0 0 0 1 1 0 0 % #C ISDN_NO_ROUTE_TO_SPECIFIED_
 TRANSIT_NETWORK
0 0 0 0 1 1 0 1 % #D ISDN_NORMAL_CALL_CLEARING
0 0 0 0 1 1 1 0 % #E ISDN_NO_USER_RESPONDING
0 0 0 0 1 1 1 1 % #F ISDN_CALL_REJECTED
0 0 0 1 0 0 0 0 % #10 ISDN_NUMBER_CHANGED

00010001	% #11 ISDN_FACILITY_REJECTED
00010010	% #12 ISDN_RESPONSE_TO_STATUS_ENQUIRY
00010011	% #13 ISDN_USER_INFORMATION_DISCARDED
00010100	% #14 ISDN_REQUESTED_CIRCUIT_NOT_AVAILABLE
00010101	% #15 ISDN_REQUESTED_FACILITY_NOT_SUBSCRIBED
00010110	% #16 ISDN_INCOMING_CALLS_BARRED
00010111	% #17 ISDN_BC_NOT_AUTHORIZED
00011000	% #18 ISDN_BC_NOT_IMPLEMENTED
00011001	% #19 ISDN_CHANNEL_TYPE_NOT_IMPLEMENTED
00011010	% #1A ISDN_ONLY_RES_DIG_INFO_BC_AVAILABLE
00011011	% #1B ISDN_SERVICE_OR_OPTION_NOT_IMPLEMENTED
00011100	% #1C ISDN_INVALID_CALL_REFERENCE_VALUE
00011101	% #1D ISDN_IDENTIFIED_CHANNEL_DOES_NOT_EXIST
00011110	% #1E ISDN_INCOMPATIBLE_DESTINATION
00011111	% #1F ISDN_INVALID_MESSAGE_UNSPECIFIED
00100000	% #20 ISDN_MANDATORY_INFO_ELEMENT_IS_MISSING
00100001	% #21 ISDN_MESSAGE_TYPE_IS_NON_EXISTENT
00100010	% #22 ISDN_INFO_ELEMENT_NON_EXISTENT
00100011	% #23 ISDN_INVALID_INFO_ELEMENT_CONTENTS
00100100	% #24 ISDN_INTERWORKING_UNSPECIFIED
00100101	% #25 ISDN_DST_OUT_OF_SERV
00100110	% #26 ISDN_TEMP_FAIL
00100111	% #27 ISDN_BC_NOT_PRESENTLY_AVAILABLE
00101000	% #28 – Not Used
00101001	% #29 ISDN_BC_ICOMPAT_WITH_REQ
00101010	% #2A ISDN_PREEMPT
00101011	% #2B ISDN_NO_PREEMPT_CIRCUIT_AVAIL
00101100	% #2C ISDN_PREEMPT_REUSE
00101101	% #2D ISDN_VACANT_CODE
00101110	% #2E ISDN_PRFX_1_DIALED_IN_ERROR
00101111	% #2F ISDN_PRFX_1_NOT_DIALED
00110000	% #30 ISDN_SERV_OPT_VIOL
00110001	% #31 – Not Used
00110010	% #32 ISDN_NORTE_TO_DSTN
00110011	% #33 ISDN_NO_ANS_FROM_USER
00110100	% #34 ISDN_RECOVERY_ON_TIMER_EXPIRY
00110101	% #35 ISDN_CHANNEL_UNACCEPTABLE
00110110	% #36 ISDN_FACILITY_NOT_IMPLEMENTED
00110111	% #37 ISDN_CALL_AWARDED
00111000	% #38 ISDN_NON_SELECTED_USER_CLEARING
00111001	% #39 ISDN_NETWORK_OUT_OF_ORDER
00111010	% #3A – Not Used

```

00111011 % #3B ISDN_A SUSPENDED_CALL_EXISTS
00111100 % #3C ISDN_CALL_IDENTITY_IN_USE
00111101 % #3D ISDN_NOT_CALL_SUSPENDED
00111110 % #3E ISDN_INCOMPATIBLE_DESTINATION
00111111 % #3F
                ISDN_INVALID_TRANSIT_NETWORK_SELECTION
01000000 % #40 ISDN_OUTGOING_CALLS_BARRED
01000001 % #41 – Not Used
01000010 % #42 – Not Used
01000011 % #43 – Not Used
01000100 % #44 – Not Used
01000101 % #45 – Not Used
01000110 % #46 – Not Used
01000111 % #47 – Not Used
01001000 % #48 – Not Used
01001001 % #49 – Not Used
01001010 % #4A ISDN_USER_NOT_MEMBER_OF_CUG
    
```

User to user information



PID (Octet 0)

```

Bits
5 4 3 2 1 0
    
```

1 0 1 1 0 1 % 45 UUI_PID

Format discriminator (Octet 0)

Bits
7 6

0 0 Optional_Parm_FD_V1

Length (Octet 1)

This parameter is defined as one byte, but ranges from 1 to 129.

B-Data (Octet 2–130)

The maximum size of B-Data is 129 bytes long.

Setup message

The following figure shows the layout of the Setup message.

7	6	5	4	3	2	1	0	
Message Tag 0 0 0 0 1 1 0 0								Octet 0
Format Discriminator								Octet 1
Spare		Bearer Capability						Octet 2
Spare				PI Location				Octet 3
Progress Indicator Description								Octet 4
TNS Network ID								Octet 5
TNS Network ID								Octet 6
CDN TON			CDN Length					Octet 7
Spare				CDN NPI				Octet 8
CGN TON			CGN Length					Octet 9
CGN PI		CGN SI		CGN NPI				Octet 10
CDN Digits (variable length)								Octet 11 .
CGN Digits (variable length)								.
Optional								Octet 354

SETUP mandatory parameters**message tag(Octet 0)**

Bits

7 6 5 4 3 2 1 0

0 0 0 0 1 1 0 0

%% 12 – Setup message tag

Format discriminator (Octet 1)

Bits

7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 0

SCP_C_FD_V1

0 0 0 0 0 0 0 1

SCP_C_FD_V2

Note: V1 has the digits in the mandatory parameters in bytes and V2 has the digits in nibble swapped format.

Bearer capability (Octet 2)

Bits

5 4 3 2 1 0

0 0 0 0 0 1

%1 BC_SPEECH

0 0 0 0 1 0

%2 BC_64K_DATA

0 0 0 0 1 1

%3 Not Used

0 0 0 1 0 0

%4 BC_56K_DATA

0 0 0 1 0 1

%5 Not Used

0 0 0 1 1 0

%6 BC_64K_RES

0 0 0 1 1 1

%7 BC_3_1_KHZ

0 0 1 0 0 0

%8 BC_7_KHZ

0 0 1 0 0 1

%9 BC_VOICE_DATA

0 0 1 0 1 0

%10 BC_64K_RATE_AD_DATA

Progress indicator location (Octet 3)

Bits

3 2 1 0

0 0 0 1

% 1 LOCATION_USER

0 0 1 0

% 2 LOCATION_PVT_NETWORK_SERV_LOCAL

0 0 1 1

% 3 LOCATION_PUB_NETWORK_SERV_LOCAL

0 1 0 0

% 4 LOCATION_TRANSIT_NETWORK

0 1 0 1

% 5 LOCATION_PVT_NETWORK_SERV_REMOTE

0 1 1 0

% 6 LOCATION_PUB_NETWORK_SERV_REMOTE

0 1 1 1

% 7 LOCATION_INTERNATIONAL_NETWORK

1 0 0 0

% 8 LOCATION_NETWORK_BEYOND_INERWORKING_POINT

1 0 0 1

% 9 LOCATION_LOCAL_INTERFACE_

CONTR_BY_THIS_SIG_LINK

Progress indicator description (Octet 4)

Bits

7 6 5 4 3 2 1

0 0 0 0 0 0 1	% 1	PROG_IND_NOT_END_TO_END_ISDN
0 0 0 0 0 1 0	% 2	PROG_IND_IN_BAND_INFO_AVAIL
0 0 0 0 0 1 1	% 3	PROG_IND_DEST_NOT_RESP
0 0 0 0 1 0 0	% 4	Not Used
0 0 0 0 1 0 1	% 5	Not Used
0 0 0 0 1 1 0	% 6	Not Used
0 0 0 0 1 1 1	% 7	PROG_IND_DEST_ADDR_IS_NOT_ISDN
0 0 0 1 0 0 0	% 8	PROG_IND_ORIG_ADDR_IS_NOT_ISDN
0 0 0 1 0 0 1	% 9	PROG_IND_CALL_RETURNED_TO_ISDN
0 0 0 1 0 1 0	% 10	Not Used
0 0 0 1 0 1 1	% 11	PROG_IND_DELAY_RESP_CALLED_INTERFACE
0 0 0 1 1 0 0	% 12	PROG_IND_OVLP_NOT_END_TO_END_ISDN

TNS (transmit network selector) (Octet 5 and 6)

This field is two bytes long which can hold 0 to 65,535 digits.

CDN length (Octet 7)

This parameter is five bytes long and ranges from 1 to 31. The CDN length value of 31 is the NIL value, which is translated to be 0.

CDN TON (Octet 7)

Bits

7 6 5

0 0 0	% 0	TON_UNKNOWN
0 0 1	% 1	TON_INTERNATIONAL
0 1 0	% 2	TON_NATIONAL
0 1 1	% 3	TON_LOCAL
1 0 0	% 4	TON_NETWORK

CDN NPI (Octet 8)

Bits

3 2 1 0

0 0 0 0	% 0	NPI_UNKNOWN
0 0 0 1	% 1	NPI_E164

0 0 1 0 % 2 NPI_PRIVATE

Calling party number (CGN) length (Octet 9)

This parameter is five bits long, which ranges from 1 to 30. The CGN length value of 31 is the NIL value, which is translated to be 0.

Calling party number (CGN) TON (Octet 9)

Bits

7 6 5

0 0 0	% 0	TON_UNKNOWN
0 0 1	% 1	TON_INTERNATIONAL
0 1 0	% 2	TON_NATIONAL
0 1 1	% 3	TON_LOCAL
1 0 0	% 4	TON_NETWORK

CGN NPI (Octet 10)

Bits

3 2 1 0

0 0 0 0	% 0	NPI_UNKNOWN
0 0 0 1	% 1	NPI_E164
0 0 1 0	% 2	NPI_PRIVATE

CGN screen indicator (Octet 10)

Bits

5 4

0 1	User_Provided
1 0	Network_Provided
1 1	User_Provided_Verif_and_Passed

CGN presentation indicator(Octet 10)

Bits

7 6

0 1	Presentation_Allowed
1 0	Presentation_Restricted
1 1	Not_Available

Setup optional parameters

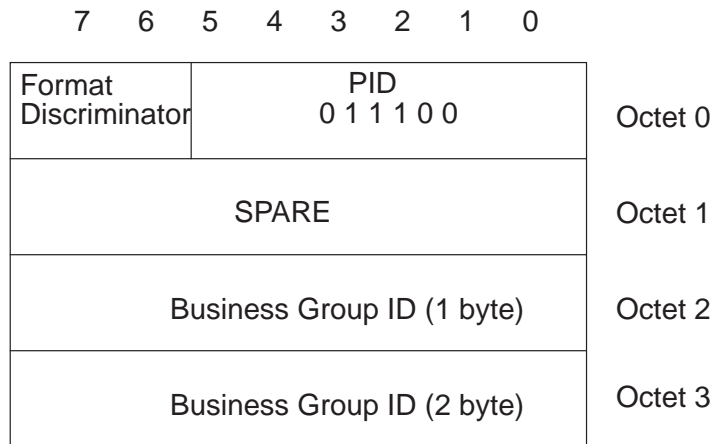
The *Setup* message support the following optional parameters:

BusinessGroup, CallingPartyNumber, CalledPartySubaddress,

CallingPartySubaddress, DisplayInformation, HigherLayerCompatibility, LowerLayerCompatibility, NetworkSpecificFacility, ProgressIndicator, TransitNetworkSelection, and UsertoUserInformation.

Note: Each new optional parameter must start a new word. If the previous optional parameter did not fill out the word, meaning that there is a byte missing to fill out the word, it then pads the reset of the word with a spare byte of 00. This spare byte will not be validated, it is just used for padding the optional parameter out to fill the rest of the word.

Business group



PID (Octet 0)

Bits
 5 4 3 2 1 0

 0 1 1 1 0 0 % 28 BG_PID

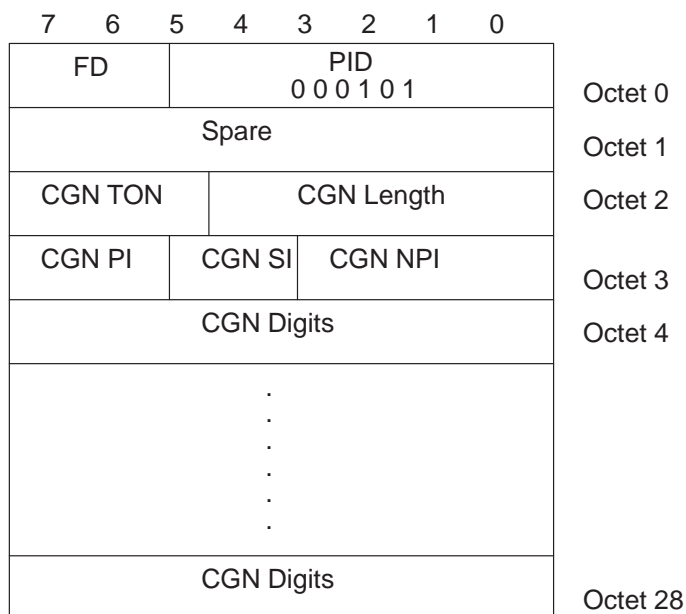
Format discriminator (Octet 0)

Bits
 7 6

 0 0 Optional_Parm_FD_V1

Business group ID (Octet 2 and 3)

The Business Group ID can hold an integer from 0 to 999.

Calling party number (CGN)**PID (Octet 0)**

```
0 0 0 1 0 1          % 5   CGN_PID
```

Format discriminator (Octet 0)

```
Bits
7 6
-----
0 0          Optional_Parm_FD_V1
```

CGN length (Octet 2)

The CGN is five bits long but internal structure is limited from 1 to 24. The CGN length indicates how many digits are contained in the CGN Digits.

CGN TON (Octet 2)

```
Bits
7 6 5
```

0 0 0	% 0	TON_UNKNOWN
0 0 1	% 1	TON_INTERNATIONAL
0 1 0	% 2	TON_NATIONAL
0 1 1	% 3	TON_LOCAL
1 0 0	% 4	TON_NETWORK

CGN NPI (Octet 3)

Bits
3 2 1 0

0 0 0 0	% 0	NPI_UNKNOWN
0 0 0 1	% 1	NPI_E164
0 0 1 0	% 2	NPI_PRIVATE

CGN SI (Octet 3)

Bits
5 4

0 1	% 1	SI_USER_PROVIDED
1 0	% 2	SI_NETWORK_PROVIDED
1 1	% 3	SI_USER_PROVIDED_VERIF_AND_PASSED

ICGN PI (Octet 3)

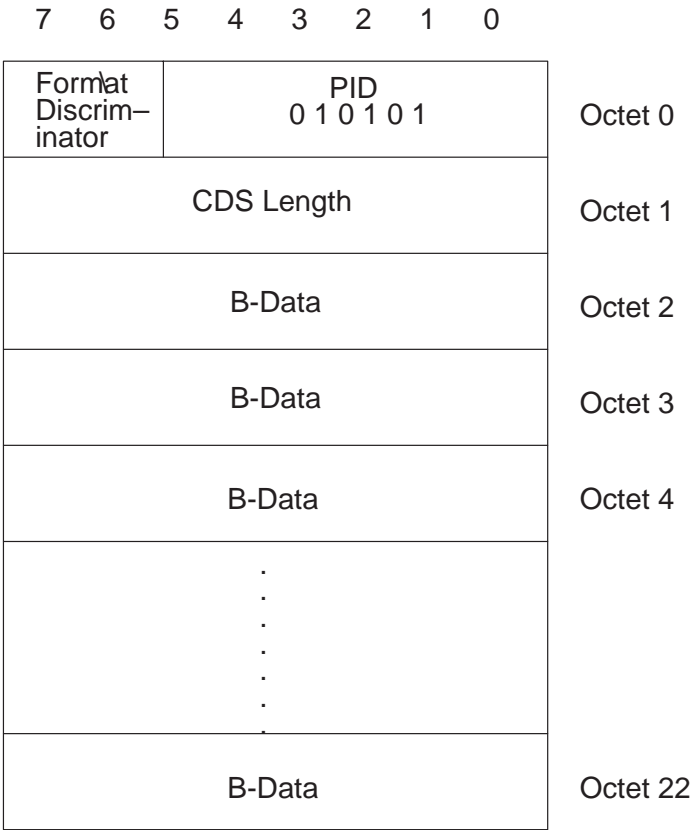
Bits
7 6

0 1	% 1	PI_PRESENTATION_ALLOWED
1 0	% 2	PI_PRESENTATION_RESTRICTED
1 1	% 3	PI_NOT_AVAILABLE

CGN digits (Octet 4 – 28)

The maximum size of CGN Digits is 24 bytes long (or 48 digits long). Each digit in the CGN Digits occupies one nibble.

Called party subaddress



PID (Octet 0)



Format discriminator (Octet 0)



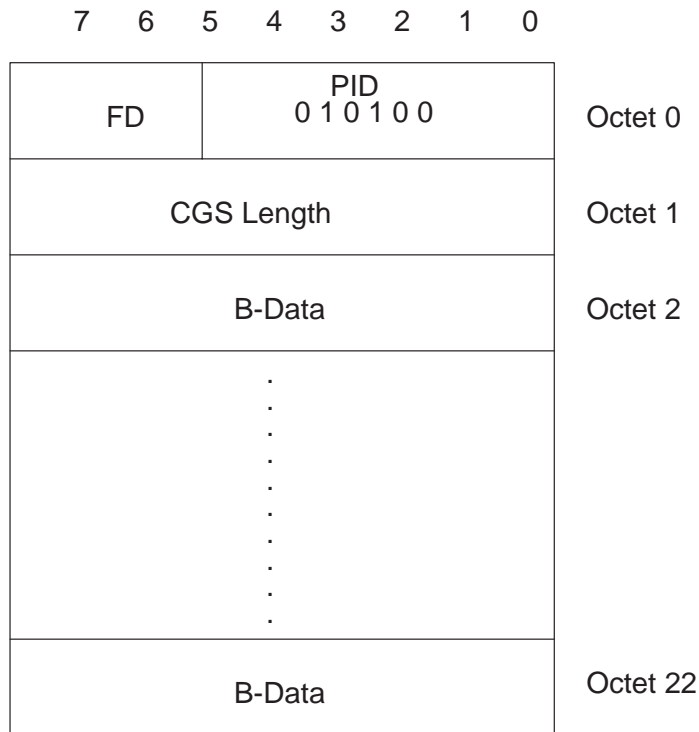
CDS length (Octet 1)

The CDS is eight bits long, which ranges from 1 to 255; however, its internal range structure is defined from 1 to 21.

B-Data (Octet 2 – Octet 22)

The maximum size of B-Data is 21 bytes long.

Calling party subaddress



PID (Octet 0)



Format discriminator (Octet 0)

Bits

7 6

0 0

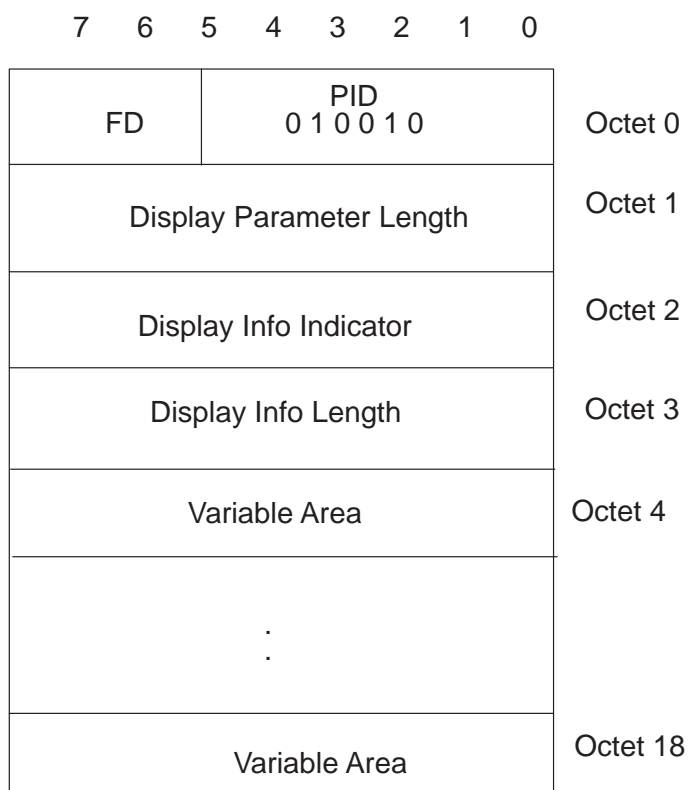
Optional_Parm_FD_V1

Calling party subaddress (CGS) (calling party length (Octet 1))

The CGS is eight bits long, which ranges from 1 to 255; however, its internal range structure is defined from 1 to 21.

B-Data (Octet 2–22)

The maximum size of B-Data is 21 bytes long.

Display

PID (Octet 0)

Bits
5 4 3 2 1 0

0 1 0 0 1 0 % 18 DISP_PID

Format discriminator (Octet 0)

Bits
7 6

0 0 Optional_Parm_FD_V1

Display parameter length (Octet 1)

Display length is 1 byte long which ranges from 5 to 19. The Display Length indicates how many bytes are in the *Display* Optional parameter itself. So there has to be a minimum of at least 5 bytes in the optional parameter for it to be valid.

Display info indicator (Octet 2)

Bits
7 6 5 4 3 2 1 0

0 0 0 0 0 0 0 1 % 1 OCN_PTY_NAME
0 0 0 0 0 0 1 0 % 2 CGN_PTY_NAME
0 0 0 0 0 0 1 1 % 3 CDN_PTY_NAME
0 0 0 0 0 1 0 0 % 4 RGN_PTY_NAME
0 0 0 0 0 1 0 1 % 5 RNN_PTY_NAME
0 0 0 0 0 1 1 0 % 6 CONN_PTY_NAME
0 0 0 0 0 1 1 1 % 7 MISC_DISPLAY_INFO
0 0 0 0 1 0 0 0 % 8 REASON_SUBSTITUTE_FOR_PTY_NAME

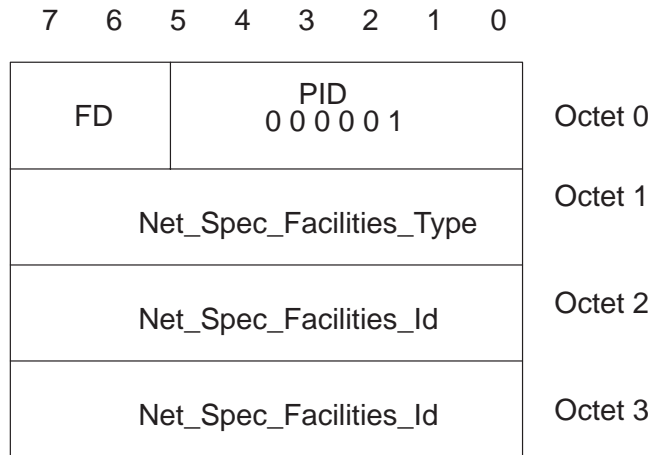
Display info length (Octet 3)

Display Info Length is 1 byte long which ranges from 1 to 15. The Display Info Length indicates how many bytes are in the Variable Area. So there has to be a minimum of at least 1 byte in the variable area for it to be valid.

Variable area (Octet 4 – 18)

Variable Area is 15 bytes long.

Network specific facility



PID (Octet 0)

Bits	
5 4 3 2 1 0	
0 0 0 0 0 1	% 1 NSF_PID

Format discriminator (Octet 0)

Bits	
7 6	
0 0	Optional_Parm_FD_V1

Net_Spec_Facilities (Octet 1)

Bits	
7 6 5 4 3 2 1 0	
0 0 0 0 0 0 0 0	% 0 NSF_NIL_VALUE
0 0 0 0 0 0 0 1	% 1 NSF_PRIVATE
0 0 0 0 0 0 1 0	% 2 NSF_INWATS

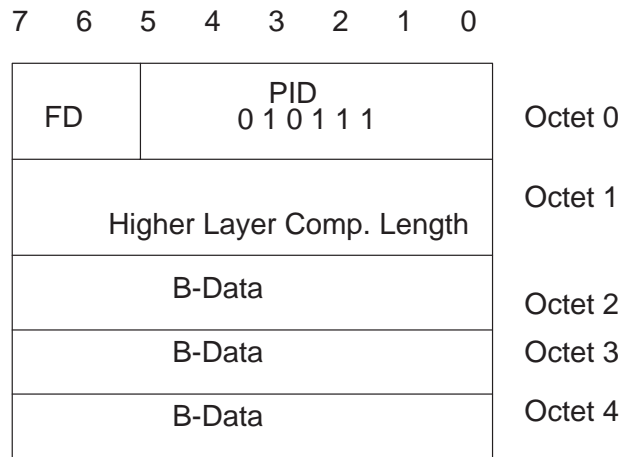
```

000000011  % 3 NSF_WATS
000000100  % 4 NSF_FX
000000101  % 5 NSF_TIE
000000110  % 6 NSF_SDS
000000111  % 7 NSF_LDS
000001000  % 8 NSF_900
    
```

Net_Spec_Facilities_ID (Octet 2)

Facility ID is two bytes long and ranges from 0 to 65,535.

Higher layer compatibility



PID (Octet 0)

```

Bits
5 4 3 2 1 0
-----
0 1 0 1 1 1          % 23  HLC_PID
    
```

Format discriminator (Octet 0)

```

Bits
7 6
-----
0 0          Optional_Parm_FD_V1
    
```


Higher layer compatibility length (Octet 1)

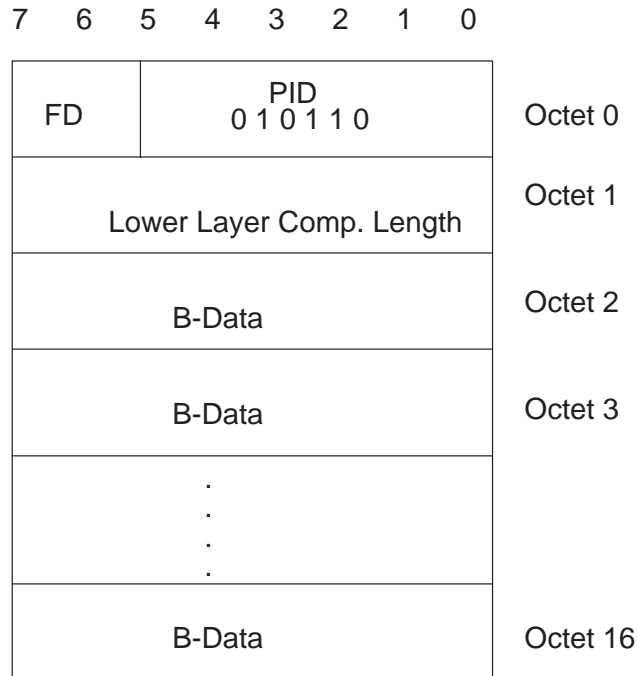
The Higher Layer Compatibility length is one byte long ; however, its internal range structure is defined as from 1 to 3.

B-Data (Octet 2– 4)

The maximum size of B-Data is three bytes long.

Lower layer compatibility

Lower layer compatibility



PID (Octet 0)



PID (Octet 0)

Bits
5 4 3 2 1 0

0 0 0 1 1 1 % 7 OCN_PID

Format discriminator (Octet 0)

Bits
7 6

0 0 Optional_Parm_FD_V1
0 1 Optional_Parm_FD_V2

Original redirection type(Octet 1)

Bits
3 2 1 0

0 0 0 0 RDR_NIL_VALUE
0 0 0 1 RDR_CFW_BUSY
0 0 1 0 RDR_CFW_NO_REPLY
0 0 1 1 RDR_CALL_TRANSFER
0 1 0 0 RDR_CALL_PICKUP
0 1 0 1 Not Used
0 1 1 0 Not Used
0 1 1 1 RDR_CFW_UNCONDITIONAL

Redirection count (Octet 1)

The Redirection Count is three bits long, which ranges from 0 to 5

Call forward no reply (Octet 1)

Bits
7

0 False
1 True

OCN length (Octet 2)

The OCN length is five bits long; however, its internal structure is limited from 0 to 14. This parameter indicates how many digits are in the *OCNDigits* parameter.

OCN TON (Octet 2)

Bits		
7 6 5		
<hr/>		
0 0 0	% 0	TON_UNKNOWN
0 0 1	% 1	TON_INTERNATIONAL
0 1 0	% 2	TON_NATIONAL
0 1 1	% 3	TON_LOCAL
1 0 0	% 4	TON_NETWORK

OCN NPI (Octet 3)

Bits		
3 2 1 0		
<hr/>		
0 0 0 0	% 0	NPI_UNKNOWN
0 0 0 1	% 1	NPI_E164
0 0 1 0	% 2	NPI_PRIVATE

OCN SI (Octet 3)

Bits		
5 4		
<hr/>		
0 1	% 1	SI_USER_PROVIDED
1 0	% 2	SI_NETWORK_PROVIDED
1 1	% 3	SI_USER_PROVIDED_VERIF_AND_PASSED

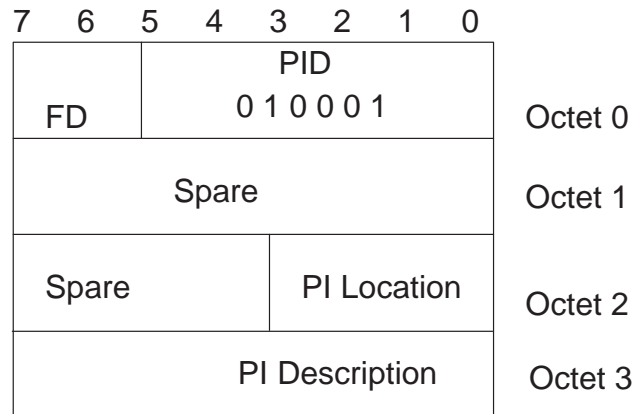
OCN PI (Octet 3)

Bits		
7 6		
<hr/>		
0 1	% 1	PI_PRESENTATION_ALLOWED
1 0	% 2	PI_PRESENTATION_RESTRICTED
1 1	% 3	PI_NOT_AVAILABLE

OCN digits (Octet 4 – 28)

The maximum size of OCN Digits is 24 bytes long. Each OCN Digit is one nibble in size. The OCN Digits are encoded in a nibble swapped format.

Progress indicator



PID (Octet 0)

Bits	
5 4 3 2 1 0	
0 1 0 0 0 1	% 17 PI_PID

Format discriminator (Octet 0)

Bits	
7 6	
0 0	Optional_Parm_FD_V1

PI location (Octet 2)

Bits	
3 2 1 0	
0 0 0 1	User
0 0 1 0	PVT_Network_Serv_local
0 0 1 1	PUB_Network_Serv_local
0 1 0 0	Transit_Network

```

0 1 0 1      PVT_Network_Serv_Remote
0 1 1 0      PUB_Network_Serv_Remote
0 1 1 1      Internationl_Network
1 0 0 0      Beyond_Interworking_Point
1 0 0 1      Interface_Contr_By_This_Sig_Link
    
```

PI description (Octet 3)

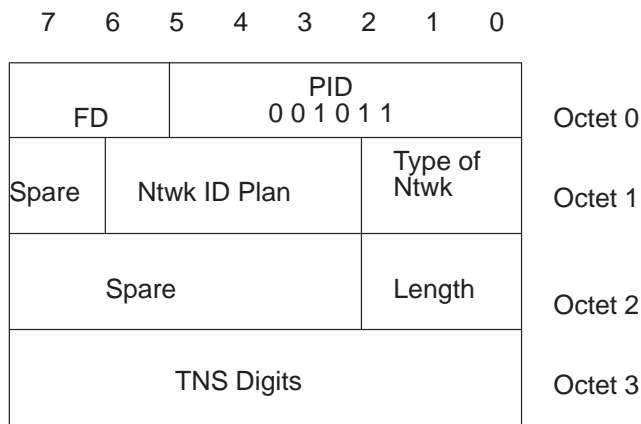
Bits

7 6 5 4 3 2 1 0

```

-----
0 0 0 0 0 0 0 1      % 1 PROG_IND_NOT_END_TO_END_ISDN
0 0 0 0 0 0 1 0      % 2 PROG_IND_IN_BAND_INFO_AVAIL
0 0 0 0 0 0 1 1      % 3 PROG_IND_DEST_NOT_RESP
0 0 0 0 0 1 0 0      % 4 Not Used
0 0 0 0 0 1 0 1      % 5 Not Used
0 0 0 0 0 1 1 0      % 6 Not Used
0 0 0 0 0 1 1 1      % 7 PROG_IND_DEST_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 0      % 8 PROG_IND_ORIG_ADDR_IS_NOT_ISDN
0 0 0 0 1 0 0 1      % 9 PROG_IND_CALL_RETURNED_TO_ISDN
0 0 0 0 1 0 1 0      % 10 Not Used
0 0 0 0 1 0 1 1      % 11 PROG_IND_DELAY_RESP_CALLED_INTERFACE
0 0 0 0 1 1 0 0      % 12 PROG_IND_OVLP_NOT_END_TO_END_ISDN
    
```

Transit network selection



PID (Octet 0)

Bits	
5 4 3 2 1 0	
<hr/>	
0 0 1 0 1 1	% 1 1 TNS_PID

Format discriminator (Octet 0)

Bits	
7 6	
<hr/>	
0 1	Optional_Parm_FD_V2

Type of network (Octet 1)

Bits	
2 1 0	
<hr/>	
0 0 1	% 1 TNS_USER_SPECIFIED
0 1 0	% 2 Not Used
0 1 1	% 3 TNS_NATIONAL
1 0 0	% 4 TNS_INTERNATIONAL

Network ID plan (Octet 1)

Bits	
6 5 4 3	
<hr/>	
0 0 0 1	% 1 UNKNOWN
0 0 1 0	% 2 CARRIER
0 0 1 1	% 3 USER
0 1 0 0	% 4 DATA

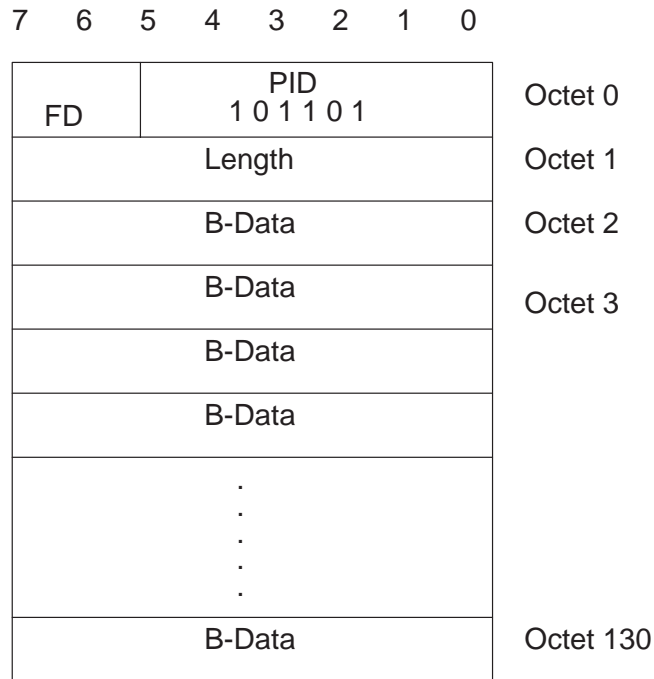
Length (Octet 2)

This parameter is three bits long, which ranges from 1 to 4. It is important to note that this parameter does not relate to the *TNSDigits* parameter.

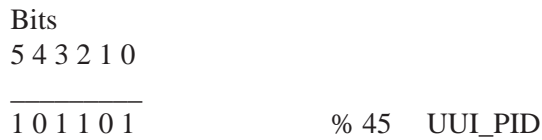
TNS digits (Octet 3)

This parameter is one byte long.

User to user information



PID (Octet 0)



Format discriminator (Octet 0)



Length (Octet 1)

This parameter is defined as one byte, but ranges from 1 to 129.

B-Data (Octet 2–130)

The maximum size of B-data is 129 bytes long.

UCS PSN parameters version 2

This chapter contains a detailed description of the UCS PSN parameters, version 2. These PSN parameters may be a part of a PSN primitive, a PSN event notification, or both. The use of these parameters depend upon the location of the primitives and the event notifications.

PSN peer-to-peer application protocol

A new peer-to-peer application protocol has been created for the PSN platform. This protocol defines the primitives that are provided to allow the SCU to control calls on the PSN, and the event notifications which are reported to the SCU from the PSN. In addition, this protocol defines the parameters and their respective primitives and event notifications.

The peer-to-peer application protocol definition uses generic, functional references to data, rather than specific PSN data requirements. In addition, this protocol, including the primitive and event notification definitions, parameter definitions, and message flow definitions (with service implementation examples), is described later in this document.

PSN parameter definitions

The SCU sends instructions through the respective primitives to the PSN, which enables it to control the service call. In conjunction, the PSN notifies the SCU of the events that occur at the switch. The instructions and event notification messages contain one or more parameters with the appropriate information.

Each parameter has one or more bytes containing the parameter information. The division of parameter contents into fields, and the encoding of these fields, is covered in detail within this chapter.

The primitives or event notification messages in which the parameter may be a part of, is described earlier.

Tables 16-1 through 16-6 illustrates the primitives and event notifications, including the associated parameters. Each parameter is marked with an “M” for mandatory, an “O” for optional, or a “-” if the parameter is not associated with the primitive or event notification.

Table 16-1
Parameters for call control primitives (Part 1)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n i n f o	
Parameters																
<i>AccessType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>BearerCapability</i>	-	-	O	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	O	O	O	O	O	O	O	O	O	O	O	M	O	O	O
<i>CallType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ControlInfo</i>	-	O	O	O	O	O	O	O	-	O	O	O	O	O	O	O
<i>DestinationTrunkGroup</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	M	-	-	-	-	-	-	-	-	M	-	-	-	-	-

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n I n f o
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitstoOutpulse</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	O
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M

Table 16-2
Parameters for call control primitives (Part 2)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n a l I n f o
Parameters															
<i>ISUPIndex</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>MessageInfo</i>	O	-	-	-	-	-	-	-	-	M	M	-	-	M	-
<i>MonitorMask</i>	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	M	M	M	M	M	M	M	M	M	M	M	-	M	M
<i>PortServiceInfo</i>	-	O	O	O	O	O	O	O	M	-	O	O	O	O	O
<i>PortStatus</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ServingTranslatio nScheme</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g I n f o
<i>SessionID</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M
<i>SignalingInfo</i>	-	-	M / O	O	-	-	-	-	-	-	-	-	-	-	O
<i>SignalingType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O
<i>SwitchID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>TimeOfDay</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 16-3
Parameters for event notifications (Part 1)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>AccessType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>CallType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	O	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	O	-	-	-	-	-	-
<i>ControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	M	-	-	-	O	-	-	-	-	-	-
<i>DigitsToOutpulse</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	M	-
<i>ErrorCause</i>	-	M	-	-	-	-	-	-	-	-	-

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l l i n g - E v e n t	T o n e - D e t e c t e d
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	O	-	-	-	-	-	-
<i>InstructionID</i>	-	O	M	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	O	M	M	M	M	M	M	M	M	M

Table 16-4
Parameters for event notifications (Part 2)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>ISUPIndex</i>	-	-	-	-	O	-	-	-	-	-	-
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	O	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	O	M	M	M	M	M	M	M	M	M
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortStatus</i>	-	O	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ServingTranslationScheme</i>	-	-	-	-	O	-	-	-	-	-	-
<i>SessionID</i>	M	O	M	M	-	M	M	M	M	M	M

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
<i>SignalingInfo</i>	-	-	-	-	O	O	O	-	-	M	-
<i>SignalingType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>SwitchID</i>	-	-	-	-	M	-	-	-	-	-	-
<i>Timeofday</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	M

Table 16-5
Parameters for non-call related events and primitives (Part 1)

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
Parameters											
<i>AccessType</i>	-	-	-	-	-	-	-	-	-	-	-
<i>AgentDataInfo</i>	M	-	-	-	-	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>Calltype</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InfoChangeReason</i>	M	-	-	-	-	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	-	-	-	-	-	-	-
CRID	-	-	-	-	-	-	-	-	-	-	-
<i>ControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DestinationTrunkGroup</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-	-	-

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
<i>DigitstoOutpulse</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	O	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	-	M	M	-	-	M	M	M	M	M	-

Table 16-6
Parameters for non-call related events and primitives (Part 2)

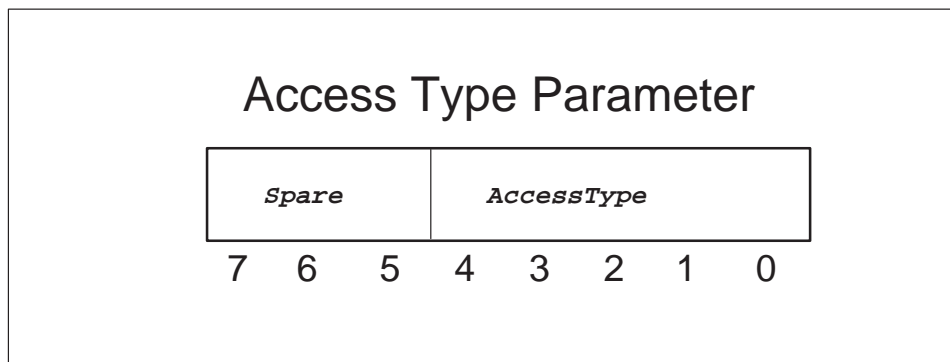
	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n - S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
Parameters											
<i>ISUPIndex</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	-	-	-	-	-	M	M	-	-	-	-
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	-	M / O	M	-
<i>PortStatus</i>	-	-	-	-	-	M	-	-	-	-	-
<i>ResetReason</i>	-	-	-	M	-	-	-	-	-	-	-
<i>ServingTranslationScheme</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SessionID</i>	-	-	-	-	-	O	O	-	-	-	-

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n - S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
<i>SignalingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SignalingType</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SwitchID</i>	M	-	-	-	-	-	-	-	-	-	-
<i>Timeofday</i>	-	M	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	-

Note: There is one mandatory PSI parameter called the “Return IP Address” parameter. It is used to send the **Instruction_Completed**. There are 0 to 20 optional PSI parameters called the IP Address to Reset parameters which are used to do the actual resetting of calls.

Access type

This information is sent in the **New_Call** event notification and contains the originating trunk information. The only values supported for this parameter are FGD, DAL, and PRI.



The *AccessType* parameter is mandatory for a **New_Call** event.

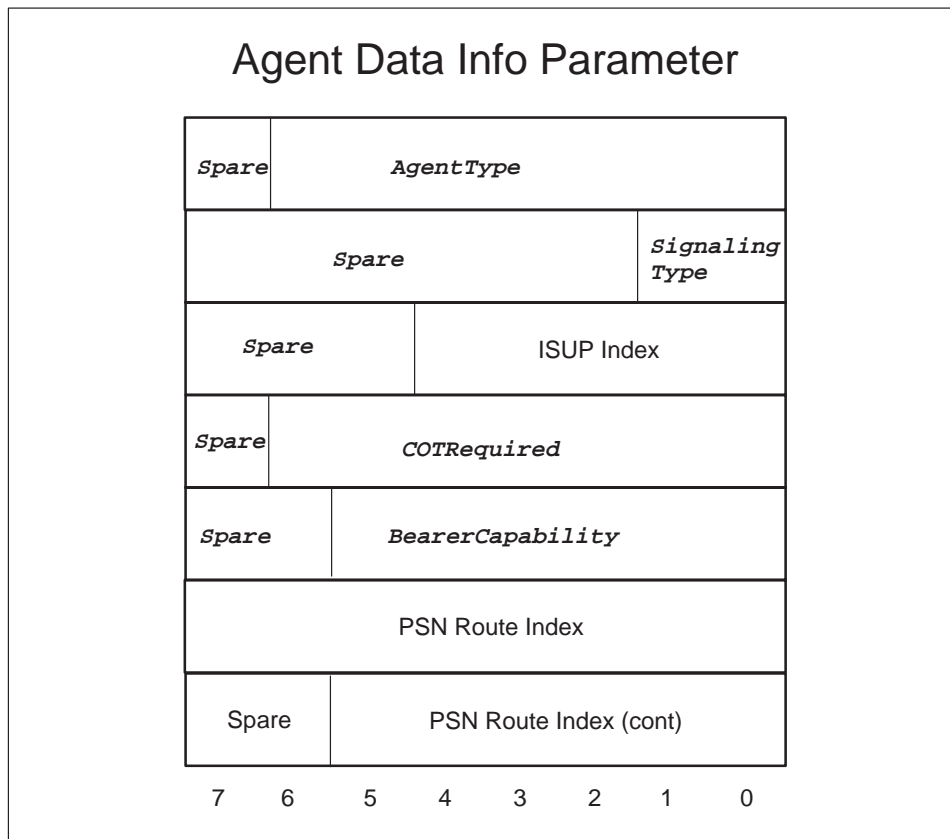
Parameter length: 1 byte

Parameter contents:

- The ACCESS TYPE field consists of three bits and the type of the port, which is encoded as follows:
 - 0 0 0 0: Unused (Spare)
 - 1 0 0 1: PTS FGD
 - 2 0 1 0: SS7 FGD
 - 3 0 1 1: DAL 4-wire
 - 4 1 0 0: DAL 2-wire
 - 5 1 0 1: PRI
 - 1 1 0 to 111: Unused (Spare)

Agent data info

This parameter is only used in the **Agent_Data** event and the **New_Call** event.



Length: 7 bytes

Parameter contents:

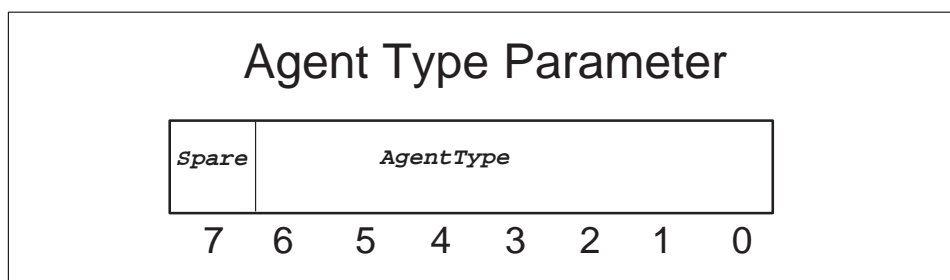
- The AGENT TYPE field consists of seven bits with *AgentType* parameter contents.
- The SIGNALING TYPE field consists of two bits with *SignalingType* parameter contents.
- The ISUP INDEX field consists of five bits with ISUP *Index* parameter contents.
- The COT REQUIRED PERCENT field consists of seven bits with *COTRequired* parameter contents.

- The PULSE TYPE field consists of two bits. This field is the pulse type of the port. Values include:
 - 0 0: Unused (Spare)
 - 0 1: MF
 - 1 0: DTMF
 - 1 1: Unused (Spare)
- The BEARER CAPABILITY field consists of six bits with *BearerCapability* parameter contents.
- The PSNROUTE INDEX consists of 14 bits and is the actual index into the table, PSNROUTE.

Note: This index is 0 when an *Agent_Data* event occurs informing the SCU of a change in only the switch ID. If the new switch ID and current switch ID differ, all other fields of the *AgentDataInfo* parameter should be ignored. At all other times (when no switch ID change occurs), this field has a value from 1 to 9999. When this occurs, all of the *AgentDataInfo* parameter fields may be considered valid.

Agent type parameter

This parameter is used in the *Agent_Data* event and in the *New_Call* event.



Optional parameter ID: 29

Length: 1 byte

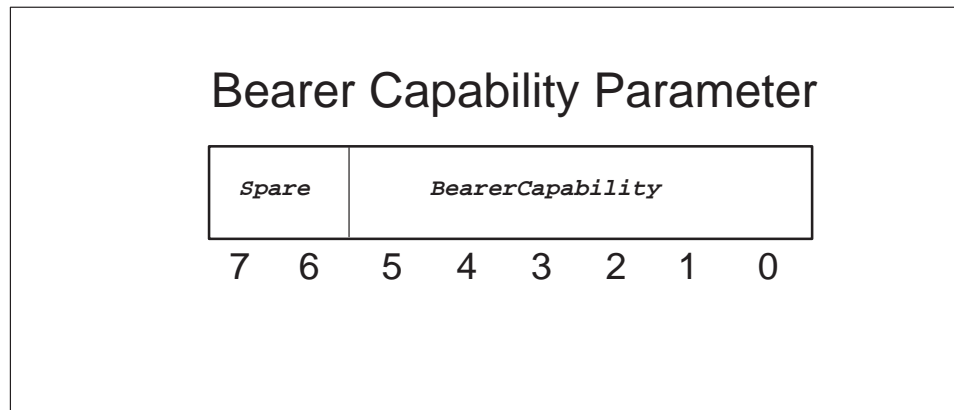
Parameter contents:

- The AGENT TYPE field consists of seven bits and allows the SCU to associate a generic trunk naming convention for each agent in the table, PSNROUTE. The values include:
 - 0 0 0 0 0 0 0: Unused
 - 1 0 0 0 0 0 1: DAL Trunk
 - 2 0 0 0 0 1 0: Unused
 - 3 0 0 0 0 1 1: ONAT Trunk
 - 4 0 0 0 1 0 0: EANT Trunk
 - 0 0 0 0 1 0 1 to 0 0 0 0 1 1 0: Unused

0 0000111: IMT Trunk
 1 0001000: Unused
 2 0001001: OP250 Trunk
 0001010 to 0001100: Unused
 0 0001101: PRA250 Trunk
 0001110 to 1111111: Unused

Bearer capability

The information below describes the *BearerCapability* parameter.



Optional parameter ID: 1

Parameter length: 6 bits

Parameter contents:

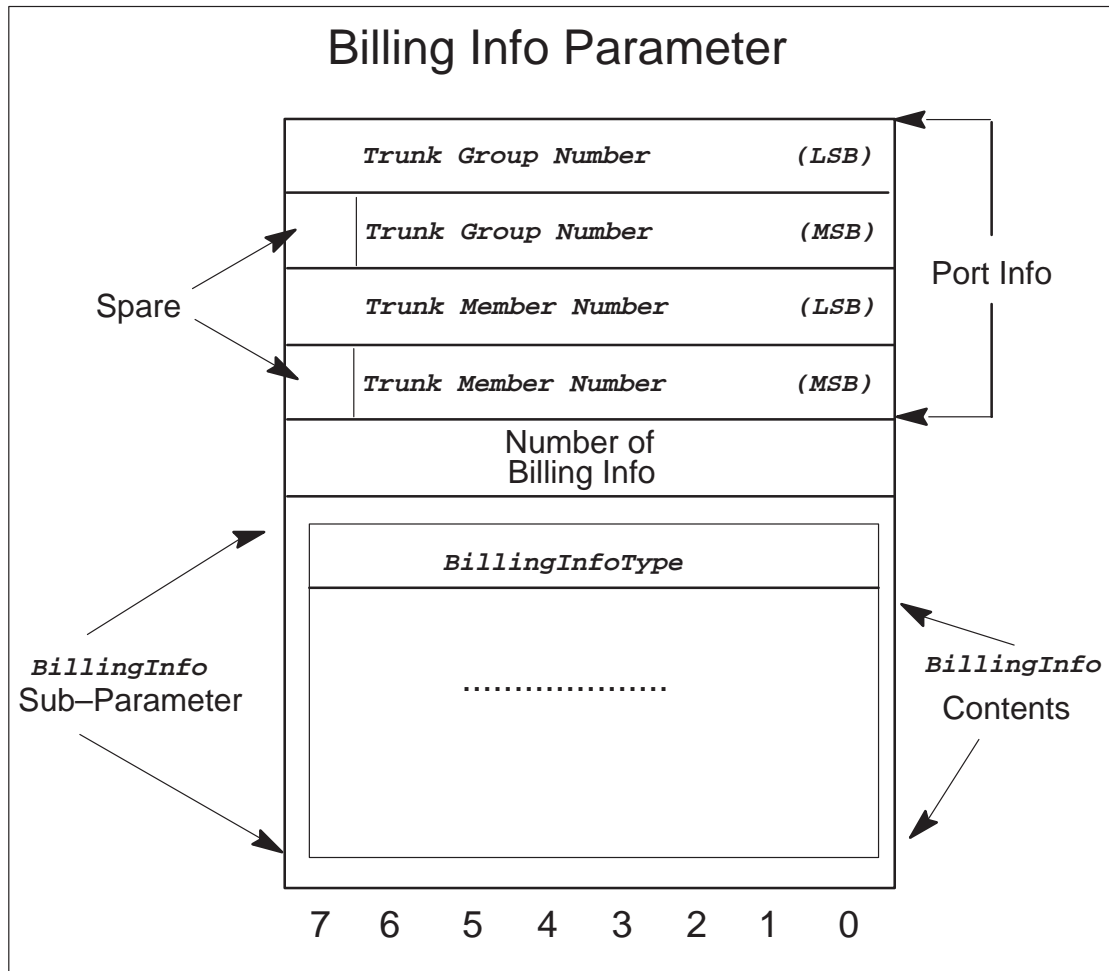
- The BEARER CAPABILITY field consists of six bits and indicates the speed and type of information that is to be carried by the call. The values include:

0 000000: Unused
 1 000001: Speech
 2 000010: 64kbyte Data
 3 000011: 64kbyte X25
 4 000100: 56kbyte Data
 5 000101: Data Unit
 6 000110: 64kbyte Restricted
 7 000111: 3.1 kHz
 8 001000: 7 kHz
 9 001001: Voice Data
 10 001010: 64kbyte Rate Data
 11 001011: 32kbyte Speech
 12 001100: Wideband

0 0 1 1 0 1 to 1 1 1 1 1 1: Spare values

Billing info

This parameter contains the billing information for the port in the service call. In addition, The *BillingInfo* parameter may be used to update multiple billing record fields for a port.



Optional parameter ID: 2

Parameter length: variable in size

Parameter contents:

- The PORT INFO field consists of four bytes and is the port responsible for updated billing records. The external Trunk Group Number and the Trunk Member Number are specified in the Port Info. The values include: 0 – 9999.

- A value of 32,767 for Trunk Group Number indicates a NIL_TRUNK_GROUP. The Trunk Group Number of NIL_TRUNK_GROUP in this parameter is considered invalid.
- A value of 32,767 for Trunk Member Number indicates a NIL_TRUNK_MEMBER. The Trunk Member Number may be a NIL_TRUNK_MEMBER only when the billing information is received for the destination port in the **Connect** primitive.

The following table illustrates how to interpret the combination of Trunk Group Number and Trunk Member Number in the *BillingInfo* parameter.

Trunk Group Number	Trunk Member Number	What is the Billing Info used for
nil_trunk_group	nil_trunk_member	Invalid
Non nil_trunk_group	nil_trunk_member	<i>BillingInfo</i> for destination trunk group in Connect primitive only.
nil_trunk_group	Non nil_trunk_member	Invalid
Non nil_trunk_group	Non nil_trunk_member	<i>BillingInfo</i> for the port specified by the trunk group/member.

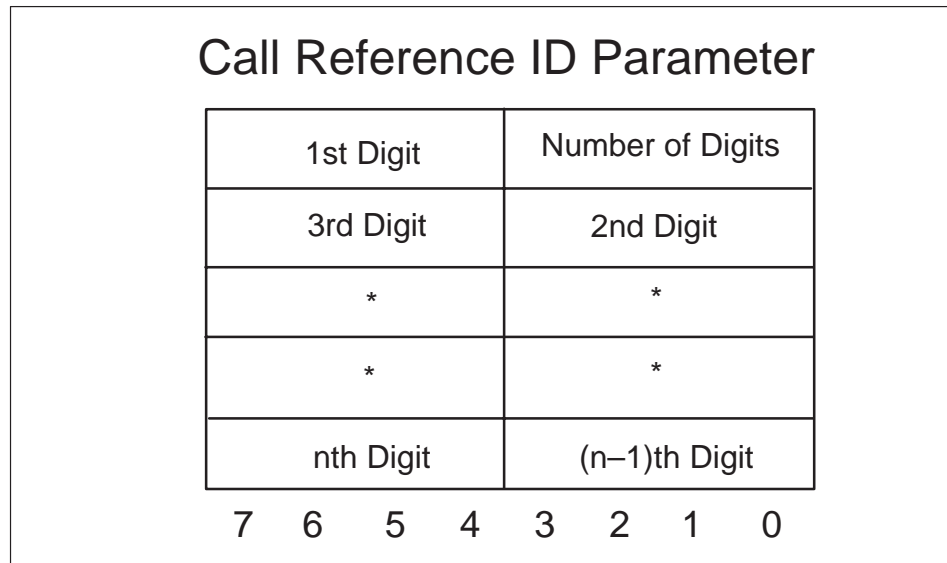
- The NUMBER OF BILLING INFO field is one byte and determines the Number of Billing Info sub-parameters to follow. Each Number of Billing Info sub-parameter consists of the Billing Info Type and the Billing Info contents. The maximum number of Billing Info in Billing Info Parameter is one.
 - 0 00000000 : Zero Number of Billing Info
 - 1 00000001 : One Billing Info
 - 00000010 to 11111111 : Spare Values
- The BILLING INFO TYPE field is one byte and indicates how to interpret the Billing Info contents. Each application may define its own set of types. As a result, the interpretation of the Billing Info contents depends upon the type.
 - 0 00000000: Unused
 - 1 00000001: CRID
 - 00000010 to 11111111: Spare values
- The BILLING INFO CONTENTS field contains the billing information that is ultimately placed in the billing record.

This parameter supports the following billing type:

Call Reference ID (CRID)

Call reference ID (CRID)

This Parameter is used for the transport of the Call Reference Identifier. The CRID digits are encoded in a variable length digits field in TBCD format. The valid range of digits for the *CRID* parameter is 1 to 9 digits.



Optional parameter ID: 3

Parameter length: Variable in Size– Maximum of 5 bytes

Parameter contents:

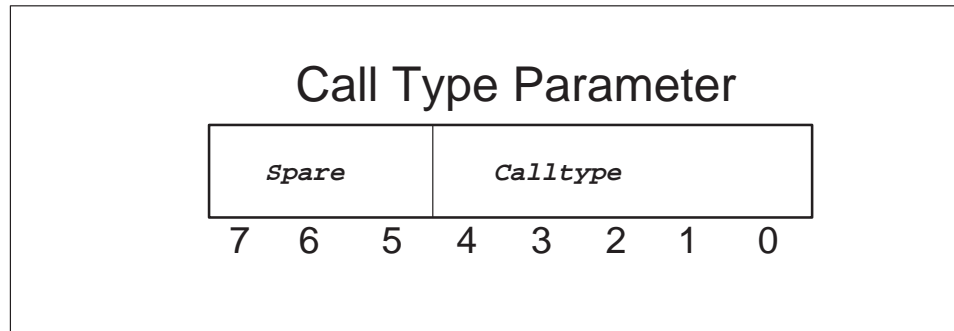
- The NUMBER OF DIGITS field consists of four bits and contains the number of CRID digits. The values include: 1 to 9.
- The 1ST DIGIT to 9TH DIGIT field consists of four bits (each digit) with all 9 digits in TBCD format, which are encoded as follows:

0 0 0 0: Filler
 1 0 0 1: Digit 1
 2 0 0 1 0: Digit 2
 3 0 0 1 1: Digit 3
 4 0 1 0 0: Digit 4
 5 0 1 0 1: Digit 5
 6 0 1 1 0: Digit 6
 7 0 1 1 1: Digit 7
 8 1 0 0 0: Digit 8
 9 1 0 0 1: Digit 9
 10 1 0 1 0: Digit 0

11 1 0 1 1: *
 12 1 1 0 0: #
 13 1 1 0 1: D
 14 1 1 1 0: E
 15 1 1 1 1: F

Call type

This parameter contains the Call Type for a call when it is determined by the PSN that the call is to be controlled by the SCU.



Mandatory parameter for a **New_Call** event.

Parameter length: 1 byte

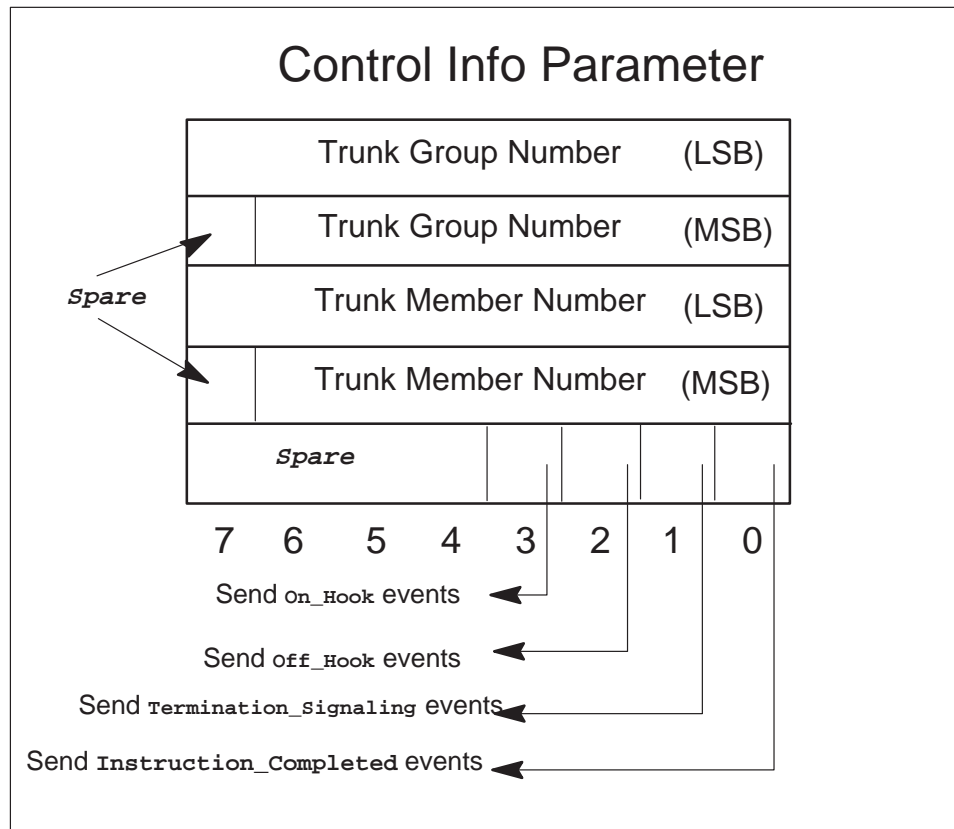
Parameter contents:

- The CALL TYPE field consists of 5 bits provides information on the type of call being made. The values include:
 - 0 0 0 0 0: Undetermined *
 - 1 0 0 0 1: Onnet *
 - 2 0 0 0 1 0: Offnet *
 - 3 0 0 0 1 1: Public Speed
 - 4 0 0 1 0 0: Private Speed
 - 5 0 0 1 0 1: Hotline Speed
 - 6 0 0 1 1 0: N00*
 - 7 0 0 1 1 1: Zero Plus – Onnet
 - 8 0 1 0 0 0: Zero Plus – Offnet
 - 9 0 1 0 0 1: INTOA *
 - 0 1 0 1 0 to 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

Control info parameter

This parameter contains the action to be taken if the primitive is successfully processed.



Optional parameter ID: 4

Parameter length: 5 Bytes

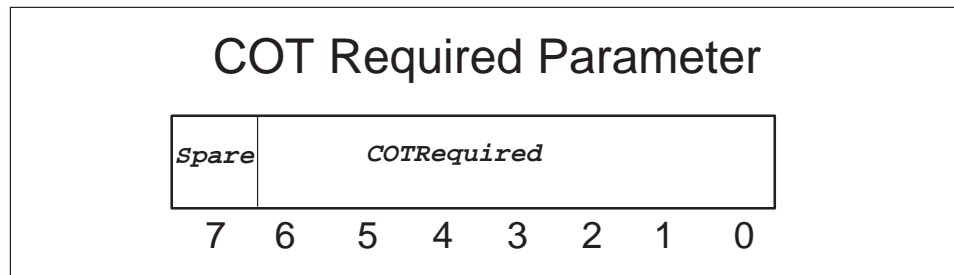
Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the external Trunk Group Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_GROUP.
- The TRUNK MEMBER NUMBER field consists of 15 Bits and contains the Trunk Member Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_MEMBER.
- The SEND INSTRUCTION COMPLETED EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any events that occur on the agent. The values include:
 - 0 (OFF) Do not send any `Instruction_Completed` events.
 - 1 (ON) Send all `Instruction_Completed` events.

- The SEND TERMINATION SIGNALING EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any `Termination_signaling` events (PTS Digits Outpulsed, SS7 ACM, and PRI PROGRESS) that occur on the agent. The values include:
 - 0(OFF) Do not send any `Connect_signaling` events.
 - 1 (ON) Send all `Connect_signaling` events.
- The SEND OFFHOOK EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any `off_Hook` events that occur on the agent. The values include:
 - 0 (OFF) Do not send any `off_Hook` events.
 - 1 (ON) Send all `off_Hook` events.
- The SEND ONHOOK EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any `On_Hook` events that occur on the agent. The values include:
 - 0 (OFF) Do not send any `On_Hook` events.
 - 1 (ON) Send all `On_Hook` events.

COT required parameter

This parameter is used in the `Agent_Data` event and in the `New_Call` event.



Optional parameter ID: 30

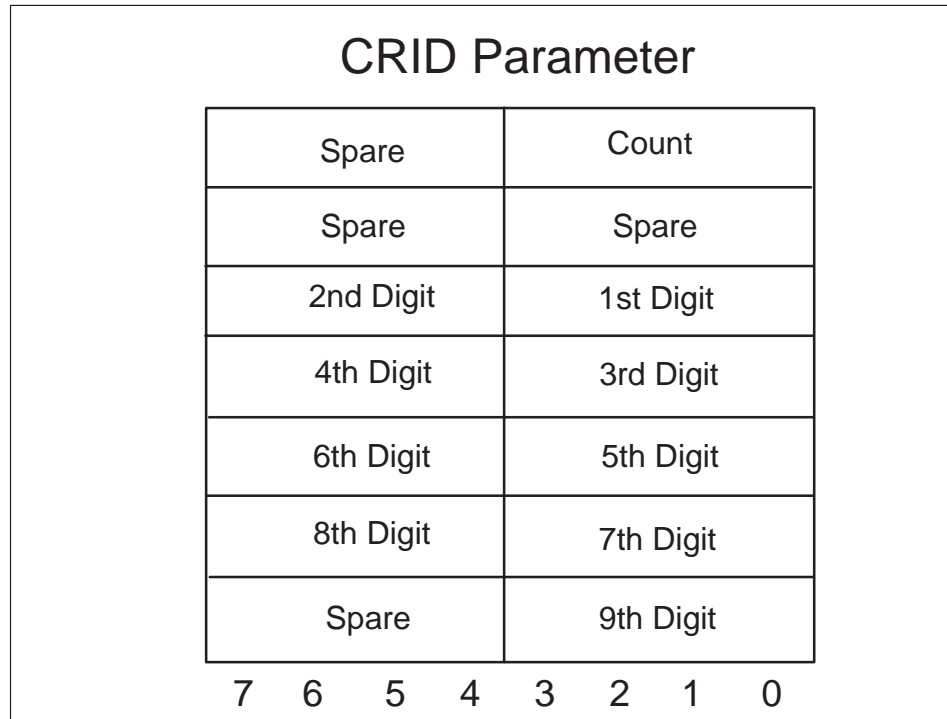
Length: 1 byte

Parameter contents:

- The COT REQUIRED PERCENT field consists of seven bits and represents the percentage of the SS7 PSN agent's trunk members, which are datafilled to have COT testing performed on them. The range of this parameter is from 0, for no COT testing to 100, for COT testing on each member.

CRID Parameter

This parameter is used in the New Call Event.



Optional parameter ID: 30

Parameter length: 7 bytes

Parameter contents:

- The COUNT field indicates the number of actual CRID digits stored within the parameter.
- The 1ST DIGIT to 9TH DIGIT field consists of four bits (each digit) of n digits. These DTMF digits are encoded in the TBCD format as follows:

```

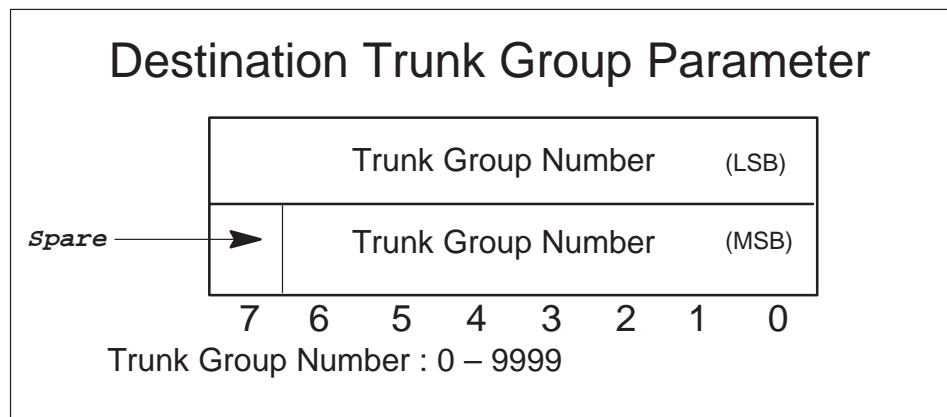
0  0 0 0 0: Filler
1  0 0 0 1: Digit 1
2  0 0 1 0: Digit 2
3  0 0 1 1: Digit 3
4  0 1 0 0: Digit 4
5  0 1 0 1: Digit 5
6  0 1 1 0: Digit 6
7  0 1 1 1: Digit 7

```

8 1 0 0 0: Digit 8
 9 1 0 0 1: Digit 9
 10 1 0 1 0: Digit 0
 11 1 0 1 1: *
 12 1 1 0 0: #
 13 1 1 0 1: D
 14 1 1 1 0: E
 15 1 1 1 1: F

Destination trunk group

This parameter contains the external Trunk Group Number of the intended trunk. The call terminates to an idle member of this trunk group.



Optional parameter ID: 5

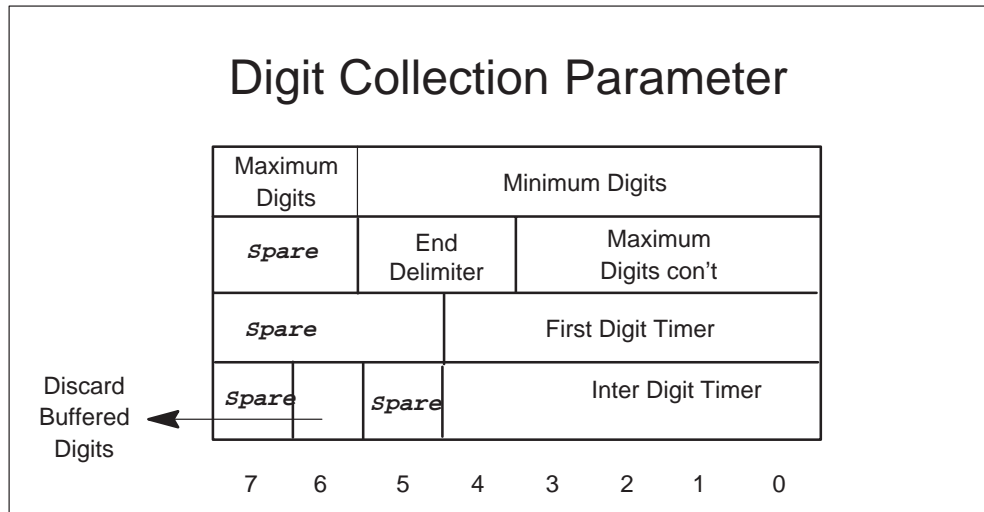
Parameter length: 2 bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the location of the external Trunk Group Number. The range is from 0 to 9999. The NIL_TRUNK_GROUP number is defined as 32,767.

Digit collection parameter

This parameter contains information required to collect digits on a specified port.



Optional parameter ID: 6

Parameter length: 4 bytes

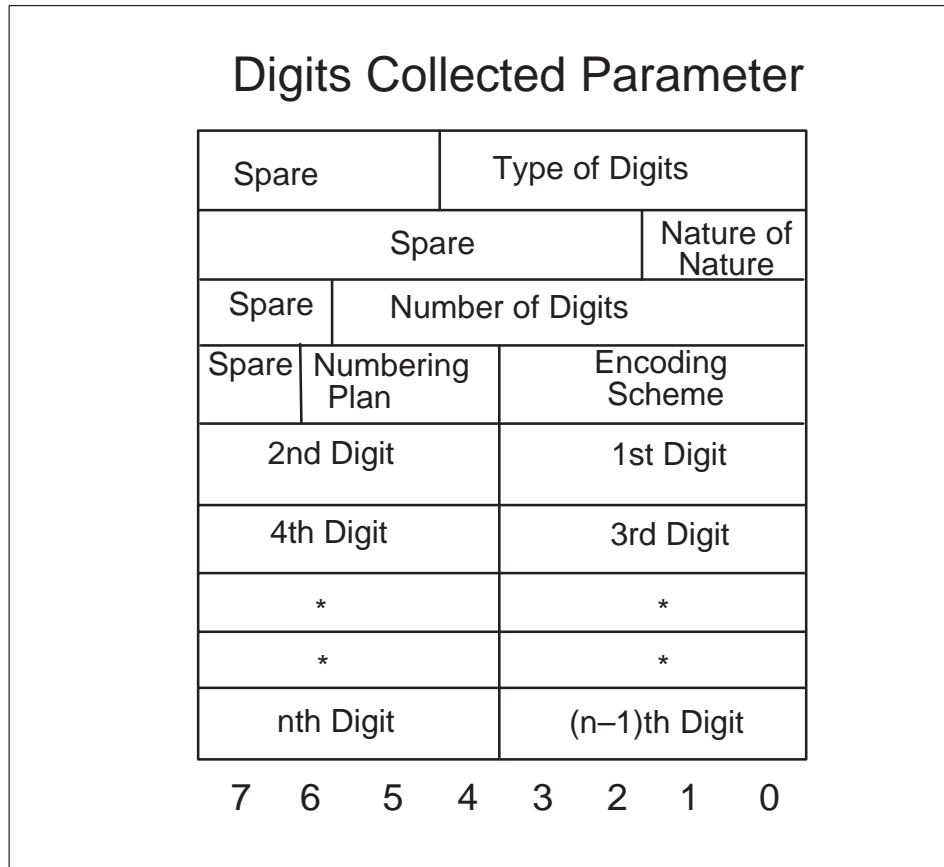
Parameter contents:

- The MINIMUM_DIGITS field consists of six bits and contains the minimum number of digits to collect. This value must be less than or equal to the MAXIMUM_DIGITS field. The values include the following (0 – 45):
 - 0 0 0 0 0 0: (Minimum Number of Digits to collect equals 0)
 - 0 0 0 0 0 1
 -
 - 1 0 1 1 0 1: (Minimum Number of Digits to collect equals 45)
 - 1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (spare)
- The MAXIMUM_DIGITS field consists of six bits and contains the Maximum Number of Digits to collect. The values include the following (0 – 45):
 - 0 0 0 0 0 0: (Maximum Number of Digits to collect equals 0)
 - 0 0 0 0 0 1
 -

-
- 1 0 1 1 0 1: (Maximum Number of Digits to collect equals 45)
 - 1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (spare)
 - The END_DELIMITER field consists of two bits and identifies the End of Digits Delimiter. Upon detection of the End of Digits Delimiter, digit collection stops and the collected digits are reported. If the delimiter is dialed as the very first digit, then the digit collection stops immediately. The delimiter digit values include:
 - 0 0: No Delimiter Specified
 - 0 1: *
 - 1 0: #
 - 1 1: * and #
 - The FIRST_DIGIT_TIMER field consists of five bits and specifies the time to wait until the first digit is dialed. The values include the following (2 – 30 seconds):
 - 0 0 0 0 0: Unused (spare)
 - 0 0 0 0 1: Unused (spare)
 - 0 0 0 1 0: Timer Value equals 2 second
 -
 - 1 1 1 1 0: Timer Value equals 30 seconds
 - 1 1 1 1 1: Unused (spare)
 - The INTER_DIGIT_TIMER consists of five bits and specifies the Inter Digit Timer value, for example, the time to wait when collecting the second and subsequent digits. The values include the following (2 – 20 seconds):
 - 0 0 0 0 0: Unused (spare)
 - 0 0 0 0 1: Unused (spare)
 - 0 0 0 1 0: Timer value equals 2 second
 -
 - 1 0 1 0 0: Timer Value equals 20 seconds
 - 1 0 1 0 1 to 11111: Unused (Spare)
 - The DISCARD BUFFERED DIGITS field consists of one bit and is a bool, which if true, indicates that if the PSN had been buffering any digits prior to receiving this parameter, then it is to discard these digits and restart the digit collection process. Refer to Chapter “PSN finite state machine” for details on the buffering of digits.

Digits collected

This parameter contains the Digits Collected (or received) on the port or agent. The digits contained in this parameter are always in the TBCD format. Also included is COUNT, which specifies the number of digits included in this parameter.



Optional parameter ID: 7

Parameter length: Variable in size

Parameter contents:

Note: The contents of this parameter (including the Nature of Number, Numbering Plan, and Encoding Scheme fields) have been encoded in accordance with the following documents: TR-NWT-000317 Switching Systems Requirements for Call Control Using ISDNUP, TR-NWT-000394 Switching Systems Requirements for IEC Interconnection using ISDNUP, and TR-NWT-000444 Switching System Requirements Supporting ISDN Access Using the ISDNUP. However, there are some values marked “Private” which are application specific.

- The TYPE OF DIGITS field consists of five bits and contains the Type of Digits encoded as shown below. The Type of Digits is known when the collected digits are sent in the **New_Call** event notification. When this parameter is sent in the **Digit_Collected** event notification, a value of “Unknown” is used.

0	0 0 0 0 0:	Unknown
1	0 0 0 0 1:	Called Party Address
2	0 0 0 1 0:	Calling Party Address (ANI)
3	0 0 0 1 1:	Caller Interaction
4	0 0 1 0 0:	Routing Number
5	0 0 1 0 1:	Billing Number
6	0 0 1 1 0:	Destination Number
7	0 0 1 1 1:	Local Access and Transport Area (LATA)
8	0 1 0 0 0:	Carrier Identification
9	0 1 0 0 1:	Referral Number (Private)
10	0 1 0 1 0:	True Billing Number (Private)
11	0 1 0 1 1:	Alternate Preferred Carrier (Private)
12	0 1 1 0 0:	Preferred INC (Private)
13	0 1 1 0 1:	Primary Preferred Carrier (Private)
14	0 1 1 1 0:	Personal Identification Number (PIN)
15	0 1 1 1 1:	Authorization Code (Private)
16	1 0 0 0 0:	TCM (Private)
17	1 0 0 0 1:	Second Alternate Preferred Carrier (Private)
18	1 0 0 1 0:	Business Customer ID (Private)
19	1 0 0 1 1:	Hop-off Office (Private)
20	1 0 1 0 0:	Outpulse Number (Private)
21	1 0 1 0 1:	Originating Station (DN)
22	1 0 1 1 0:	MCCS Card Number
23	1 0 1 1 1:	Account Code Number
24	1 1 0 0 0:	COSOVE Number
25	1 1 0 0 1:	Generic Digits Number
26	1 1 0 1 0:	Dialed Digits Number
27	1 1 0 1 1:	Facility Code
28	1 1 1 0 0:	Country Code
29	1 1 1 0 1:	STS Digits
30	1 1 1 1 0:	OPart Digits
31	1 1 1 1 1:	TPart Digits

- The NATURE OF NUMBER field consists of two bits and is encoded as follows:

0	0 0:	Not Applicable
1	0 1:	International
2	1 0:	National
3	1 1:	Network Specific

- The NUMBER OF DIGITS field is 6 bits and contains the number of digits that are sent in this parameter. This number may be as low as 0 and as high as 45.
- The ENCODING SCHEME field consists of four bits and contains the scheme used to encode the digits that are sent in this parameter. The following illustrates how the Encoding Scheme is encoded:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Binary Coded Decimal (BCD)
 - 0 0 1 0 to 1 1 0 1: Spare Values
 - 14 1 1 1 0: Telephony Binary Coded Decimal (TBCD)
 - 1 1 1 1: Spare
- The NUMBERING PLAN field consists of three bits. The following illustrates how the Numbering Plan is encoded:
 - 0 0 0 0: Unknown or Not Applicable
 - 1 0 0 1: ISDN Numbering Plan (E.164)
 - 2 0 1 0: Telephony Numbering Plan (E.163)
 - 3 0 1 1: Data Numbering Plan (X.121)
 - 4 1 0 0: Telex Numbering Plan (F.69)
 - 5 1 0 1: Maritime Mobile Numbering Plan (E.120, 211)
 - 6 1 1 0: Land Mobile Numbering Plan (E.212, 213)
 - 1 1 1 Spare Value
- The 1ST DIGIT to NTH DIGIT field consists of four bits per each digit and contains n digits. These DTMF digits may be encoded in the BCD format as follows:
 - 0 0 0 0: Digit 0
 - 1 0 0 1: Digit 1
 - 2 0 0 1 0: Digit 2
 - 3 0 0 1 1: Digit 3
 - 4 0 1 0 0: Digit 4
 - 5 0 1 0 1: Digit 5
 - 6 0 1 1 0: Digit 6
 - 7 0 1 1 1: Digit 7
 - 8 1 0 0 0: Digit 8
 - 9 1 0 0 1: Digit 9
 - 10 1 0 1 0: Filler
 - 11 1 0 1 1: *
 - 12 1 1 0 0: #
 - 13 1 1 0 1: D
 - 14 1 1 1 0: E
 - 15 1 1 1 1: F

Or the DTMF Digits may be encoded in the TBCD format as follows:

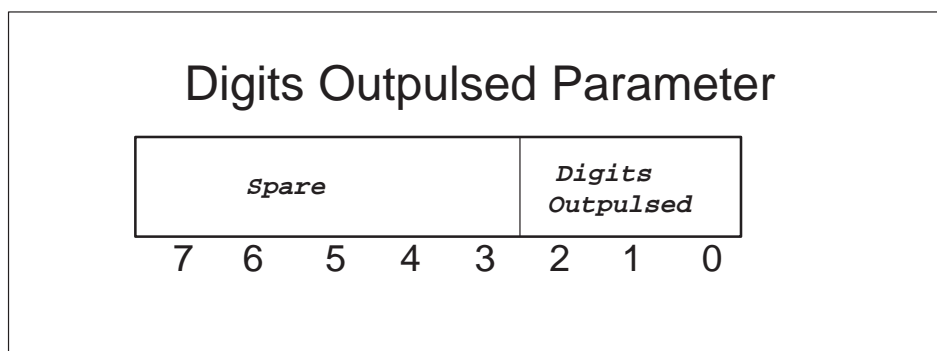
```

0 0 0 0 0: Filler
1 0 0 0 1: Digit 1
2 0 0 1 0: Digit 2
3 0 0 1 1: Digit 3
4 0 1 0 0: Digit 4
5 0 1 0 1: Digit 5
6 0 1 1 0: Digit 6
7 0 1 1 1: Digit 7
8 1 0 0 0: Digit 8
9 1 0 0 1: Digit 9
10 1 0 1 0: Digit 0
11 1 0 1 1: *
12 1 1 0 0: #
13 1 1 0 1: D
14 1 1 1 0: E
15 1 1 1 1: F

```

Digits outpulsed

This parameter indicates information on the digits that were outpulsed on a PTS trunk agent. This is especially useful in multi-stage outpulsing. This parameter is included in the **signaling** Event notification message to let the SCU know that all the digits were outpulsed on the trunk agency.



Optional parameter ID: 8

Parameter length: 1 byte

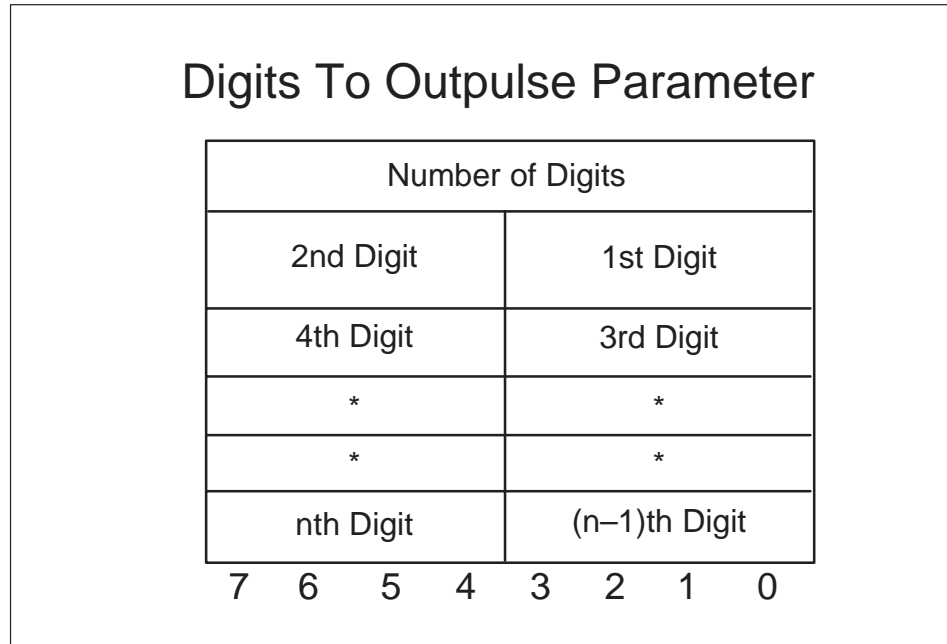
Parameter contents:

- The OUTPUTSED field consists of three bits and is encoded as follows:
 - 0 0 0: Unknown

- 0 0 1: All Streams Outputpulsed
- 0 1 0 to 1 1 1: Spare

Digits to outputse

This parameter contains the digits to be outputpulsed on a PTS trunk agent.



Optional parameter ID: 9

Parameter length: Variable in size

Parameter contents:

- The NUMBER OF DIGITS consists of one byte and contains the number of digits that are to be outputpulsed on this port.
The maximum number of digits that may be outputpulsed is 23. The range for this field is from 1 to 23.
- The 1ST DIGIT to NTH DIGIT field consists of four bits per each digit and contains n digits.

DTMF digits are encoded as follows:

- 0 0 0 0: Filler
- 1 0 0 1: Digit 1

2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: *
12 1100: #
13 1101: D
14 1110: E
1111: F

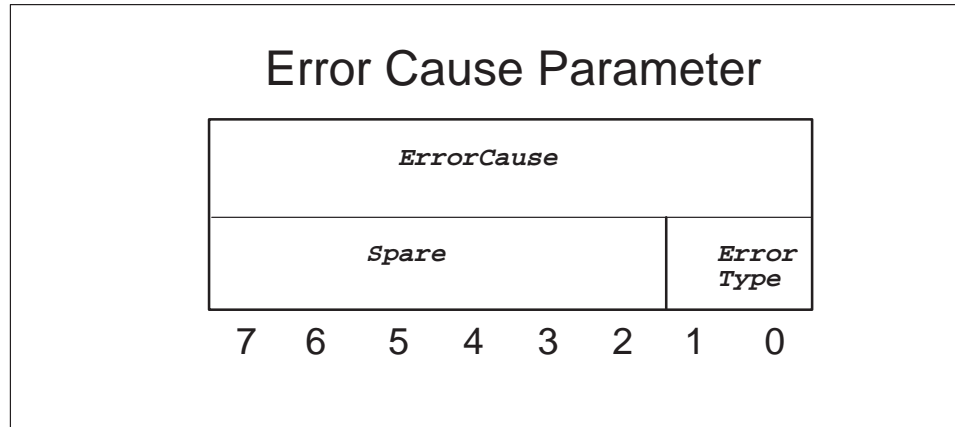
MF digits are encoded as follows:

0 0000: Filler
1 0001: Digit 1
2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: KP3 and ST3P
12 1100: KPP and STP
13 1101: KP and STKP
14 1110: KP2 and ST2P
1111: ST

If multiple “*DigitstoOutputpulse*” parameters are contained in a message, then multiple streams of digits are outputted on the agent or port with each stream contained in one parameter of type “*DigitstoOutputpulse*”.

Error cause

This parameter contains the cause of an error that is detected on the PSN.



Optional parameter ID: 10

Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of one byte and is used to represent the cause of the error. The values include:
 - 0 00000000: Nil Error Cause
 - 1 00000001: Header decode failure
 - 2 00000010: Bad macro tag
 - 3 00000011: Unrecognized primitive
 - 4 00000100: Missing mandatory parameter
 - 5 00000101: Mandatory parameter decode failure
 - 6 00000110: Optional parameter decode failure
 - 7 00000111: Parameter contents out of range
 - 8 00001000: Primitive userclass mismatch
 - 9 00001001: Maximum primitive exceeded
 - 10 00001010: Missing mandatory SigInfo parameter
 - 11 00001011: One or more agents in the primitive are not PSN agents
 - 12 00001100: Port not in table PSNROUTE
 - 13 00001101: Agent not supported
 - 14 00001110: Port down due to WARM restart
 - 15 00001111: Primitive invalid for current port state
 - 16 00010000: Unexpected message

17 00010001: STR not available (affects the **Monitor** primitive)
 18 00010010: UTR not available
 19 00010011: Conference circuit not available
 20 00010100: No IDLE message
 21 00010101: Primitive extension block not available
 22 00010110: Scratchpad extension block not available
 23 00010111: Software Resources unavailable
 24 00011000: Message failure
 25 00011001: Software error
 26 00011010: Not minimum number ports to Bridge
 27 00011011: Maximum ports to Bridge exceeded
 28 00011100: Bearer Capability incompatible
 29 00011101: Message index not in table PSNMSGIX
 30 00011110: Unsupported signaling type
 31 00011111: Duplicate message
 32 00100000: Bad Agent state
 33 00100001: Termination failure
 34 00100010: Abnormal Exit
 35 00100011: Message not playing
 36 00100100: Tone duration unsupported
 37 00100101: Prompt failure
 38 00100110: Digit collection failure
 39 00100111: Q764 protocol problem
 40 00101000: Invalid duration gap
 41 00101001: Unexpected FC message
 42 00101010: Agent not in Table TRKGRP

- The ERROR TYPE field consists of two bits and contains the type of the error that is detected. The following illustrates how the Error Type field is encoded:
 - 00: Non Fatal Error
 - 01: Fatal Error

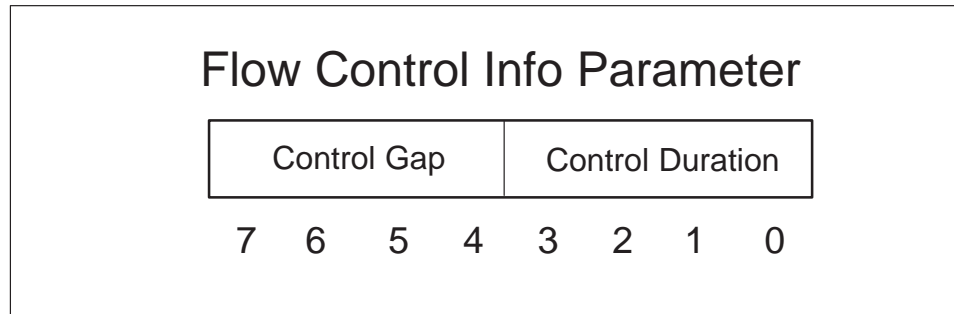
Fatal errors are defined as errors that are so severe that they do not allow normal call processing to proceed on this port.

If the error that is detected is non-fatal, then the PSN does not take any action other than report this error to the SCU. If the error that is detected is fatal, then the PSN reports the error to the SCU. In addition, the PSN takes down the associated port when a fatal error is detected.

Flow control info parameter

This parameter consists of a control duration, which specifies the maximum amount of time the Flow Control Info is effective at the PSN. In addition, this parameter consists of a control gap, which specifies the maximum rate

at which the **New_Call** event notifications may be sent to the SCU while the control is effective.



Optional parameter ID: 11

Parameter length: 1 byte

Parameter contents:

The CONTROL DURATION field consists of four bits and indicates the maximum amount of time that the Flow Control is effective at the PSN. The Control Duration is encoded as follows:

0	0 0 0 0:	Not Used
1	0 0 0 1:	1 Second
2	0 0 1 0:	2 Seconds
3	0 0 1 1:	4 Seconds
4	0 1 0 0:	8 Seconds
5	0 1 0 1:	16 Seconds
6	0 1 1 0:	32 Seconds
7	0 1 1 1:	64 Seconds
8	1 0 0 0:	128 Seconds
9	1 0 0 1:	256 Seconds
10	1 0 1 0:	512 Seconds
11	1 0 1 1:	1024 Seconds
12	1 1 0 0:	2048 Seconds
	1 1 0 1 – 1 1 1 1:	Spare Values

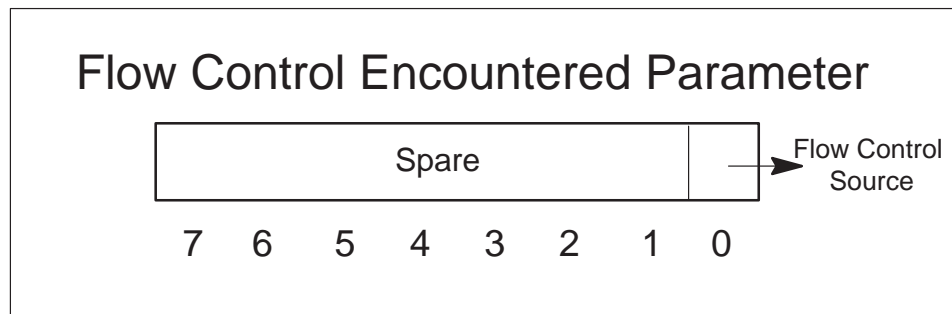
The CONTROL GAP field consists of four bits and indicates the maximum rate at which the **New_Call** event notifications may be sent to the SCU while the Flow Control is effective. The Control Gap is encoded as follows:

0	0 0 0 0:	Remove Gap Control
---	----------	--------------------

1	0 0 0 1:	1 1/10th of a Second
2	0 0 1 0:	3 1/10ths of a Second
3	0 0 1 1:	5 1/10ths of a Second
4	0 1 0 0:	1 Second
5	0 1 0 1:	2 Seconds
6	0 1 1 0:	5 Seconds
7	0 1 1 1:	10 Seconds
8	1 0 0 0:	15 Seconds
9	1 0 0 1:	30 Seconds
10	1 0 1 0:	50 Seconds
11	1 0 1 1:	80 Seconds
12	1 1 0 0:	120 Seconds
13	1 1 0 1:	300 Seconds
14	1 1 1 0:	600 Seconds
15	1 1 1 1:	Stop All Calls

Flow control encountered parameter

This parameter is sent within **New_Call** events, which are then sent to the SCU while the Flow Control is active. In addition, this parameter informs the SCU that the Flow Control is active and identifies the source that initiated the Flow Control.



Optional parameter ID: 12

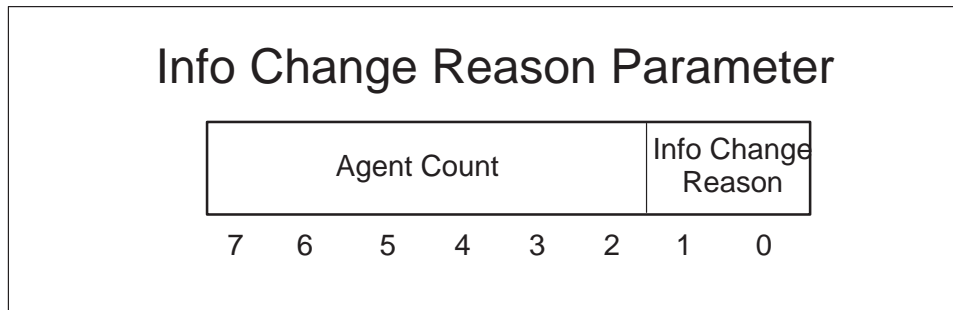
Parameter length: 1 byte

Parameter contents:

- The FLOW CONTROL SOURCE field consists of one bit and is the source that initiated the Flow Control. The Flow Control Source is encoded as follows:
 - 0: Flow Control initiated by the SCU
 - 1: Flow Control initiated by the PSN

Info change reason parameter

This parameter contains the Info Change Reason and Agent Count information used in **Agent_Data** event notifications.



Parameter length: 1 byte

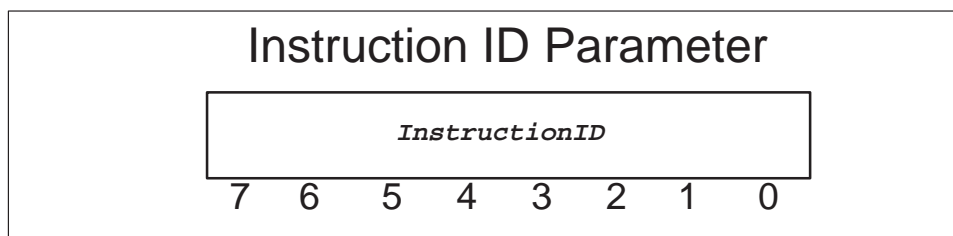
Parameter contents:

- The INFO CHANGE REASON field consists of two bits and indicates what type of **Agent_Data** event is occurring for the given message. The values include:
 - 0 0: Agent Deleted in table PSNROUTE
 - 0 1: Agent Added in table PSNROUTE
 - 1 0: Agent information Modified in table TRKGRP or TRKSGRP
- The AGENT COUNT field consists of six bits and indicates the number of **AgentDataInfo** parameters that exist in the given **Agent_Data** event message. The values include:

0 0 0 0 1 – 1 1 1 1 1 1

Instruction ID

This parameter contains the Identifier of the primitive that is received from the SCU.



Optional parameter ID: 13

Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION ID field consists of one byte which is used to represent the primitive instruction. The values include:

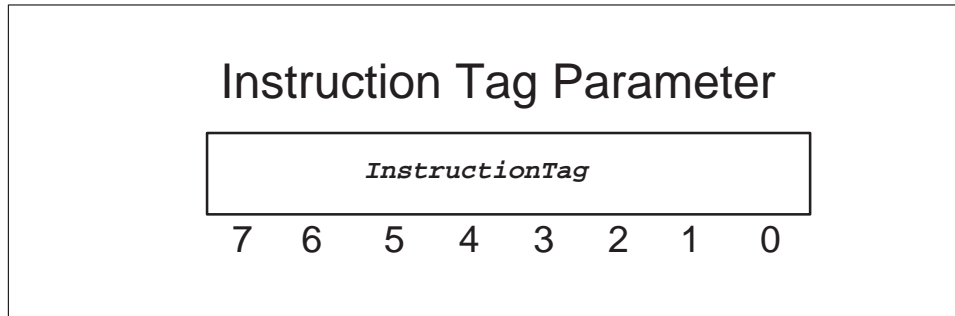
0	00000000	: Unknown
1	00000001	: Bridge
2	00000010	: Collect Digits & Report
3	00000011	: Connect
4	00000100	: Disconnect
5	00000101	: Hold
6	00000110	: Monitor
7	00000111	: Mute
8	00001000	: New Call Accepted
9	00001001	: New Call Rejected
10	00001010	: Play Message
11	00001011	: Play Prompt, Collect Digits & Report
12	00001100	: Query Port
13	00001101	: Reconnect
14	00001110	: Reset Switch
15	00001111	: Set Billing Record
16	00010000	: Set IP Address
17	00010001	: Stop Message
18	00010010	: Transmit SigInfo
19	00010011	: Heartbeat
20	00010100	: Query Time of Day
21	00010101	: Error Detected
22	00010110	: Port Status
23	00010111	: Flow Control
	00011000 – 11111111	: Spare Values

Instruction tag

This parameter contains the tag associated with the instruction.

- For primitives that are sent from the SCU, the SCU generates an Instruction tag and then sends the Instruction tag to the PSN within this parameter. When a response for this primitive is generated by the PSN, the PSN includes this tag in the response, which enables the SCU to correlate the response with the primitive request that it had sent earlier.
- For asynchronous event notifications generated by the PSN, such as `On_Hook`, `Off_Hook` and `signaling` event, the Instruction tag is hard-coded to #01.

- For instructions that are generated by the PSN that require a reply from the SCU, such as New Call and Query Port from the Audit application, the PSN sends a nil Instruction tag.



Optional parameter ID: 14

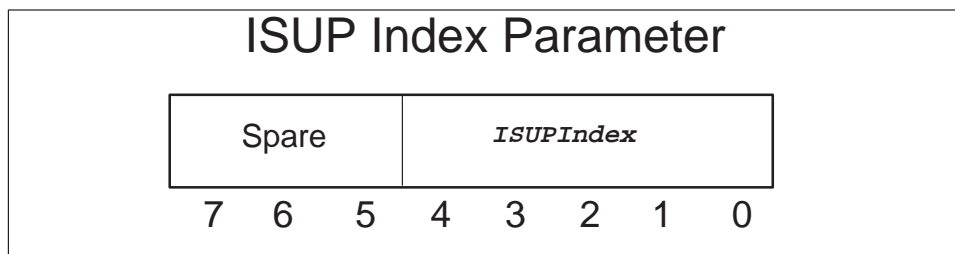
Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION TAG field consists of one byte, which is used to represent the tag of the instruction that is received from or sent to the SCU. The values include:
 - 0 0 0 0 0 0 0 0: Nil Instruction Tag
 - 0 0 0 0 0 0 0 1: Asynchronous Event Notification from the PSN.
 - 0 0 0 0 0 0 1 – 1 1 1 1 1 1 1 1: Valid Tags for Instructions from the SCU.

ISUP index

This parameter is used in the `Agent_Data` event and in the `New_Call` event.



Optional parameter ID: 31

Parameter length: 1 byte

Parameter contents:

- The ISUP INDEX field consists of five bits and represents the type of facility that the SS7 agent connects to. Please note that this field is only valid for SS7 signaling types. The values include:

```

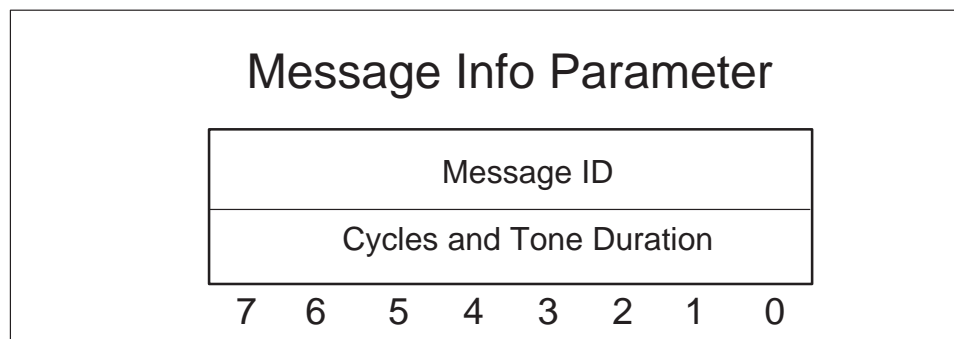
0  0 0 0 0 0: NILIDX
   0 0 0 0 1 – 0 0 1 0 1: Unused
6  0 0 1 1 0: UCS2UCS
7  0 0 1 1 1: UCS2DEX8
8  0 1 0 0 0: UCS2ACD
9  0 1 0 0 1: UCS2EAE0
   0 1 0 1 0 – 0 1 1 0 1: Unused
14 0 1 1 1 0: UCS2USP
15 0 1 1 1 1: USP2MCI
   1 0 0 0 0: Unused
17 1 0 0 0 1: UCSGITU
   1 0 0 1 0 – 1 1 1 1 1: Unused

```

Message info

This parameter contains the Message ID and the Cycles and Tone Duration information.

The Message ID is used to index into the new table PSNMSGIX to get either an index into the table ANNS (if the message ID corresponds to an announcement) or into the table TONES (if the message ID corresponds to a tone).



Optional parameter ID: 15

Parameter length: 2 bytes

Parameter contents:

- The MESSAGE ID field consists of one byte with values of 1 to 255. The values are an index into the table PSNMSGIX. For the Message ID field, a value of 0 is invalid.
- The CYCLES field consists of one byte with a second byte of the parameter contents treated as the number of cycles in which to play the announcement, if the message ID field corresponds to an announcement. The values include:

0 0 0 0 0 0 0 0: Play the Announcement indefinitely

0 0 0 0 0 0 0 1: Play the Announcement for 1 cycle

0 0 0 1 1 1 1 0: Play the Announcement for 30 cycles

0 0 0 1 1 1 1 1 – 1 1 1 1 1 1 1 1: Play the Announcement indefinitely

- The TONE DURATION field consists of one byte with a second byte of the parameter contents treated as the tone duration value, if the message ID corresponds to a tone. The values include:

0 0 0 0 0 0 0 0: Play the Tone forever

0 0 0 0 0 0 0 1: Invalid

0 0 0 0 0 0 1 0: Invalid

0 0 0 0 0 0 1 1: Play the Tone for 3 seconds

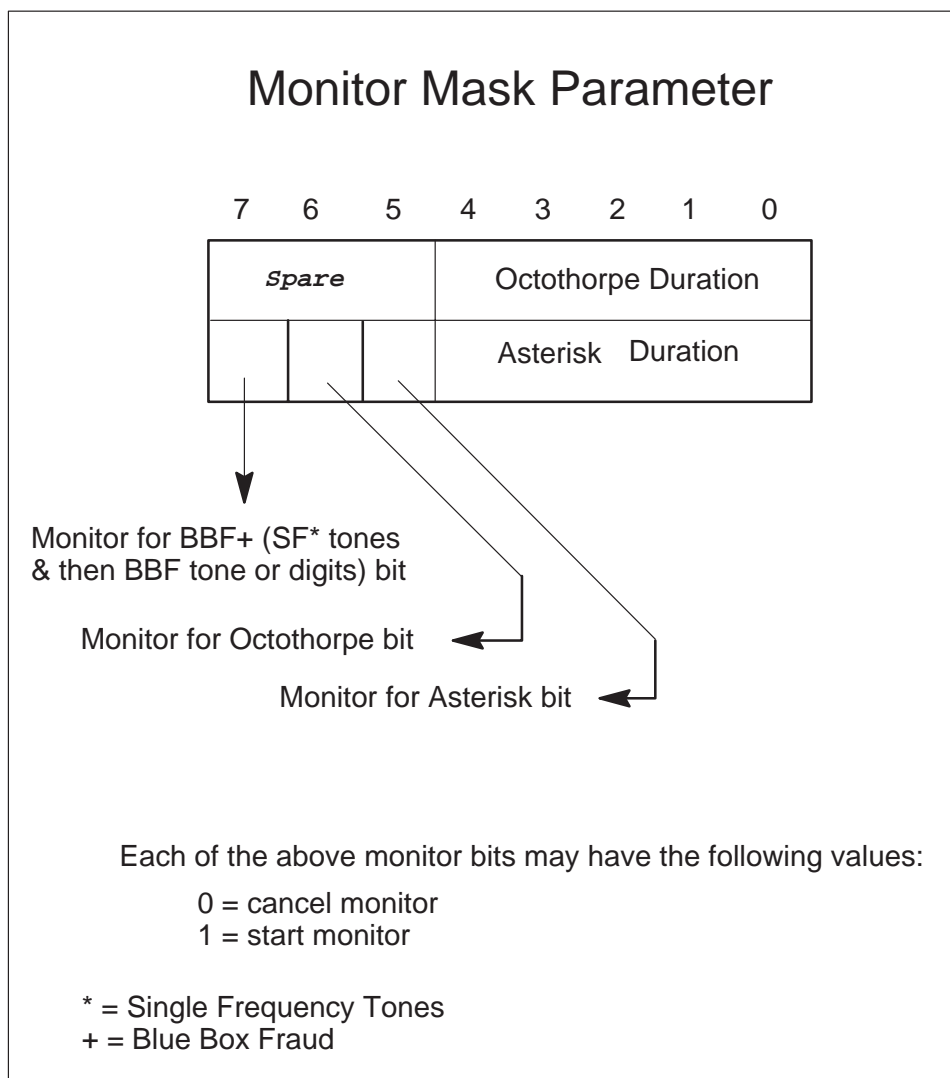
0 0 0 0 0 1 0 0: Play the Tone for 4 seconds

1 1 1 1 1 1 1 1: Play the Tone for 255 seconds

Note: The values for the cycles or tone duration may be set to any value when the *MessageInfo* parameter is sent in the *Stop_Message* primitive. The *Stop_Message* primitive is only concerned with the Message ID (to stop the appropriate message on the PSN) in the *MessageInfo* parameter.

Monitor mask

This parameter is a bitmap of monitor values. Each bit in this bitmap indicates what digit or tone to monitor, or not to monitor, for a port. Also, each bit has two values: 0, which indicates to cancel the monitor, or 1, which indicates to start the monitor.



Optional parameter ID: 16

Parameter length: 2 bytes

Parameter contents:

- The **ASTERISK** and **OCTOTHORPE DURATION** fields consist of five bits each with the duration in 100 milliseconds for which the asterisk or octothorpe must be detected before it is reported as a valid tone or digit to the SCU. The values range from 5 to 30.

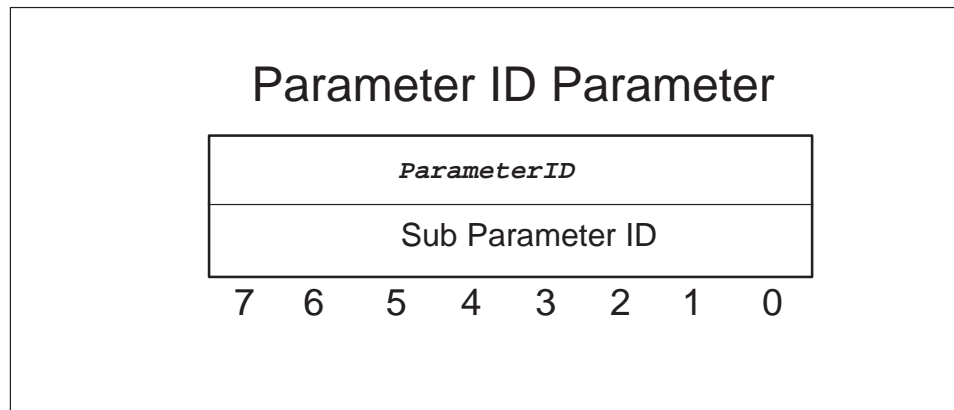
For example a duration of 5 for an Asterisk implies that the user has to hold down the asterisk for 500 msec before it is detected as a valid asterisk by the PSN and reported to the SCU.

Note: For **BBF**: the Single Frequency Tone duration, the signaling type (MF/DTMF), the Minimum Digits to Collect before a Blue Box Fraud is declared, and the Partial Dial Timer values are all determined from the table **BBTKSGRP**. Therefore, it is necessary to datafill the “terminating” port (where “terminating” indicates Party B of a **Connect** or a **Reconnect** primitive) in the table **BBTKSGRP**.

- The **TONE BITMAP** field consists of three bits and is a boolean for **BBF**, **Octothorpe**, and **Asterisk**. If the bool is true, then the port is monitored for the appropriate tone or digit. If the bool is false, then the appropriate tone or digit being monitored on that port is cancelled.

Parameter ID

This parameter contains the Identifier of the parameter, including the Identifier of the sub-parameter (in the event that the parameter is composed of sub parameters). The Identifier is returned to the SCU in case the decoding of the parameter resulted in an error.



Optional parameter ID: 17

Parameter length: 2 bytes

Parameter contents:

- The PARAMETER ID field consists of one byte, which is used to represent the parameter in error. The values include:
 - 0 00000000: Nil Error Cause
 - 1 00000001: Bearer Capability
 - 2 00000010: Billing Info
 - 3 00000011: Call Reference Identifier
 - 4 00000100: Control Info
 - 5 00000101: Destination Trunk Group
 - 6 00000110: Digit Collection
 - 7 00000111: Digits Collected
 - 8 00001000: Digits Outputted
 - 9 00001001: Digits To Output
 - 10 00001010: Error Cause
 - 11 00001011: Flow Control Information
 - 12 00001100: Flow Control Encountered
 - 13 00001101: Instruction ID
 - 14 00001110: Instruction Tag
 - 15 00001111: Message Info
 - 16 00010000: Monitor Mask
 - 17 00010001: Parameter ID
 - 18 00010010: Port Count
 - 19 00010011: Port Info
 - 20 00010100: Port Service Information
 - 21 00010101: Port Status
 - 22 00010110: Reset Reason
 - 23 00010111: Session ID
 - 24 00011000: SigInfo Mask
 - 25 00011001: Signaling Info
 - 26 00011010: Switch ID
 - 27 00011011: Time of Day
 - 28 00011100: Tone Detected
 - 00011101 – 10000001: Spare Values
 - 0 10000010: UCS Point In Call
 - 1 10000011: UCS STS
 - 10000100 – 11111111: Spare Values
- The SUB PARAMETER ID field consists of one byte, which is used to represent the parameter in error.

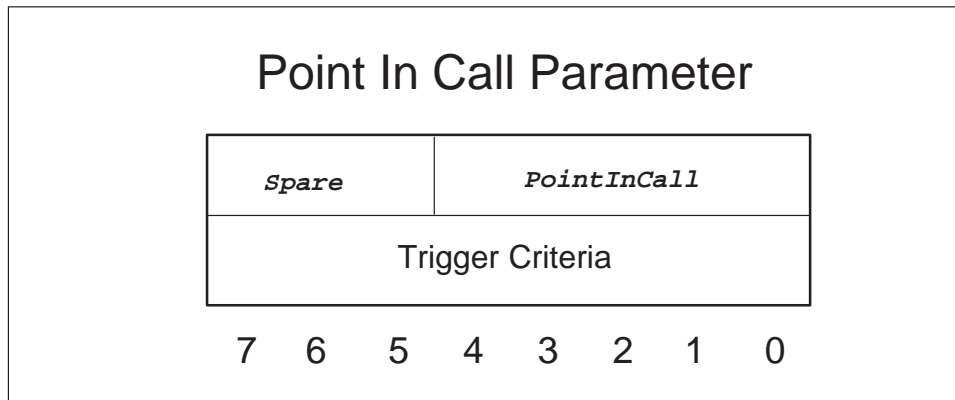
The values for PTS agents include:

- 0 00000000: Unknown
- 1 00000001: Message Type
- 2 00000010: Digits Outputted
- 3 00000011: Digits to Output

- 4 00000100: PTS Off-hook
- 5 00000101: PTS On-hook
- 00000110 – 11111111: Spare

Point in call

This parameter contains the Point in Call when all criteria are met and the PSN gives the SCU control over the call.



Optional parameter: 130

Parameter length: 2 byte

Parameter contents:

- The POINT IN CALL field consists of five bits and contains the Point in Call where the PSN determines that the call is a service call and should be controlled by the SCU. The Point in Call is encoded as follows:
 - 0 00000: Not Used
 - 1 00001: Orig Null
 - 2 00010: Authorize Orig Attempt
 - 3 00011: Collect Information
 - 4 00100: Analyze Information*
 - 5 00101: Select Route
 - 6 00110: Authorize Call Setup
 - 7 00111: Send Call
 - 8 01000: Orig Alerting
 - 9 01001: Orig Active
 - 10 01010: Orig Suspended
 - 11 01011: Term Null
 - 12 01100: Authorize Termination
 - 13 01101: Select Facility

14 0 1 1 1 0: Present Call
 15 0 1 1 1 1: Term Alerting
 16 1 0 0 0 0: Term Active
 17 1 0 0 0 1: Term Suspended
 1 0 0 1 0 – 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

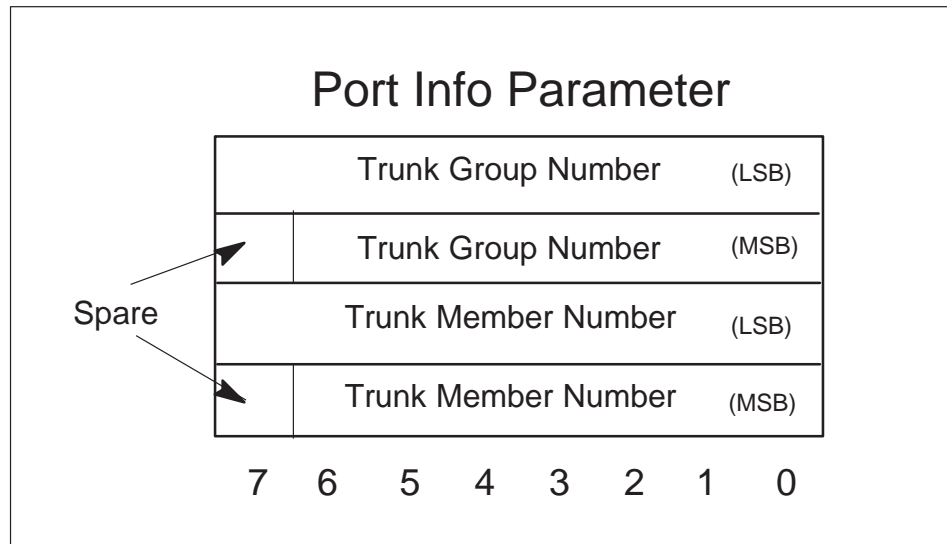
- The TRIGGER CRITERIA field consists of eight bits and is encoded as follows:

0 0 0 0 0 0 0 0: Feature Activator
 1 0 0 0 0 0 0 1: Vertical Service Code
 2 0 0 0 0 0 1 0: Customized Access
 3 0 0 0 0 0 1 1: Customized Intercom*
 4 0 0 0 0 1 0 0: NPA
 5 0 0 0 0 1 0 1: NPA_NXX
 6 0 0 0 0 1 1 0: NXX
 7 0 0 0 0 1 1 1: NXX_XXXX
 8 0 0 0 0 1 0 0: NPA_NXXXXXX*
 9 0 0 0 0 1 0 1: Country_Code_NPA_NXXXXXX*
 10 0 0 0 1 0 0 0 0: Off-hook Immed
 11 0 0 0 1 1 0 0 0: Net Busy
 12 0 0 0 1 1 0 1 1: Orig Called Party Busy
 13 0 0 0 1 1 1 0 1: Orig No Answer
 14 0 0 1 0 0 0 0 0: Orig Feature Activator
 15 0 1 1 0 0 0 0 0: Channel Setup PRI CLID
 16 0 1 1 0 0 0 0 1: Channel Setup PRI Addr
 17 0 1 1 0 0 0 1 0: Channel Setup PRI N00
 18 0 1 1 0 0 0 1 1: Channel Setup PRI Intl
 19 0 1 1 0 0 1 0 0: Specific Digit String Info*
 20 0 1 1 0 0 1 0 1: Specific Digit String ANI*
 21 0 1 1 0 0 1 1 0: Specific Digit String N00*
 22 0 1 1 0 0 1 1 1: Specific Digit String CIC
 23 0 1 1 0 1 0 0 0: Shared Interoffice CIC
 24 0 1 1 0 1 0 0 1: Shared Interoffice Info
 25 0 1 1 0 1 0 1 0: Shared Interoffice ANI
 26 0 1 1 0 1 0 1 1: Shared Interoffice Addr
 27 0 1 1 0 1 1 0 0: Shared Interoffice N00
 28 0 1 1 0 1 1 0 1: Shared Interoffice Intl
 0 1 1 0 1 1 1 0 – 1 1 1 1 1 1 1 1: Unused (Spare)

The values marked with an “*” are the only ones that are currently supported.

Port info

This parameter contains the port information for an agent. The Port Info consists of the external Trunk Group Number and the Trunk Member Number.



Optional parameter ID: 19

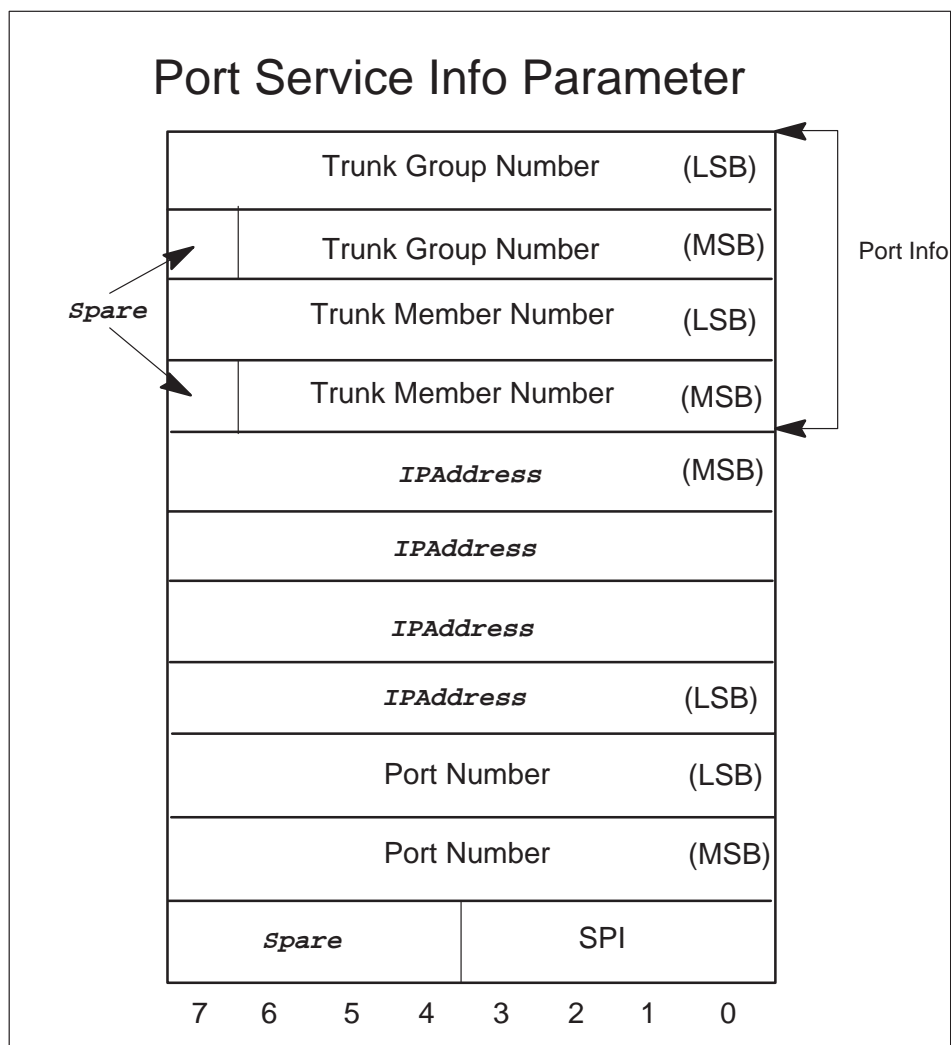
Parameter length: 4 Bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the external Trunk Group Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_GROUP.
- The TRUNK MEMBER NUMBER field consists of 15 bits and contains the Trunk Member Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_MEMBER.

Port service info

This parameter contains the IP Address and the Service Programming Interface (SPI) information. The SCU returns the IP Address and the SPI to the PSN. Please note that the IP Address is the return address used to send the event notification messages back to the SCU (to the correct address and port). In addition, the SPI number is the version for the specified agent.



Optional parameter ID: 20

Parameter length: 11 bytes

Parameter contents:

- The PORT INFO field consists of four bytes and is the location responsible for updating the IP Address and the SPI. The external Trunk Group Number and the Trunk Member Number are specified in the Port Info. The range is from 0 to 9999 for both.

A value of 32,767 for Trunk Group Number indicates a NIL_TRUNK_GROUP.

A value of 32,767 for Trunk Member Number indicates a NIL_TRUNK_MEMBER.

If the Trunk Group Number is set to NIL_TRUNK_GROUP and the Trunk Member Number is set to NIL_TRUNK_MEMBER, then the returned IP Address and the SPI information corresponds to the SCU Arbitrator address. The SCU Arbitrator address is the initial point of contact in which to send all the **New_Call** event notification messages.

The following table illustrates how to interpret the combination of Trunk Group Number and Trunk Member Number in the Port Service.

Trunk Group Number	Trunk Member Number	What is the IP Address Info used for
nil_trunk_group	nil_trunk_member	Arbitrator's IP address if received in the Set_IP_Address primitive. INVALID if received with other primitives.
Non nil_trunk_group	nil_trunk_member	IP Address for destination trunk group in Connect primitive only.
nil_trunk_group	Non nil_trunk_member	INVALID
Non nil_trunk_group	Non nil_trunk_member	IP Address for the port specified by the Trunk Group Member.

- The IP ADDRESS field consists of four bytes and contains the IP Address sent by the SCU, which is eventually used to return the event notification messages.

For example, an IP Address of 47.122.64.153 stored as four bytes where byte 1 is 47, byte 2 is 122, byte 3 is 64, and byte 4 is 153. The `nil_ip_address` is 0.0.0.0.

- The PORT NUMBER field consists of two bytes and contains the transport layer Port Number associated with the IP Address. The `nil_port_number` is 0.

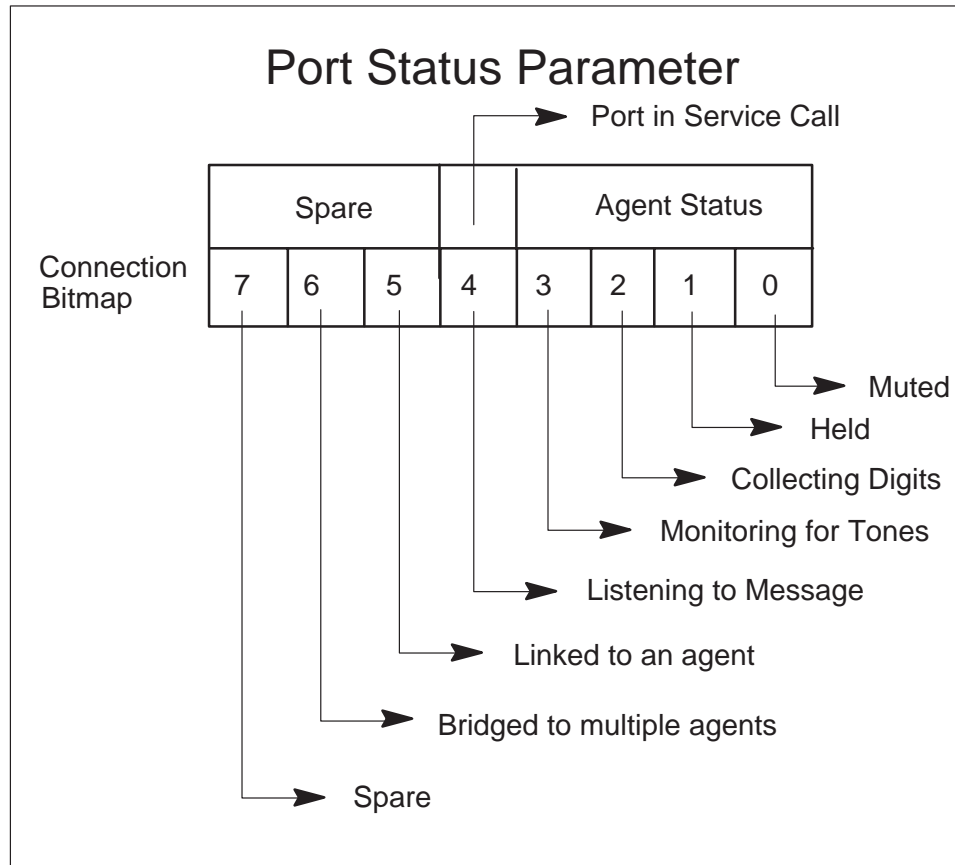
The following table illustrates how to interpret the different primitives for combination of the *IPAddress* and the Port Number in the *IPAddressInfo* parameter.

IP Address	Port Number	How is the IP Address Info used for in RESET SWITCH	How is the IP Address Info used in all the other Primitives
<code>nil_ip_address</code>	<code>nil_port_number</code>	Idle all ports that are currently serviced by the SCU (System Reset)	INVALID
NON <code>nil_ip_address</code>	<code>nil_port_number</code>	Idle all ports with the given IP Address and any port number (Shelf Reset)	INVALID
<code>nil_ip_address</code>	Non <code>nil_port_number</code>	INVALID	INVALID
Non <code>nil_ip_address</code>	Non <code>nil_port_number</code>	Idle ports with the appropriate IP Address Info (Service Reset)	<i>IPAddressInfo</i> parameter applies to a specific port – update the port's IP Address Info.

- The SPI field consists of four bits and contains the SPI version for the specified agent. The range is from 1 to 15.

Port status

This parameter contains the status of the port or agent as well as information which indicates if the port is currently controlled by the SCU.



Optional parameter ID: 21

Parameter length: 2 bytes

Parameter contents:

- The AGENT STATUS field consists of four bits and contains the Agent Status of the port, as illustrated below:

```

0  0 0 0 0: Idle
1  0 0 0 1: Seized
2  0 0 0 1 0: Answered
3  0 0 0 1 1: ManBusy
4  0 0 1 0 0: Lockout

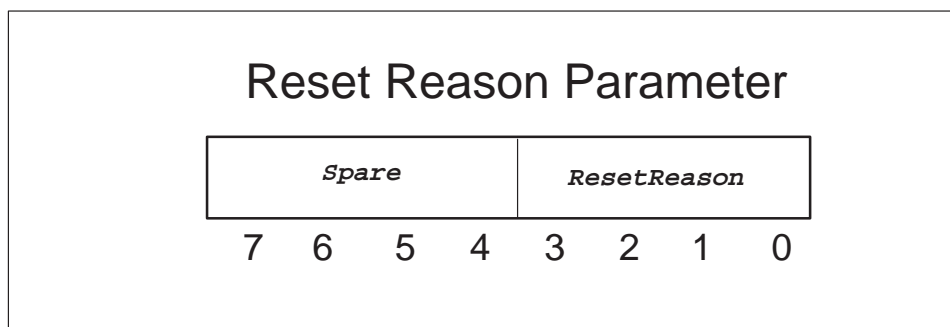
```

5 0 0 1 0 1: System Busy
 6 0 0 1 1 0: PM Busy
 7 0 0 1 1 1: Unknown
 1 0 0 0 to 1 1 1 1: Spare Values

- The PORT IN SERVICE CALL field consists of one bit. The purpose of this field is to indicate if the port is currently a part of a service call or if the port is currently being serviced by the SCU.
 - 0: Port not a part of a service call
 - 1: Port is a part of a service call.
- The CONNECTION BITMAP field consists of six bits and is a bitmap, where each bit indicates the connection status of the agent. The bits are used to represent the connection states as specified below:
 - Bit 0: Muted
 - Values: 0 = port is unmuted
 - 1 = port is muted
 - Bit 1: Held
 - Values: 0 = port is not held
 - 1 = port is held
 - Bit 2: Collecting Digits
 - Values: 0 = port is not collecting digits
 - 1 = port is in the process of collecting digits
 - Bit 3: Monitoring for Tones
 - Values: 0 = port is not monitoring for tones
 - 1 = port is monitoring for tones
 - Bit 4: Listening to Message
 - Values: 0 = port is not listening to a message
 - 1 = port is listening to message (announcement and tones)
 - Bit 5: Linked to an Agent
 - Values: 0 = port is not linked to another agent
 - 1 = port is linked to another agent
 - Bit 6: Bridged to multiple agents
 - Values: 0 = port is not bridged to two or more agents
 - 1 = port is bridged to two or more agents

Reset reason

This parameter contains the reason why a restart occurred on the PSN. In addition, the Reset Reason indicates (to the SCU) the type of reset that is required at the SCU.



Optional parameter ID: 22

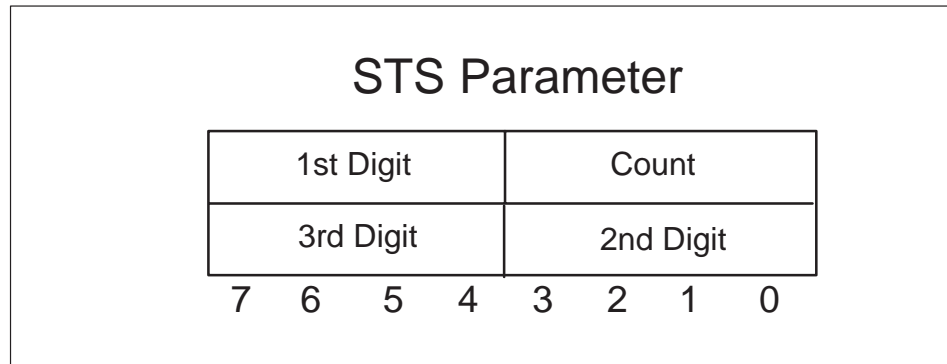
Parameter length: 1 byte

Parameter contents:

- The RESET REASON field consists of four bits and indicates the reason a restart occurred on the PSN. The values include:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Warm Restart performed on the PSN
 - 2 0 0 1 0: Cold Restart performed on the PSN
 - 3 0 0 1 1: Reload Restart performed on the PSN
 - 4 0 1 0 0: PSN has just come into service
 - 5 0 1 0 1: Arbitrator Heartbeat has failed
 - 0 1 1 0 – 1 1 1 1: Spare Values

Serving translation scheme

This parameter contains the Serving Translation Scheme (STS) of the call. The switch sends this parameter to the SCU in the `New_Call` event notification message.



Optional parameter: 131

Parameter length: 2 bytes

Parameter contents:

- The COUNT field consists of four bits and contains the number of digits in the STS. The values include 1 to 3.
- The DIGITS 1, 2, and 3 field consists of four bits for each digit and contains each digit (1,2, and 3) in the TBCD format, which is encoded as follows:

```

0 0000: Filler
1 0001: Digit 1
2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: *
12 1100: #
13 1101: D
14 1110: E

```


Signaling info

This parameter contains the Signaling Information in the standard format that is applicable to the signaling type of the port. This parameter may be used to receive or send signaling information from or to the SCU.

Depending upon the primitive or event notification in which this parameter is sent or received, and the signaling type of the associated port, its contents may differ. In general, the contents contain parameters that are encoded in the standard format.

For PTS agents, there are no standards used. For SS7 agents, the parameters are encoded based on the TR444 and GR394, and for PRI agents, refer to Chapter UCS PRI, for more details.

Please refer to the primitives and event notification messages below for information on Signaling Info.

- **Connect** primitive from the SCU (receive use):
 - The parameters below are sent in Signaling Info (on a PTS agent). At least one stream of digits (a *DigitstoOutpulse* parameter) must be sent, with the capability of sending up to three streams. These parameters are encoded as described earlier in “Bearer capability” and “Digits to outpulse.”

Bearer Capability	(Optional)
Digits to Outpulse	(Mandatory)
Digits to Outpulse	(Optional)
Digits to Outpulse	(Optional)

- The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)
Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)
RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)

XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)
Originating Line Information	(Optional)
Supplementary Line Info	(Optional)
Transit Network Selection	(Optional)

- The following SETUP message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **Disconnect** primitive from the SCU (receive use):

- The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR39.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)

- The following DISC message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)

- The following REL message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- **Transmit_Siginfo** primitive from the SCU (receive use):
 - The following parameters are sent in Signaling Info (on a PTS agent). At least one stream of digits (a *DigitstoOutputse* parameter) must be sent with the capability of sending up to three streams. The signaling parameters are encoded as described earlier in “Digits To Outputse”.

Message Type	(Mandatory)
Digits to Outputse	(Mandatory)
Digits to Outputse	(Optional)
Digits to Outputse	(Optional)

- The following information is sent (on a PTS agent) for a PTS On-hook. There is no Signaling Information present when the message type is PTS On-hook.

Message Type	(Mandatory)
--------------	-------------

- The following information is sent (on a PTS agent) for a PTS Off-hook. There is no Signaling Information present when the message type is PTS Off-hook.

Message Type	(Mandatory)
--------------	-------------

The Transmit SigInfo is used to send the IAM, ANM, CPG, FAA, FAR, FRJ, PAM, SUS, REL, and RES for an SS7 port.

The Transmit SigInfo is used to send the CONNECT, CALLPROC, FACILITY, RLC, REL, and DISC messages for a PRI port.

For the definition of the above SS7 and PRI messages, please refer to **signaling_Event** bullet in this section.

- **off_Hook** event notification to the SCU (send use):
 - The following ANM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Optional)
Call Reference	(Optional)
Carrier Selection	(Optional)
Internetwork Specific ANM	(Optional)
Intranetwork Specific ANM	(Optional)
Network Specific ANM	(Optional)
Operator Information	(Optional)
US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following CONNECT message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User to User Information	(Optional)

- **On_Hook** event notification to the SCU (send use):

- The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following RSC message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
--------------	-------------

- the following DISC message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)
User to User Information	(Optional)

- the following REL message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- **New_Call** event notification to the SCU (send use):

- The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)

Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)
RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)
XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)
Originating Line Information	(Optional)
Supplementary Line Info	(Optional)
Transit Network Selection	(Optional)

- The following SETUP message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **signaling_Event** event notification to the SCU (send use):

- The following parameters are sent in Signaling Info (on a PTS agent). These parameters are encoded as described earlier in “Digits outpulsed.”

Digits Outpulsed Indicator	(Mandatory)
----------------------------	-------------

- The following ACM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Mandatory)
Call Reference	(Optional)

US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

— The following COT message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Continuity Indicators	(Mandatory)

— The following CPG message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Event Information	(Mandatory)
Backward Call Indicator	(Mandatory)
Party Information	(Optional)
Redirection Number	(Optional)
Supply end-to-end Inf. Req.	(Optional)
Supply end-to-end Inf. Resp.	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

— The following FAA message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)

— The following FAR message parameters (on an SS7 agent) are encoded as per the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)
Called Party Address	(Optional)
Calling Party Number	(Optional)
Charge Adjustment	(Optional)
Charge Number	(Optional)
Callid (GD)	(Optional)
Operator Information	(Optional)
Originating Line Info	(Optional)

— The following FRJ message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)

Cause Indicator	(Mandatory)
Call Reference	(Optional)

- The following PAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
RPM Message	(Mandatory)

- The following RES message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend and Resume Indicators	(Mandatory)

- The following SUS message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend and Resume Indicators	(Mandatory)

- The following PROGRESS message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Progress Indicator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- The following ALERT message parameters (on a PRI agent) are encoded as indicated in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User–User Information	(Optional)
Progress Indicator	(Optional)

- The following CALL PROCEEDING message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)

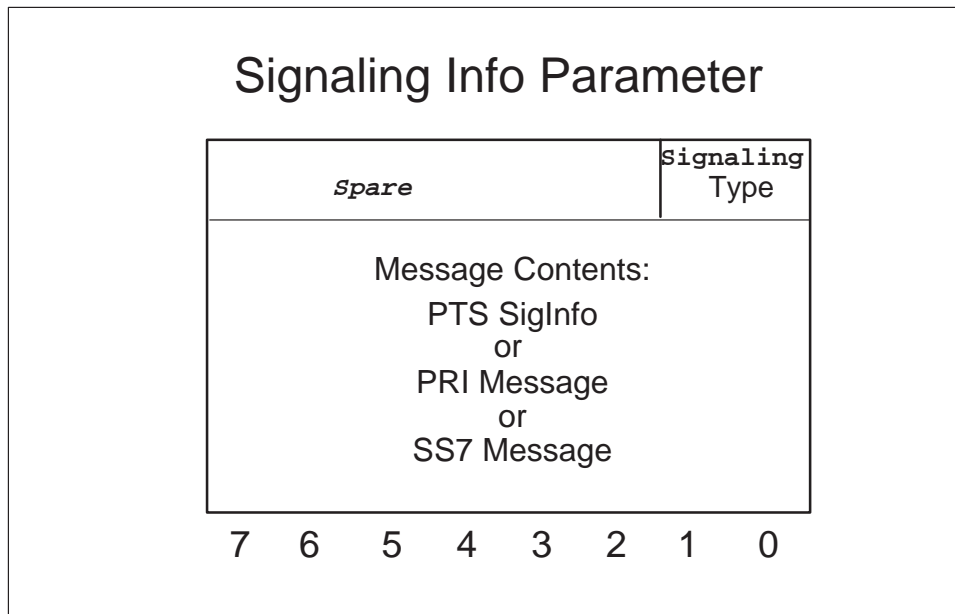
- The following FACILITY message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)

Called Party Number (Optional)
 Called Party Subaddress (Optional)

— The following REL Complete (RLC) message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type (Mandatory)
 User to User Information (Optional)



Optional parameter ID: 25

Parameter length: maximum of 412 bytes

Parameter contents:

- The Signaling TYPE field consists of two bits and is the field which indicates the Signaling TYPE.
- The MESSAGE CONTENTS field consists of a maximum of 411 bytes. In addition, this field possesses the message contents, which contain the parameters in the standard format listed earlier. The contents are encoded, depending upon the type of signaling used.

Primitive / Event Notification	PTS Messages	SS7 Messages	PRI Messages
Connect	Digits_To_Outpulse, Digits_To_Outpulse with Bearer_Capability	IAM	SETUP
Disconnect	Bearer_Capability	REL	DISC, REL
Transmit_Signaling	Digits_To_Outpulse PTS_On-hook PTS_Off-hook	ACM, IAM, ANM, REL, CPG, FAA, FAR, FRJ, PAM, RES, and SUS	CONNECT, REL, DISC, ALERT, and FACILITY
New_Call		IAM	SETUP
Off_Hook		ANM	CONNECT
On_Hook		REL and RSC	DISC and REL
Signaling_Event	Digits_Outpulsed_Indicator	ACM, CPG, COT, FAA, FAR, FRJ, PAM, RES, and SUS	PROGRESS, ALERT, CALL PROCEEDING, FACILITY, RELEASE COMPLETE
—end—			

— PTS Message Contents:

- The PTS MESSAGE TYPE field consists of one byte and is the Message Type for a PTS message. The values include:
 - 0 00000000: Unknown
 - 1 00000001: Digits to Outputpulse
 - 2 00000010: Digits to Outputpulse with Bearer Capability
 - 3 00000011: PTS Off-Hook
 - 4 00000100: PTS On-Hook
 - 5 00000101: Digits Outputpulsed
 - 00000110–11111111: Spare

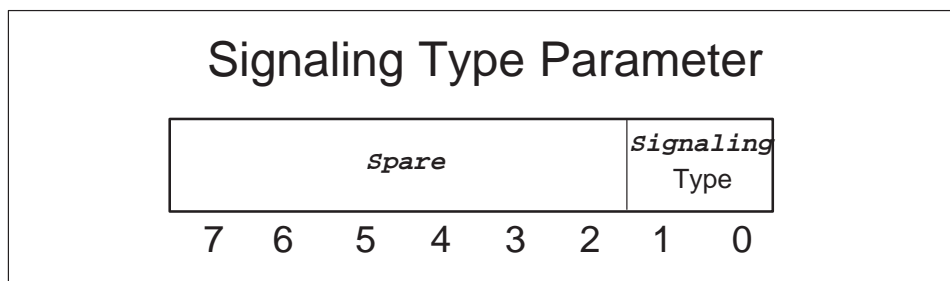
- The PTS SIGINFO MESSAGE AREA field consists of a maximum of 410 bytes and is the area that contains the SigInfo message; however, this field depends on the PTS Message Type.

The following list provides examples of the PTS Message Type contents, which appear in the PTS SigInfo Message Area field.

- Digits to Outpulse: the PTS SigInfo Message Area contains from 1 to 3 *DigitstoOutpulse* parameters (refer to “Digits to outpulse” for more information).
- Digits to Outpulse with Bearer Capability: the PTS SigInfo Message Area contains the *BearerCapability* parameter first in the area, (refer to “Bearer capability” for more information), with the following bytes containing from 1 to 3 *DigitstoOutpulse* parameters (refer to “Digits to outpulse” for more information).
- PTS Off-Hook: no information appears in the PTS SigInfo Message Area. There is no signaling information associated with a PTS Off-Hook.
- PTS On-Hook: no information appears in the PTS SigInfo Message Area. There is no signaling information associated with a PTS On-Hook.
- Digits Outpulsed: PTS SigInfo Message Area contains the *DigitsOutpulsed* parameter (refer to “Digits outpulsed” for more information).
- SS7 Message Contents:
Use of the TR444 and the GR394 SS7 with ISDN support, define the contents of the SS7 Message Contents.
- PRI Message Contents:
SS7 and PRI parameter contents are not re-defined in this document.

Signaling type

This parameter is used in the *Agent_Data* event and in the *New_Call* event.



Optional parameter ID: 32

Parameter length: 1 byte

Parameter contents:

- The Signaling TYPE field consists of two bits and represents the type of signaling the port is using. The values include:
 - 0 0: PTS (4 Wire)
 - 0 1: PTS (2 Wire)
 - 1 0: SS7
 - 1 1: PRI

SigInfo mask

This parameter allows the SCU to control which of the optional SS7 and PRI messages that are reported to the SCU in the `signaling_Event` event notification message.

For example, if the port is an SS7 port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ACM (Address Complete)
- COT (Continuity)
- CPG (Call Progress)
- FAA (Facility Activation), FAR (Facility Request), FRJ (Facility Reject)
- PAM (Pass Along Message)
- RES (Resume) and SUS (Suspend)

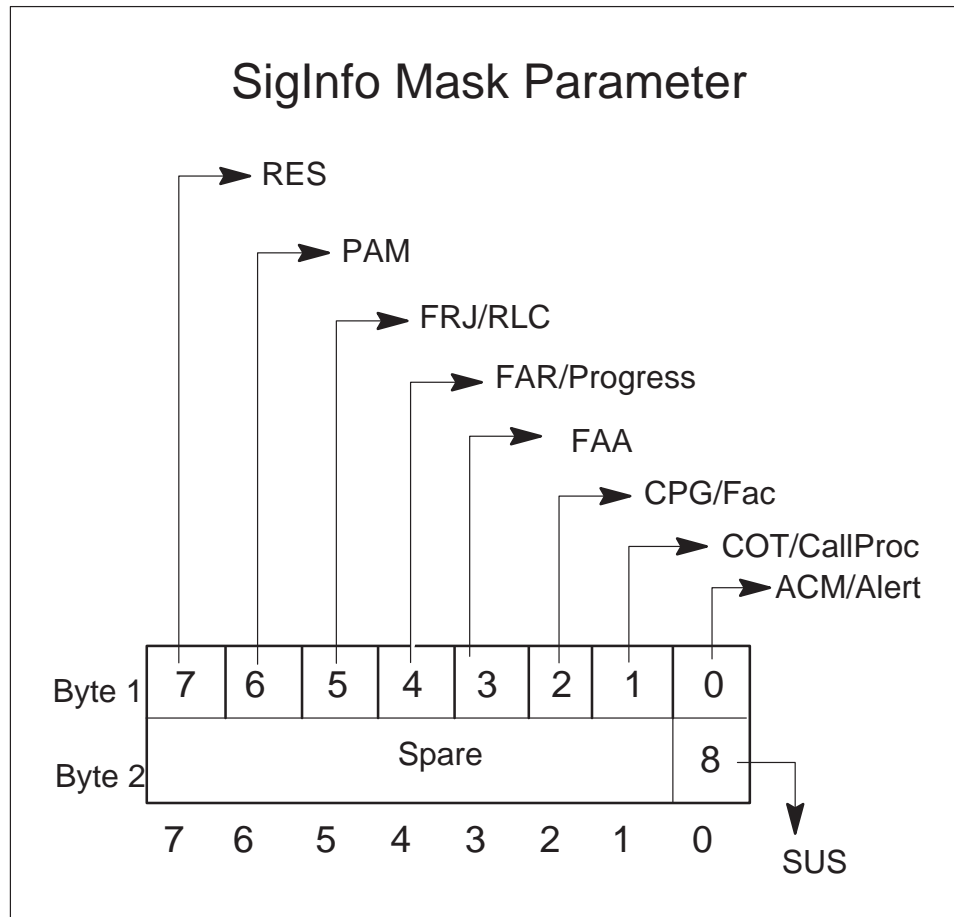
If the port is a PRI port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ALERT (Alerting Message)
- CALLPROC (Call Proceeding)
- FAC (Facility)
- RLC (Release Complete)
- PROG (Progress)

The messages that follow are mandatory. The receipt of these messages cannot be turned off by the SCU. In addition, these messages are always sent to the SCU. These messages include:

- ANM (Answer), REL (Release)/RSC (Reset Circuit) for SS7 ports

- CONNECT and REL (Release)/DISC (Disconnect) for PRI ports



Optional parameter ID: 24

Parameter length: 2 bytes

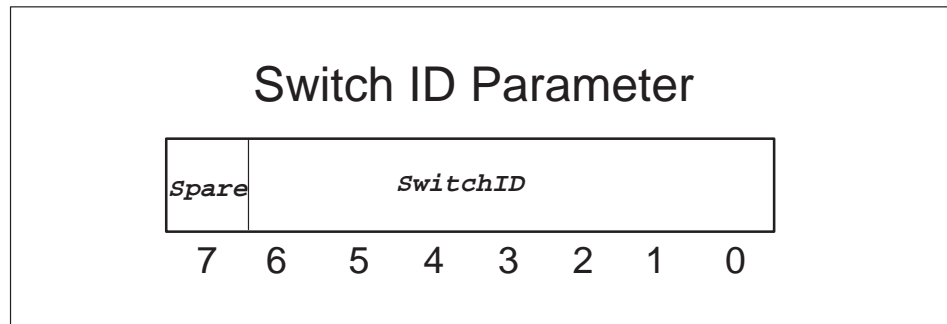
Parameter contents:

- The SIGINFO MASK BITMAP field consists of nine bits and is a bitmap, where each bit indicates the connection status of the agent. The bits are used to represent the connection states as specified below:
 - Bit 0: ACM/Alert
 - Values: 0 = do not send ACM (for SS7 port) or Alert (for PRI port) message to the SCU.

- 1 = send the ACM (for SS7 port) or Alert (for PRI port) to the SCU.
- Bit 1: COT/CallProc
 - Values: 0 = do not send COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU
 - 1 = send the COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU
- Bit 2: CPG/FAC
 - Values: 0 = do not send CPG (for SS7 port) or FAC (for PRI port) message to the SCU
 - 1 = send the CPG (for SS7 port) or FAC (for PRI port) to the SCU
- Bit 3: FAA
 - Values: 0 = do not send FAA (for SS7 port) message to the SCU
 - 1 = send the FAA (for SS7 port) to the SCU
- Bit 4: FAR/Progress
 - Values: 0 = do not send FAR (for SS7 port) or Progress (for PRI port) message to the SCU
 - 1 = send the FAR (for SS7 port) or Progress (for PRI port) to the SCU
- Bit 5: FRJ/RLC
 - Values: 0 = do not send FRJ (for SS7 port) or RLC (for PRI port) message to the SCU
 - 1 = send the FRJ (for SS7 port) or RLC (for PRI port) to the SCU
- Bit 6: PAM
 - Values: 0 = do not send PAM (for the SS7 port) to the SCU
 - 1 = send the PAM (received on the SS7 port) to the SCU
- Bit 7: RES
 - Values: 0 = do not send RES (for the SS7 port) to the SCU
 - 1 = send the RES (received on the SS7 port) to the SCU
- Bit 8: SUS
 - Values: 0 = do not send SUS (for the SS7 port) to the SCU
 - 1 = send the SUS (received on the SS7 port) to the SCU
- Bit 9 – 15: Spare

Switch ID

This parameter contains the switch ID of the PSN.



Optional parameter ID: 26

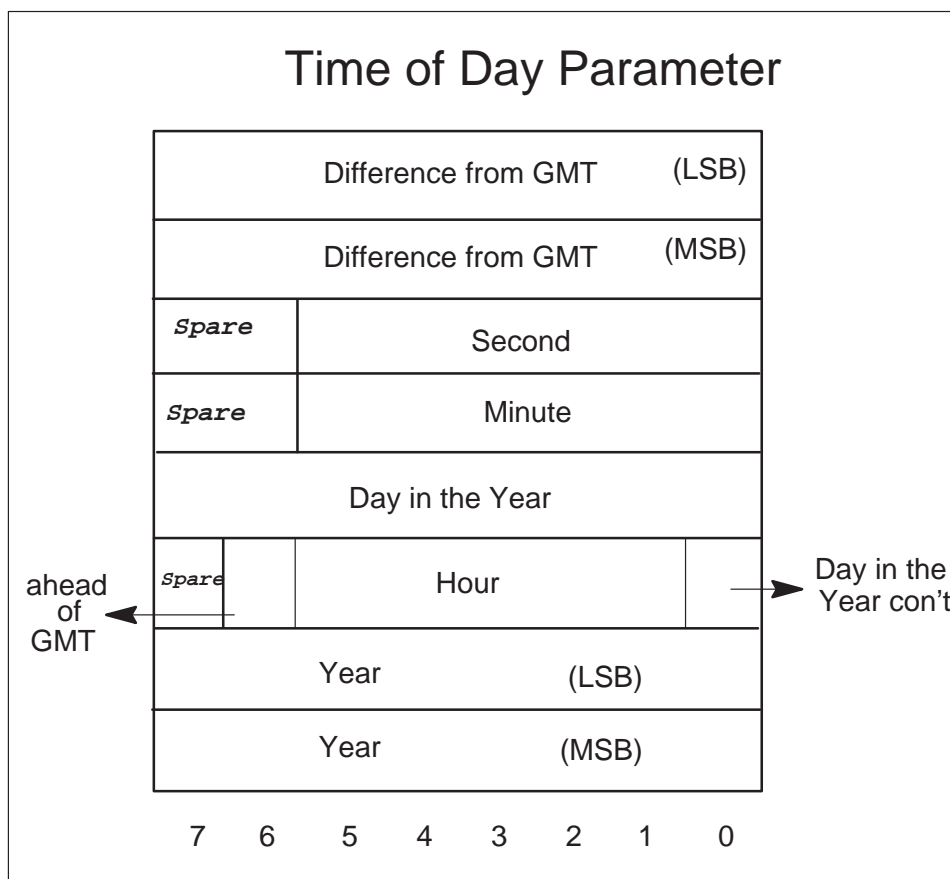
Parameter length: 1 byte

Parameter contents:

- The SWITCH ID field consists of seven bits and contains the switch ID with a range from 0 to 127.

Time of day

This parameter contains the time of the day and is returned in the `Current_Time_of_Day` event notification to the SCU.



Optional parameter ID: 27

Parameter length: 8 bytes

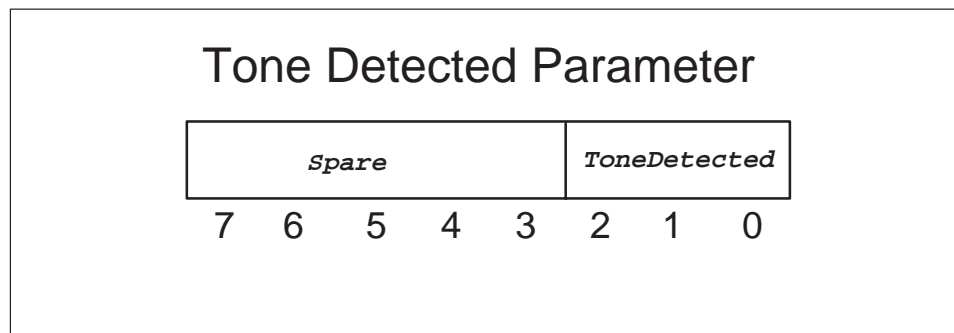
Parameter contents:

- The YEAR field consists of two bytes and contains the year.
- The DIFFERENCE FROM GMT field consists of two bytes and contains the time difference between the time specified by the “hour, minute and second” fields and the Greenwich Mean Time.
- The DAY IN THE YEAR field consists of nine bits and contains the day of the year. The values include 1 to 365.

- The MINUTE field consists of six bits and contains the minute. The values include 0 to 59 with the remaining as spare values.
- The AHEAD OF GMT field consists of one bit. When this field is true, it indicates that the switch time is AHEAD of the Greenwich Mean Time (GMT). The switch time is BEHIND the GMT if this field is set to false.
- The HOUR field consists of five bits and contains the hour. The values include 0 to 23 with the remaining as spare values.
- The SECOND field consists of six bits and contains the minute. The values include 0 to 59 with the remaining as spare values.

Tone detected

This parameter contains the tone that is detected on a given port or agent by the PSN.



Optional parameter ID: 28

Parameter length: 1 byte

Parameter contents:

- The TONE OR DIGIT DETECTED field consists of three bits and contains the tone or the digit that is detected, and is encoded as follows:
 - 0 0 0: Tone or Digit Monitoring Aborted
 - 1 0 0: Asterisk Detected
 - 2 0 1 0: Octothorpe Detected
 - 3 0 1 1: SF Tone Detected
 - 4 1 0 0: BBF Tone Detected
 - 5 1 0 1: BBF Digits Detected
 - 6 1 1 0: BBF not possible on this port
 - 1 1 1: Unused (Spare)

UCS PSN LOPER messages and parameters version 2

This chapter contains a detailed description of the UCS PSN messages and parameters Version 2 described in the Low Overhead Protocol Encoding Rule (LOPER) message format.

LOPER format

Each of the messages in the LOPER follow the same basic format and rules.

The following rules apply to LOPER:

- A message is byte aligned, meaning that each parameter starts on a new byte.
- The first byte in a message is considered to be the 0th index of the message, the second byte in the message is considered to be the 1st index of the message, and so on.
- Two bytes are used in describing the length of a message and parameter. Two bytes are also used to contain at which byte location in the message that the optional parameters begin in the message.
- A LOPER message can either contain one primitive, one event, or one macro which can contain up to five primitives.
- The SS7 signaling message of the SigInfo contents in an optional *SigInfo* parameter to be transmitted on a SS7 agent is not validated. Since the SS7 signaling message is to be built by the SCU in the TR444 BellCore standard, it is assumed that the SS7 signaling message has been correctly built and is sent onto the SS7 network without validation. Any error in the SS7 signaling message is caught by the network and handled accordingly by the SS7 network.

Messages sent from the SCU to the PSN

This section covers messages sent from the SCU to the PSN.

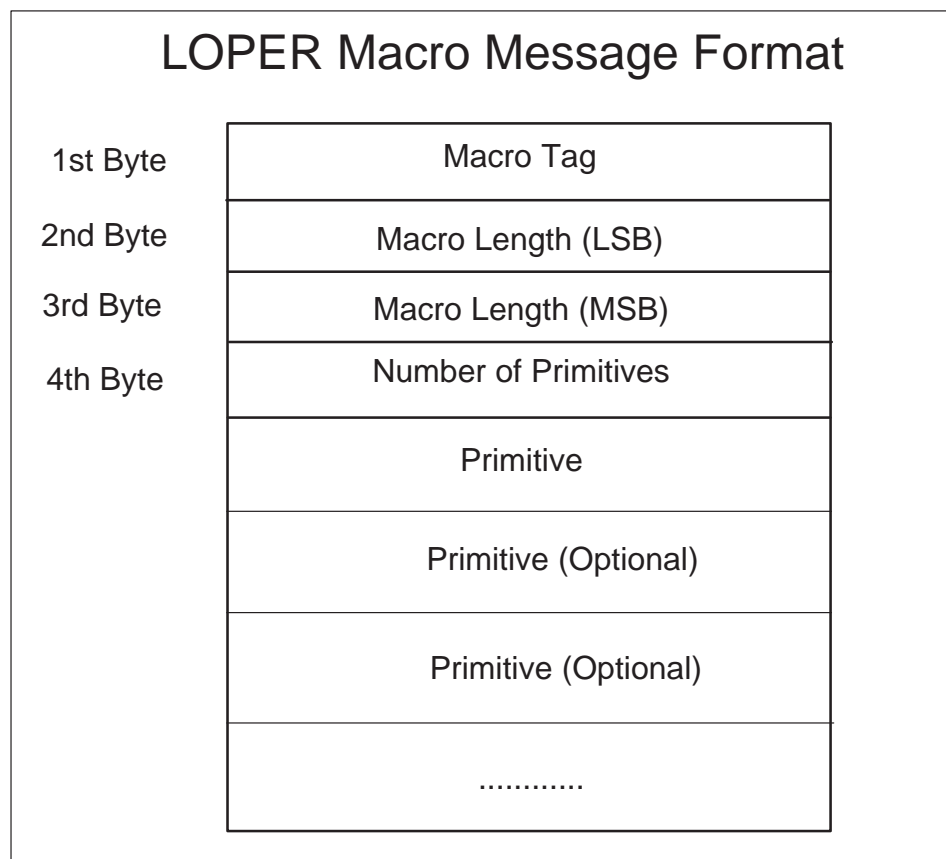
Macros

The first byte of a macro designates that the message is a macro. The second and third bytes indicate the length of the macro in the number of bytes that

comprise the length of the message from the Macro Tag to the end of the last primitive in the macro. The fourth byte identifies how many primitives there are in the primitive Area. The primitive area contains the individual primitives.

If there is an error in the decoding of one of the primitives found in the macro, that primitive is dropped and the decoding process is ceased since the rest of the macro is considered to be corrupted. Any primitive decoded before the error is still processed.

LOPER macro message format



- The MACRO TAG field consists of one byte and is the location which is used to identify the primitive in the message. The values include:
 0 0 0 0 0 0 0 0 : Macro Tag

Primitives

The first byte is the primitive tag and identifies the primitive. The second and third bytes indicate the length of the primitive beginning with the primitive tag to the end of the mandatory parameters or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters. The contents of the mandatory parameter depend on the primitive and event and the customer using LOPER. Refer to Figure 17-1.

Figure 17-1

LOPER primitive message format

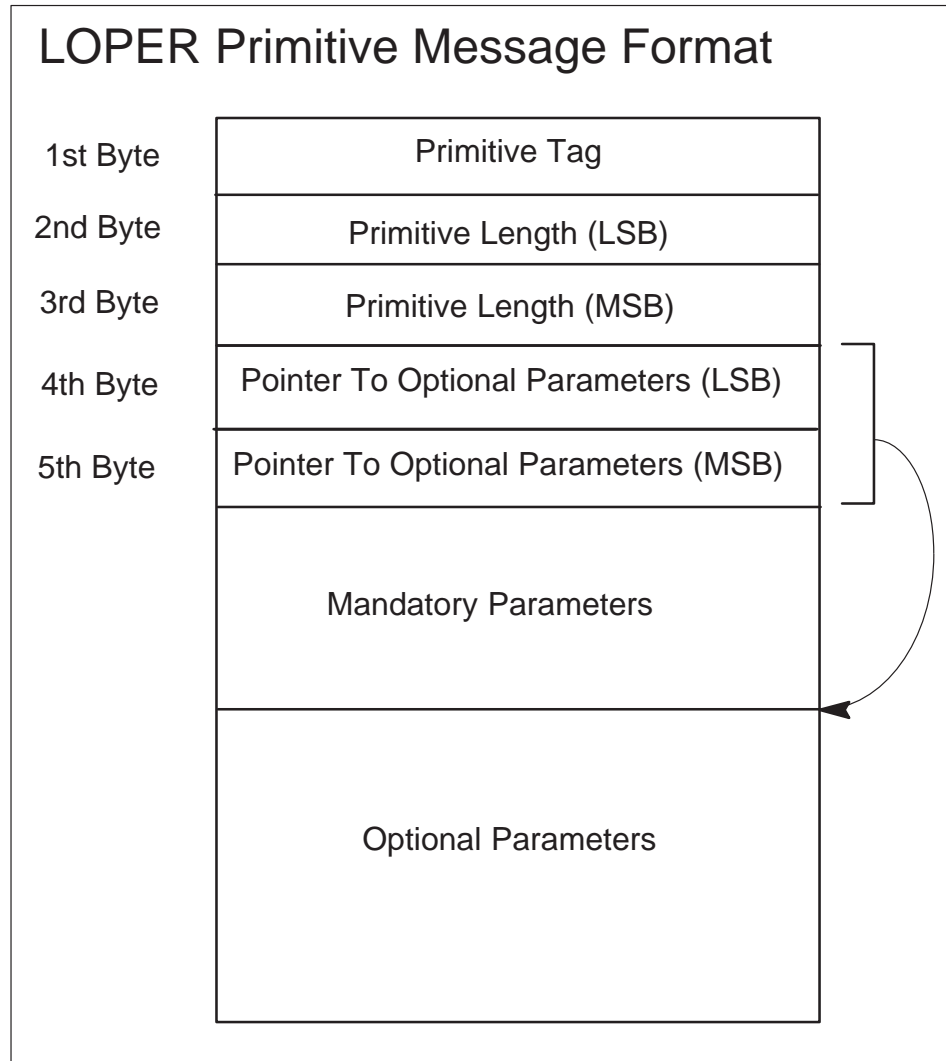
- The PRIMITIVE TAG field consists of one byte and is a byte that is used to identify the primitive in the message. The values include:

0	00000000	: Nil Primitive Tag
1	00000001	: Bridge
2	00000010	: Collect Digits
3	00000011	: Connect
4	00000100	: Disconnect
5	00000101	: Hold
6	00000110	: Monitor
7	00000111	: Mute
8	00001000	: RESERVED FOR INTERNAL PSN USE
9	00001001	: New Call Accepted
10	00001010	: New Call Rejected
11	00001011	: Play Message
12	00001100	: PPCD
13	00001101	: Query Port
14	00001110	: Reconnect
15	00001111	: Set Billing Record
16	00010000	: Stop Message
17	00010001	: Transmit Signaling Info
18	00010010	: Error Detected
19	00010011	: Heartbeat
20	00010100	: Port Status
21	00010101	: Query TOD
22	00010110	: Reset Switch
23	00010111	: Set IP Address
24	00011000	: Flow Control
	00011001 to 11111111	: Spare

Messages sent from the PSN to the SCU

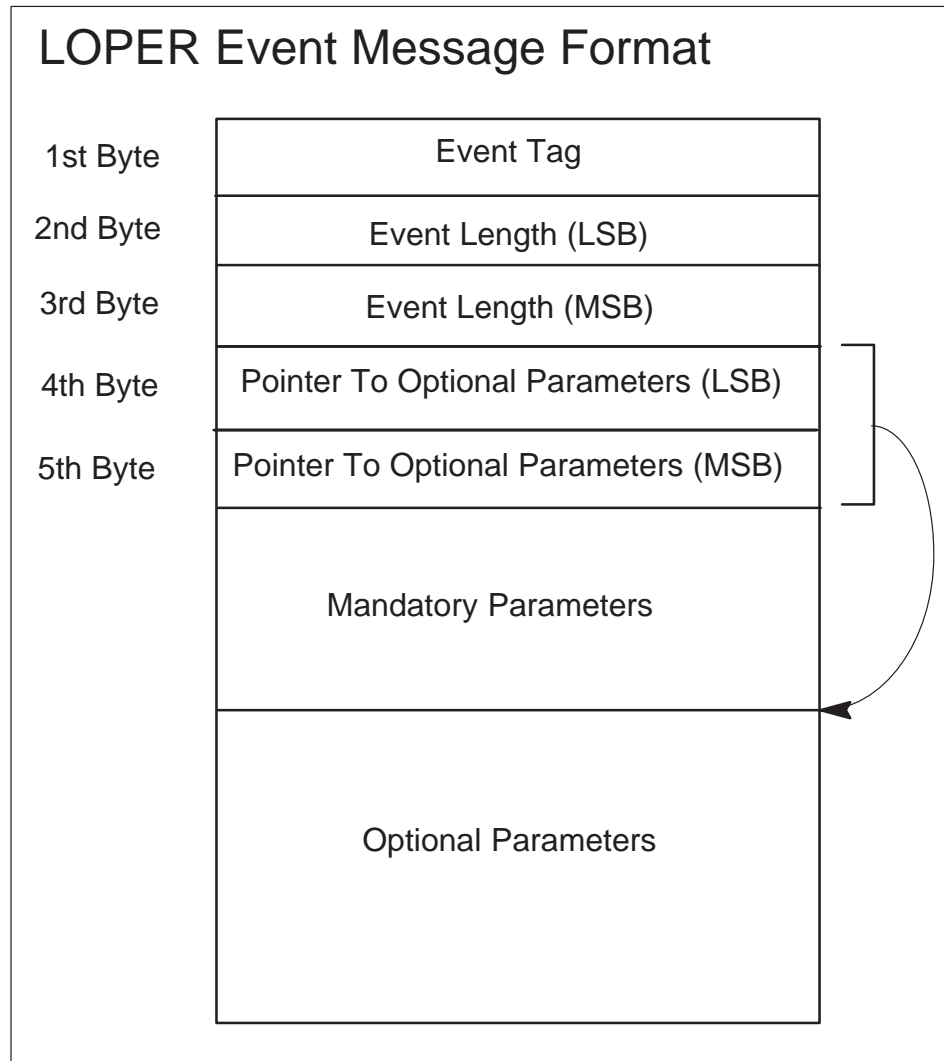
Events

The first byte is the Event Tag and identifies the event. The second and third bytes indicate the length of the event, beginning with the Event Tag to the



end of the mandatory parameters, or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters. The contents of the mandatory parameters depend on the primitive and event, and the customer using LOPER. Refer to the following figure.

LOPER event message format



- The EVENT TAG field consists of one byte and is a byte that is used to identify the event in the message. The values include:

```

0 00000000 : Current TOD
1 00000001 : Digits Collected
2 00000010 : Error Detected
3 00000011 : In Service
4 00000100 : Instruction Completed
5 00000101 : Message Played
6 00000110 : New Call
7 00000111 : Off Hook
8 00001000 : On Hook
9 00001001 : Port Status

```

10 0 0 0 0 1 0 1 0 : Query Port
11 0 0 0 0 1 0 1 1 : Route Not Available
12 0 0 0 0 1 1 0 0 : Route Selected
13 0 0 0 0 1 1 0 1 : Signaling Event
14 0 0 0 0 1 1 1 0 : Tone Detected
15 0 0 0 0 1 1 1 1 : Agent Data
16 0 0 0 1 0 0 0 0 : Stop Heartbeat
0 0 0 1 0 0 0 1 to 1 1 1 1 1 1 1 1 : Spare

Parameters

LOPER divides parameters into two classes, mandatory and optional. Mandatory parameters contain the information required by the primitive and event in order to be processed. Optional parameters contain the information not required by the primitive and event but add extra information for processing. For a mapping of which parameters are mandatory and/or optional for primitives and events, refer to the tables within Chapter “PRI messages.”

Mandatory parameters

Mandatory parameters in LOPER are customized for each primitive and event. The sixth byte in the primitive and event is always the beginning of the mandatory parameters for primitives and events. There is no need for a mandatory parameter tag. Mandatory parameters are identified by the primitive and event ID and can only be found in their corresponding primitive and event, such as a *Connect* mandatory parameter can only be found in a *Connect* primitive. Also, there is no need for a length indicator for the mandatory parameters. For example, since the pointer to the optional parameters designates the end of the mandatory parameters, and if there are no optional parameters, then the length of the primitive and event is the end of the mandatory parameters. The contents of mandatory parameters is covered in the Chapter “PSN parameters Version 1.”

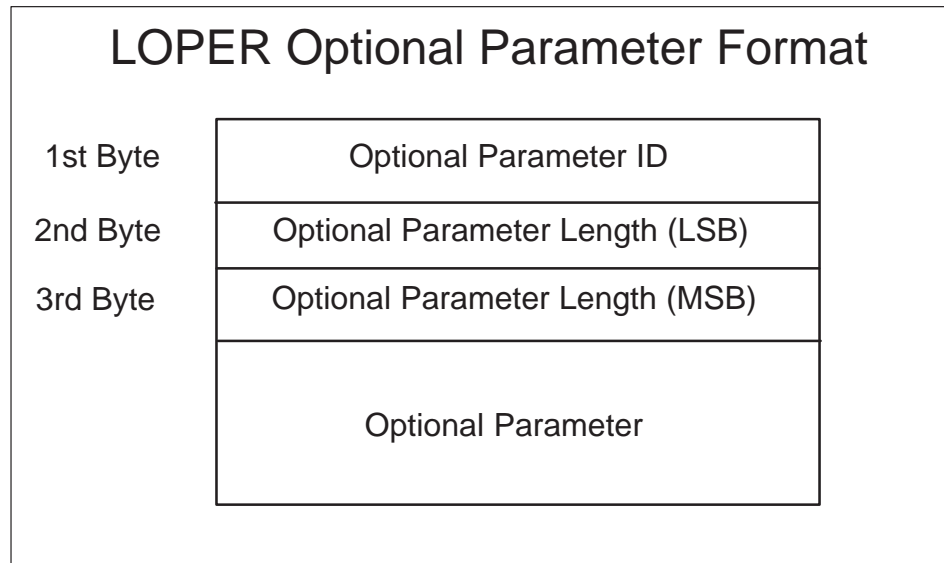
If there is an error in the decoding of the mandatory parameters, then entire message is dropped, the OM PSN_ERPS:DECODEFL is incremented, the logs PSN202 and PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause HEADER_DECODE_FAILURE_EC.

If there is an error in the validating of the mandatory parameters values, then the entire message is dropped, the OM PSN_ERPS:MANDPDEF is incremented, the logs PSN202 and PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause PARAM_CONTENTS_OUT_OF_RANGE_EC.

Optional parameters

The first byte of an optional parameter is the optional parameter tag, which identifies the optional parameter. The second and third bytes indicate the length of the optional parameter in bytes, starting with the beginning of the optional parameter ID, to the end of the optional parameter. The contents of the optional parameter depends on the specific parameter and the customer using LOPER. The contents of the optional parameters are defined in PSN PARMS. Refer to the following figure.

If a LOPER message is received with an unknown optional parameter ID, the unknown optional parameter is skipped and the decoding of the rest of the optional parameters in the message continues. If there is an error in the decoding of a known optional parameter, the optional parameter is dropped and the decoding process is ceased. As a result, the reset of the message is considered to be corrupted, and the OM PSN_ERPS:OPPRMDEF is incremented, the logs PSN202 and PSN212 are produced, and the primitive, along with the optional parameters, are decoded before the error is processed.



- The OPTIONAL parameter ID field consists of one byte and is the location which is used to represent the parameter. The values include:
 - 0 00000000 : Unknown
 - 1 00000001 : Bearer Capability
 - 2 00000010 : Billing Info
 - 3 00000011 : Call Reference Identifier
 - 4 00000100 : Control Info
 - 5 00000101 : Destination Trunk Group

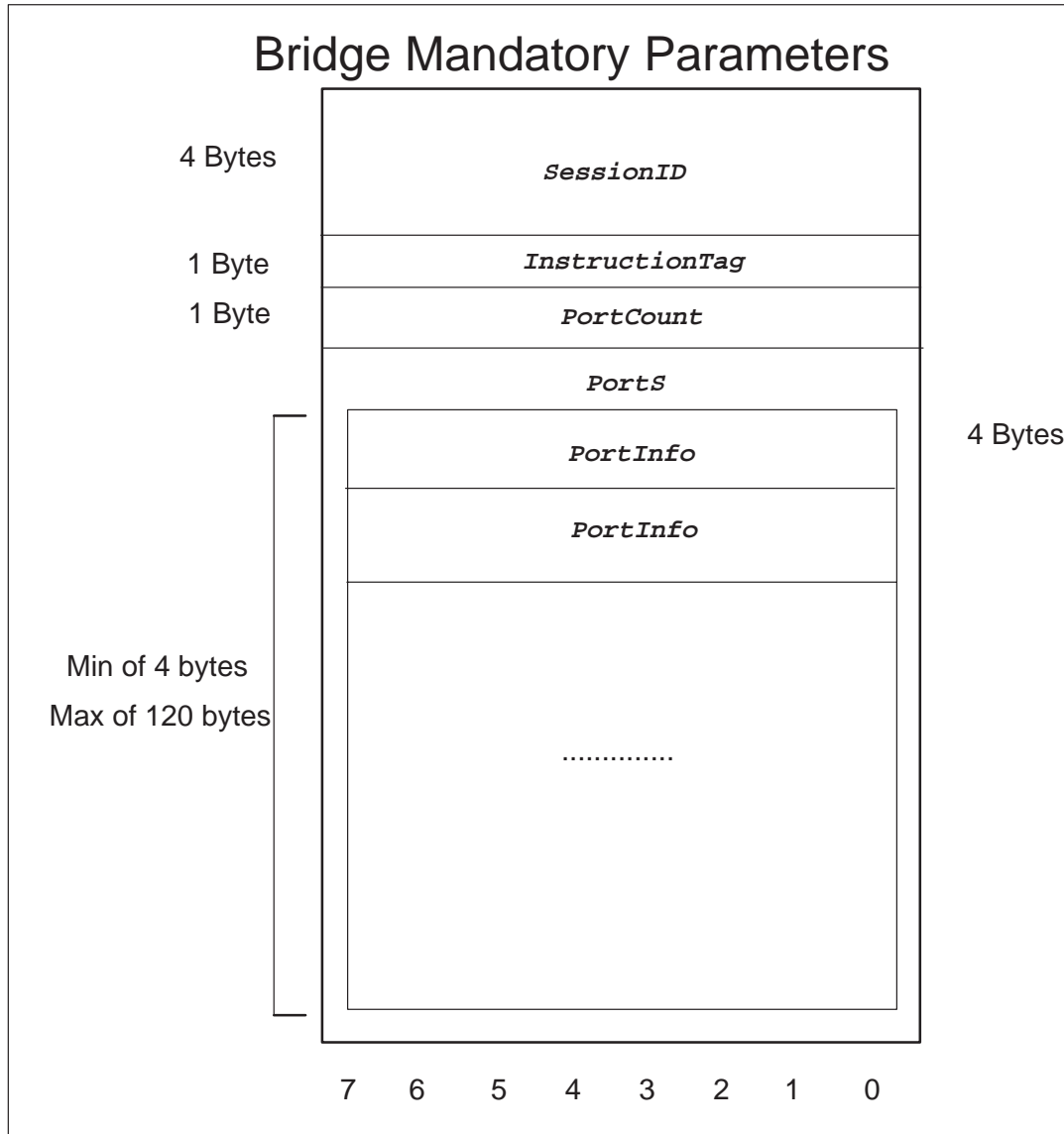
6 00000110 : Digit Collection
7 00000111 : Digits Collected
8 00001000 : Digits Outpulsed
9 00001001 : Digits To Outpulse
10 00001010 : Error Cause
11 00001011 : Flow Control Information
12 00001100 : Flow Control Encountered
13 00001101 : Instruction ID
14 00001110 : Instruction Tag
15 00001111 : Message Info
16 00010000 : Monitor Mask
17 00010001 : Parameter ID
18 00010010 : Port Count
19 00010011 : Port Info
20 00010100 : Port Service Information
21 00010101 : Port Status
22 00010110 : Reset Reason
23 00010111 : Session ID
24 00011000 : SigInfo Mask
25 00011001 : Signaling Info
26 00011010 : Switch ID
27 00011011 : Time of Day
28 00011100 : Tone Detected
29 00011101 : Agent Type
30 00011110 : COT Required
31 00011111 : ISUP Index
32 00100000 : Signaling Type
33 00100001 to 10000001 : Spare Values
130 10000010 : UCS Point In Call
131 10000011 : UCS STS
10000100 to 11111111 : Spare Values

PSN LOPER mandatory parameters for primitives

The following mandatory parameters are used to build primitives. Their description includes the length of the parameter as well as the type of each field in the parameter.

Bridge mandatory parameters

The mandatory parameters for a **Bridge** primitive are as follows:



Parameter length: minimum of 10 bytes and a maximum of 126 bytes

Parameter contents:

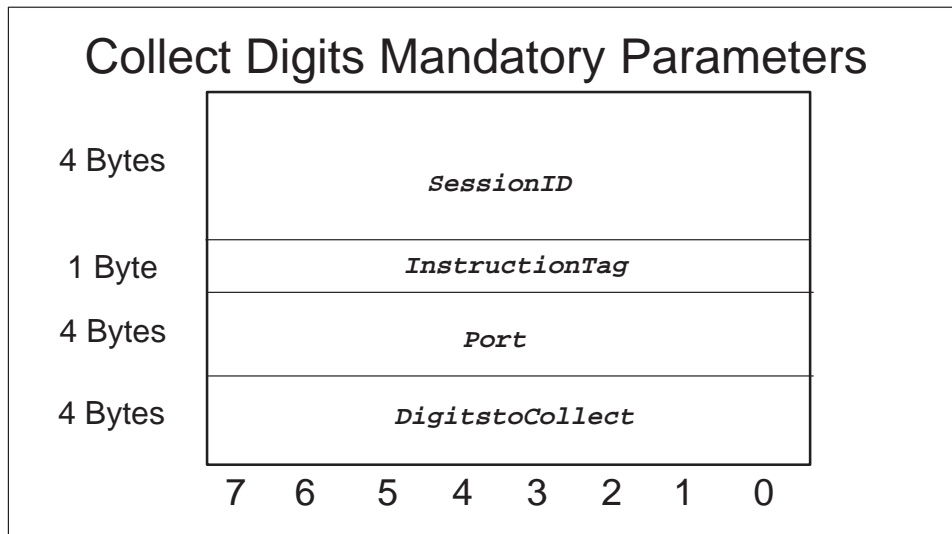
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The PORT COUNT field consists of one byte and is the location which includes the TYPE: Count of 1 to 30.
- The PORTS field consists of a minimum of four bytes and a maximum of 120 bytes. Also, the PORTS field consists of a minimum of one port to a maximum of 30 ports.

The Port TYPE: *PortInfo* parameter.

Collect digits mandatory parameters

The mandatory parameters for a `Collect_Digits` primitive.



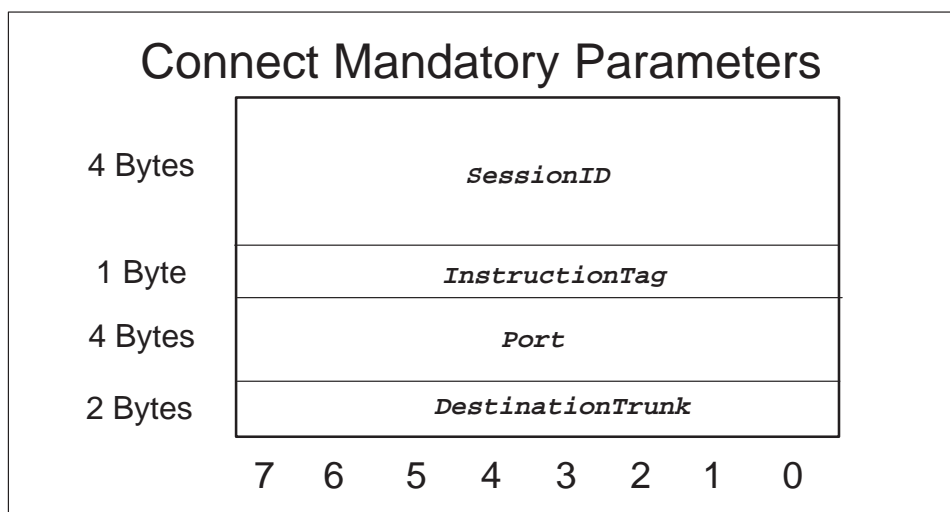
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitCollection* parameter.

Connect mandatory parameters

The mandatory parameters for a `Connect` primitive are as follows:



Parameter length: 11 bytes

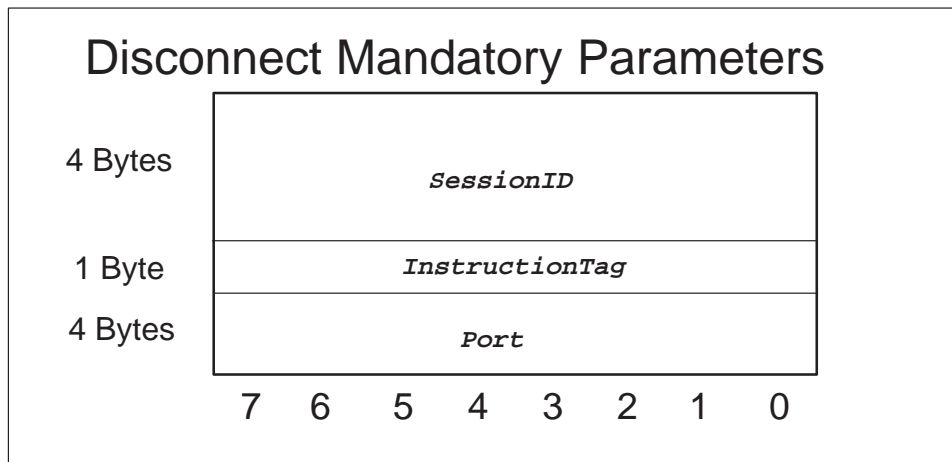
Instruction

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DESTINATION TRUNK field consists of two bytes and is the location which includes the TYPE: *Destinationtrunkgroup* parameter.

Disconnect mandatory parameters

The mandatory parameters for a **Disconnect** primitive are as follows:



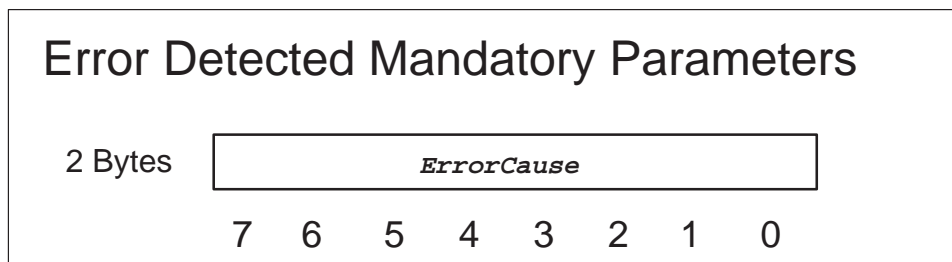
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Error detected mandatory parameters

The mandatory parameters for a **Error_Detected** primitive are as follows:



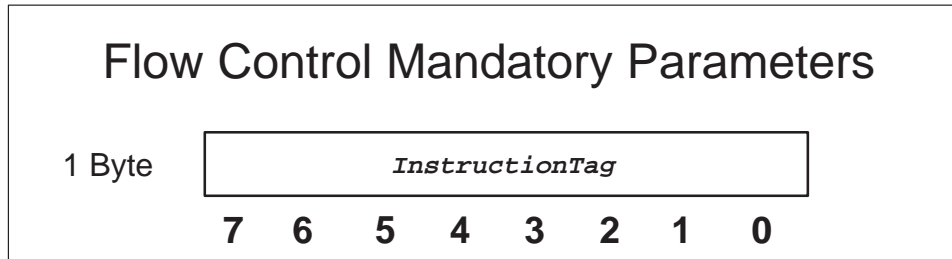
Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

Flow control mandatory parameters

The mandatory parameters for a `Flow_Control` primitive are as follows:



Parameter length: 1 byte

Parameter contents:

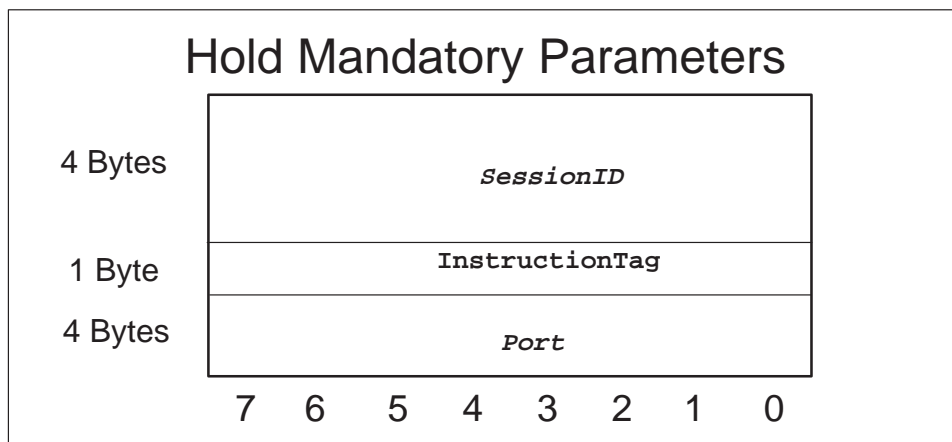
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Heartbeat mandatory parameters

The `Heartbeat` primitive does not contain any information in the primitive. A `Heartbeat` primitive message contains only what is mandatory for a LOPER primitive which is the primitive tag, the primitive length, and the optional pointer, totaling five bytes.

Hold mandatory parameters

The mandatory parameters for a `Hold` primitive are as follows:



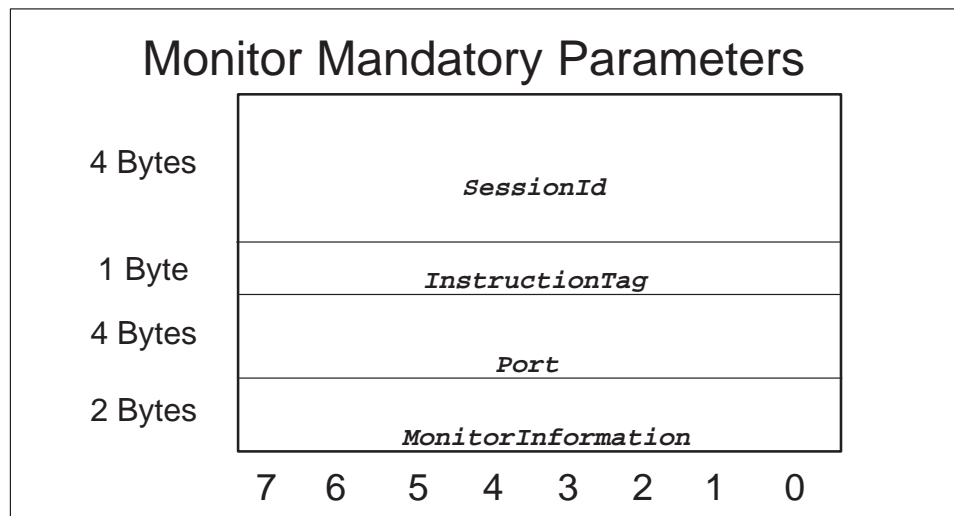
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Monitor mandatory parameters

The mandatory parameters for a **Monitor** primitive are as follows:



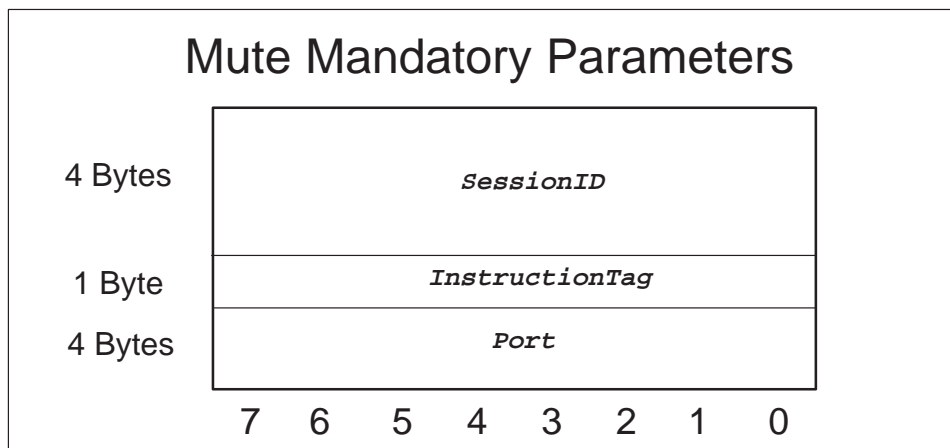
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MONITOR INFORMATION field consists of two bytes and is the location which includes the TYPE: *MonitorMask* parameter.

Mute mandatory parameters

The mandatory parameters for a **mute** primitive are as follows:



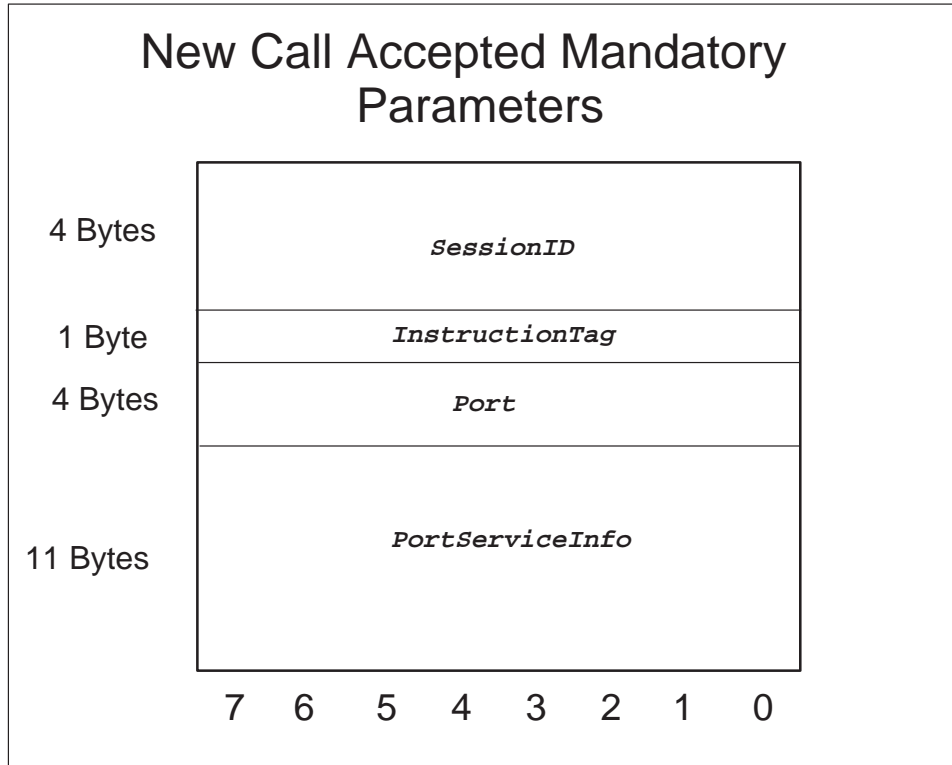
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call accepted mandatory parameters

The mandatory parameters for a `New_Call_Accepted` primitive are as follows:



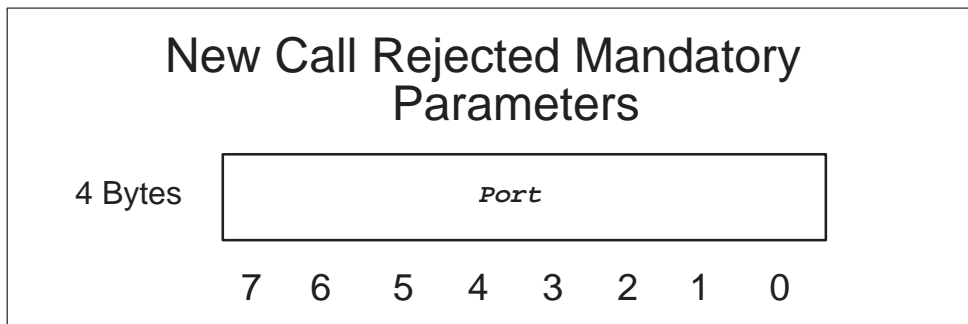
Parameter length: 20 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT SERVICE INFO field consists of 11 bytes and is the location which includes the TYPE: *PortServiceInfo* parameter.

New call rejected mandatory parameters

The mandatory parameters for a `New_Call_Rejected` primitive are as follows:



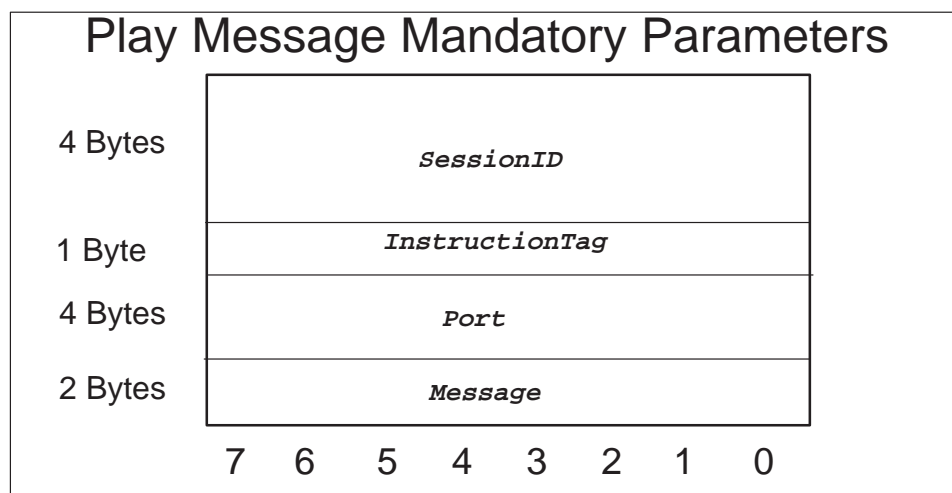
Parameter length: 4 bytes

Parameter contents:

- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Play message mandatory parameters

The mandatory parameters for a `Play_Message` primitive are as follows:



Parameter length: 11 bytes

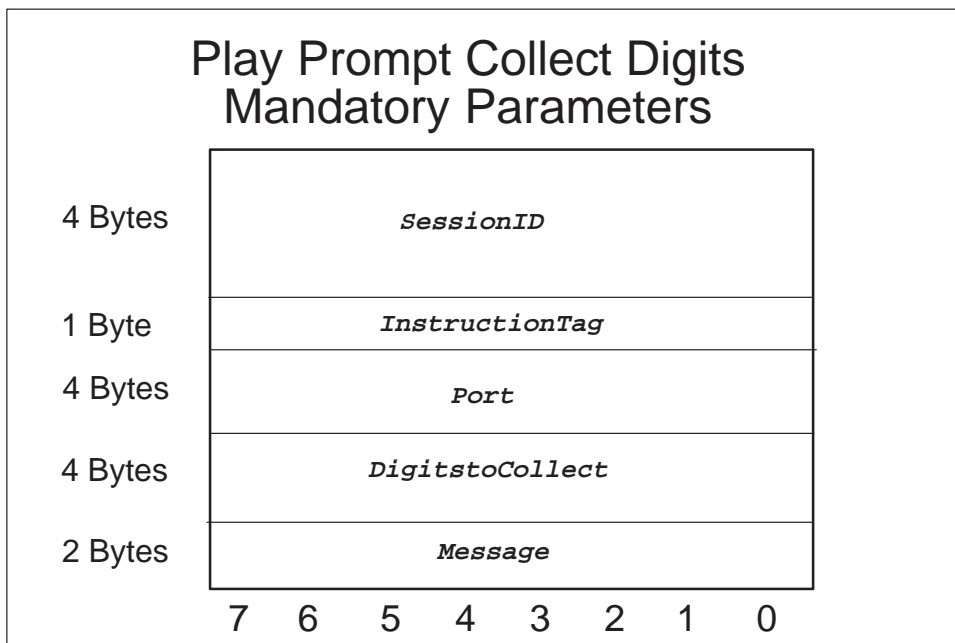
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Play prompt collect digits mandatory parameters

The mandatory parameters for a `Play_Prompt_Collect_Digits_&_Report` primitive are as follows:



Parameter length: 15 bytes

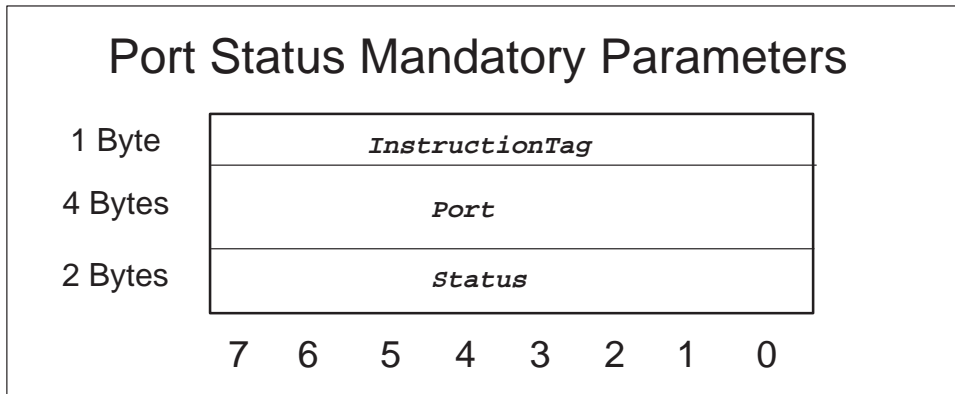
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitCollection* parameter.

- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` primitive are as follows:



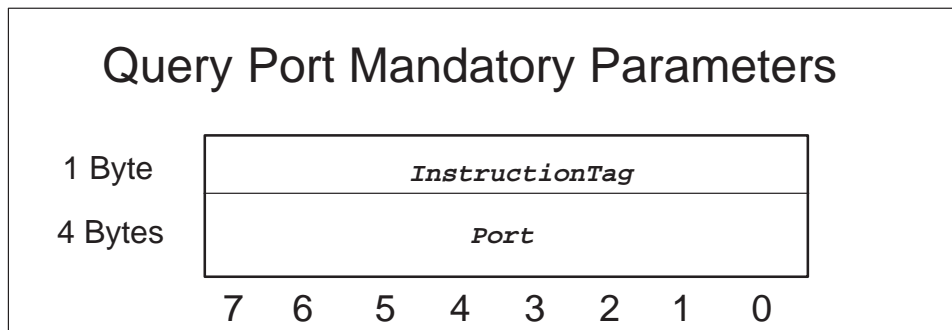
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` primitive are as follows:



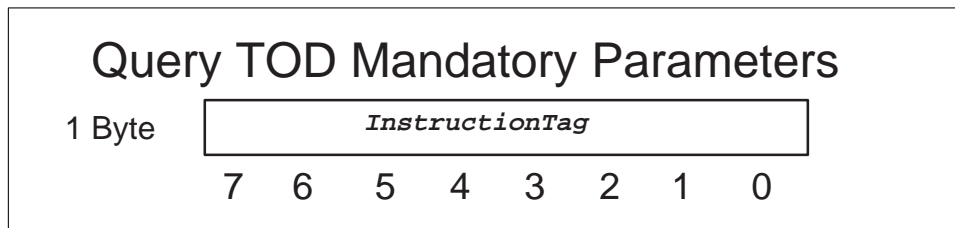
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Query time of day (TOD) mandatory parameters

The mandatory parameters for a `query_TOD` primitive are as follows:



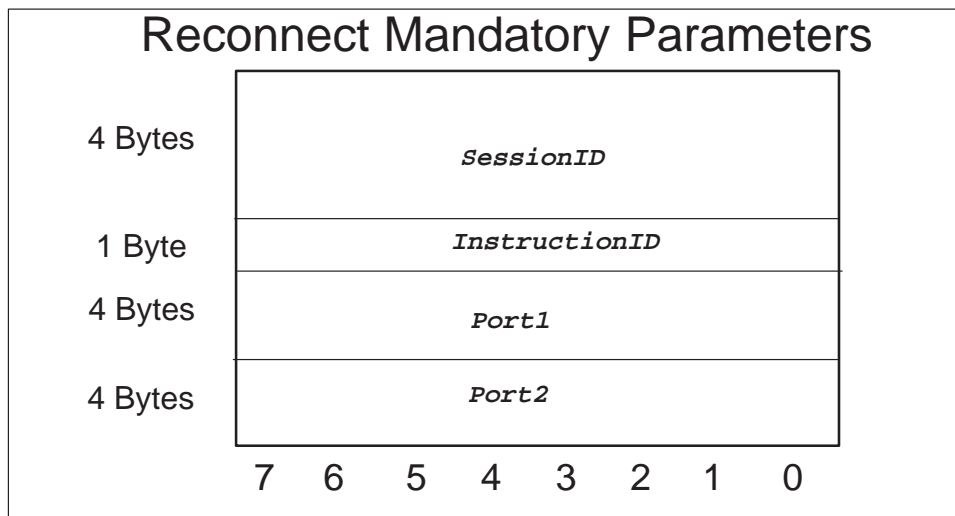
Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Reconnect mandatory parameters

The mandatory parameters for a `Reconnect` primitive are as follows:



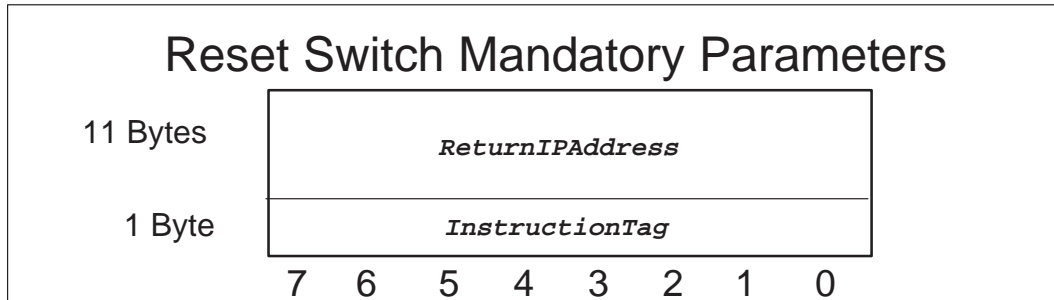
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT 1 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT 2 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Reset switch mandatory parameters

The mandatory parameters for a `Reset_Switch` primitive are as follows:



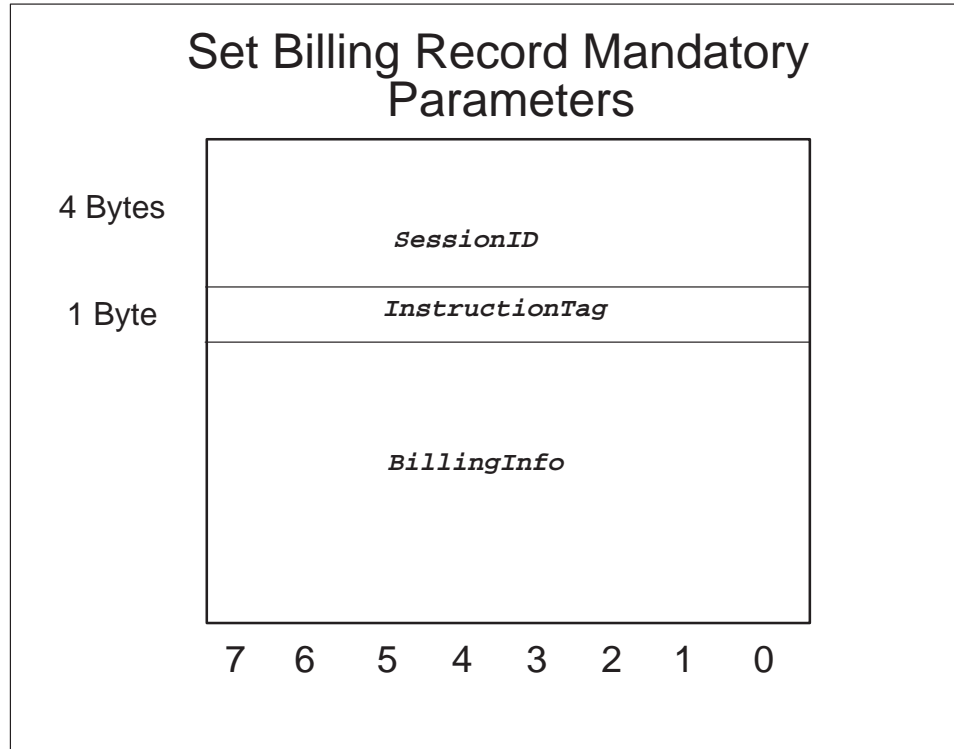
Parameter length: 12 bytes

Parameter contents:

- The RETURN IP ADDRESS field consists of 11 bytes and is the location which includes the TYPE: *PortService* parameter.
Note: There is a mandatory PSI parameter called the Return IP Address parameter, which is used to send the **Instruction_Completed**. There are 0 to 20 optional PSI parameters called the IP Address to Reset parameters, which are used to do the actual resetting of calls.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Set billing record mandatory parameters

The mandatory parameters for a `Set_Billing_Record` primitive are as follows:



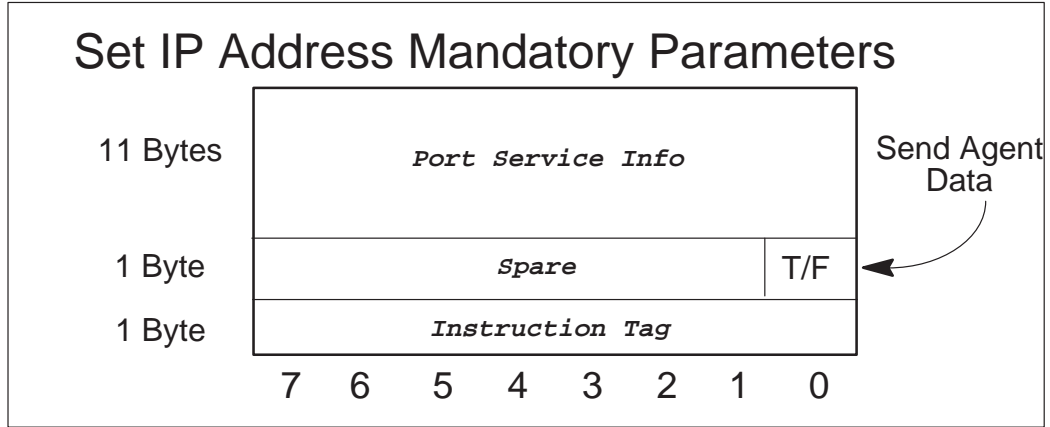
Parameter length: variable in size

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The BILLING INFORMATION is variable in size and is the location which includes the TYPE: *BillingInfo* parameter.

Set IP address mandatory parameters

The mandatory parameters for a `Set_IP_Address` primitive are as follows:



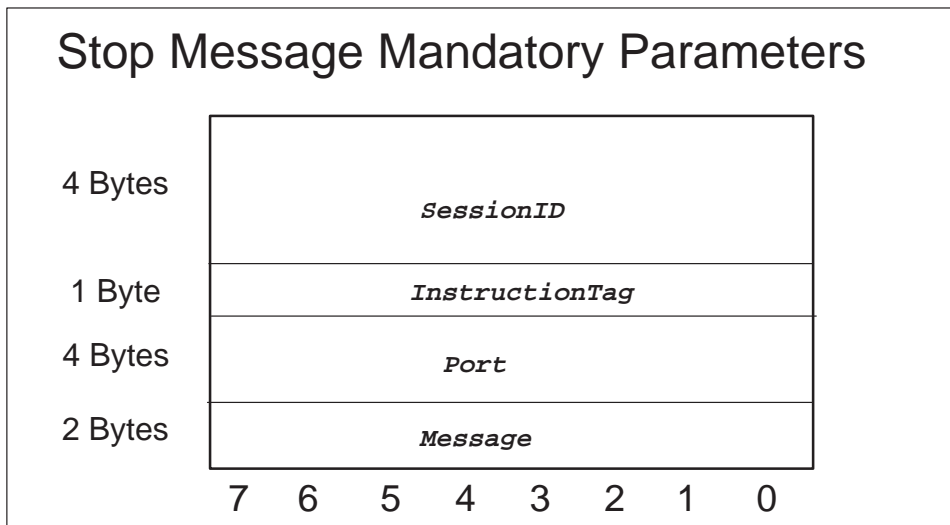
Parameter length: 13 bytes

Parameter contents:

- The PORT SERVICE INFO field consists of 11 bytes and is the location which includes the TYPE: *PortServiceInfo* parameter.
- The SEND AGENT DATA field consists of 1 bit and is the location which includes the TYPE: Boolean (TRUE = 1, FALSE = 0).
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Stop message mandatory parameters

The mandatory parameters for a `Stop_Message` primitive are as follows:



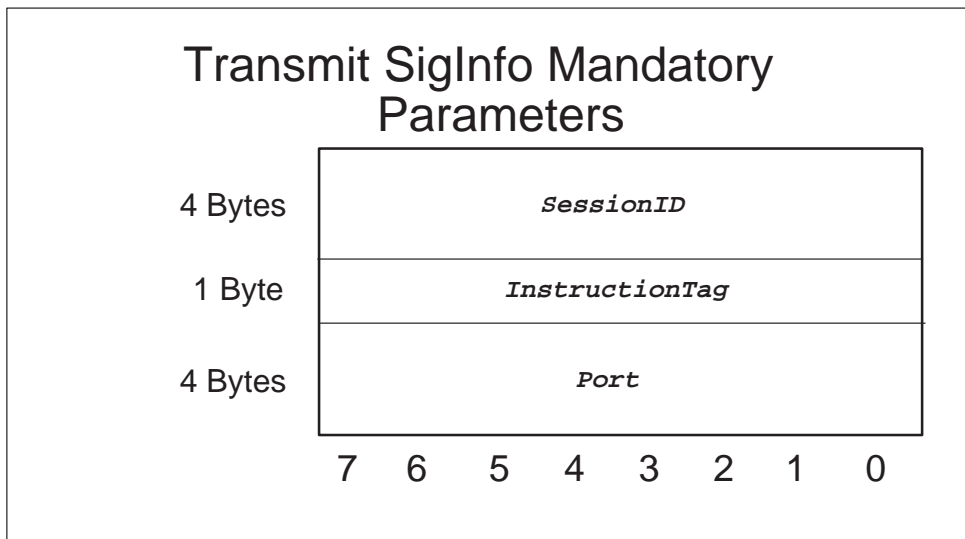
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Transmit siginfo mandatory parameters

The mandatory parameters for a `Transmit_siginfo` primitive are as follows:



Parameter length: 9 bytes

Parameter contents:

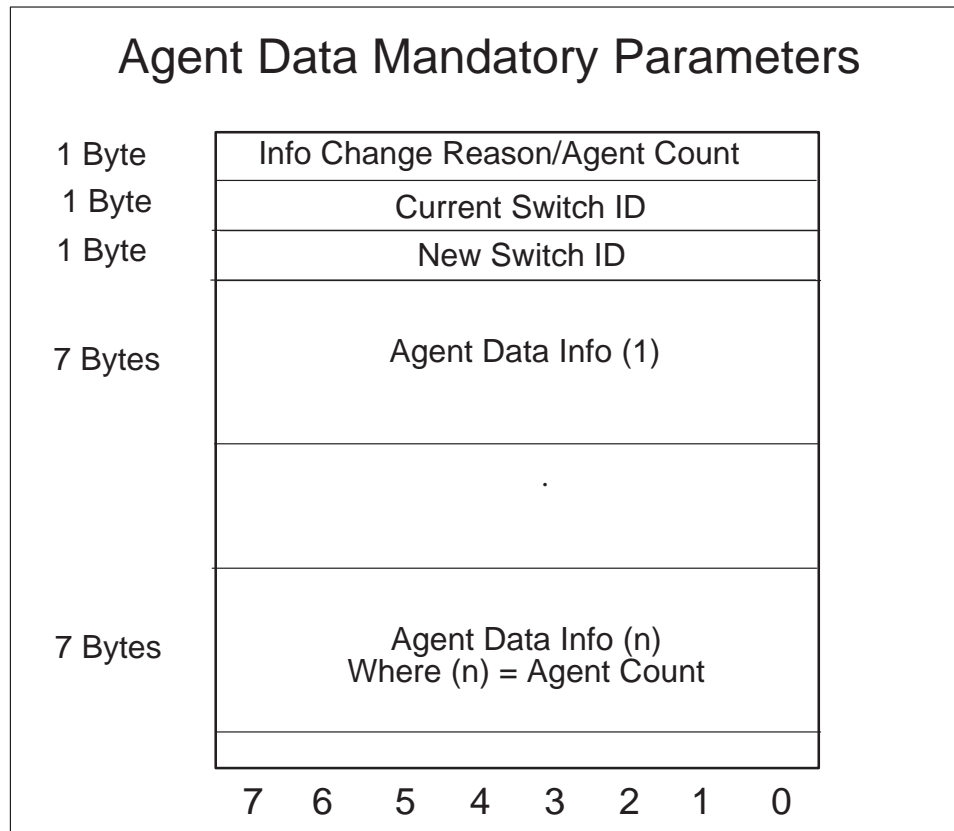
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

PSN LOPER mandatory parameters for events

The following mandatory parameters are used to build event notifications. Their description includes the parameter length as well as the type of each field in the parameter.

Agent data mandatory parameters

The mandatory parameters for an `Agent_Data` Event are as follows:



Parameter length: a minimum of 10 bytes and a maximum of 444 bytes

Parameter contents:

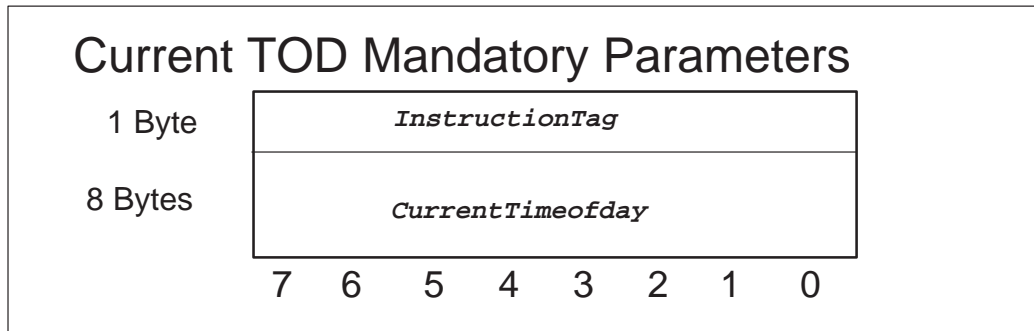
- The INFO CHANGE REASON/AGENT COUNT field consists of one byte and is the location which includes the TYPE: *InfoChangeReason* parameter.
- The CURRENT SWITCH ID field consists of one byte and is the location which includes the TYPE: *SwitchID* parameter.
- The NEW SWITCH ID field consists of one byte and is the location which includes the TYPE: *SwitchID* parameter.

- The AGENT DATA INFO 1 field consists of seven bytes and is the location which includes the TYPE: *AgentDataInfo* parameter.
- The AGENT DATA INFO n field consists of seven bytes and is the location which includes the TYPE: *AgentDataInfo* parameter.

Note: The number of *AgentDataInfo* parms must equal the Agent Count Value in the first byte.

Current time of day (TOD) mandatory parameters

The mandatory parameters for a `Current_TOD` Event are as follows:



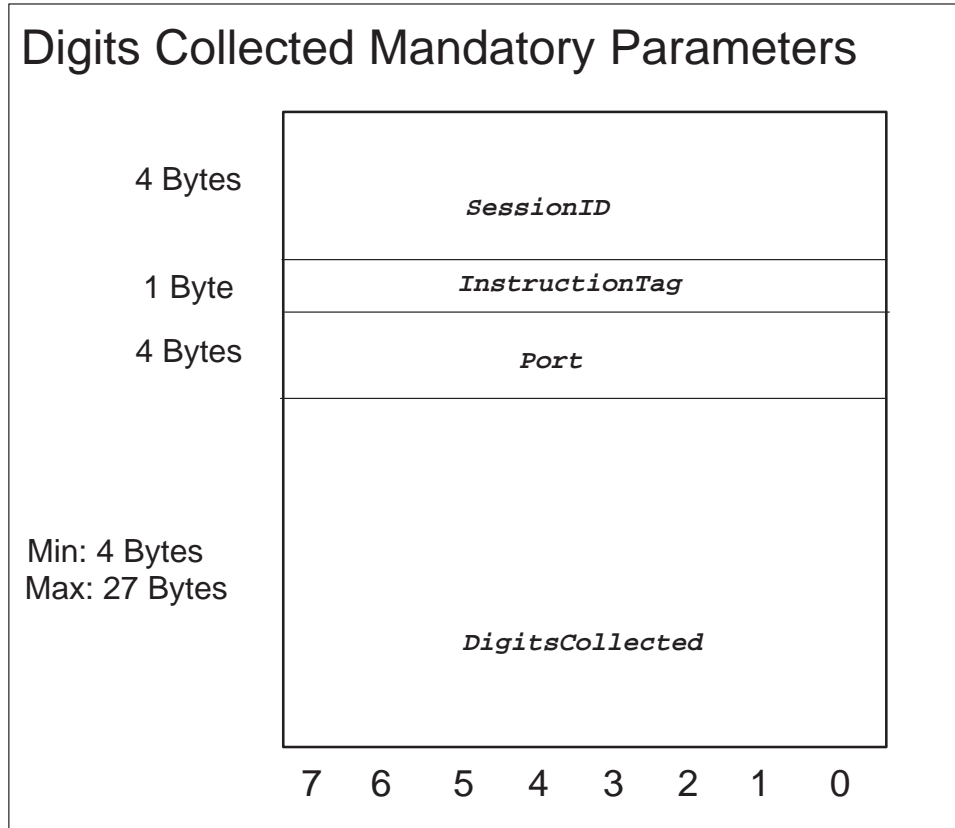
Parameter length: 9 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The CURRENT TIME OF DAY field consists of eight bytes and is the location which includes the TYPE: *Timeofday* parameter.

Digits collected mandatory parameters

The mandatory parameters for a `Digits_Collected` Event are as follows:



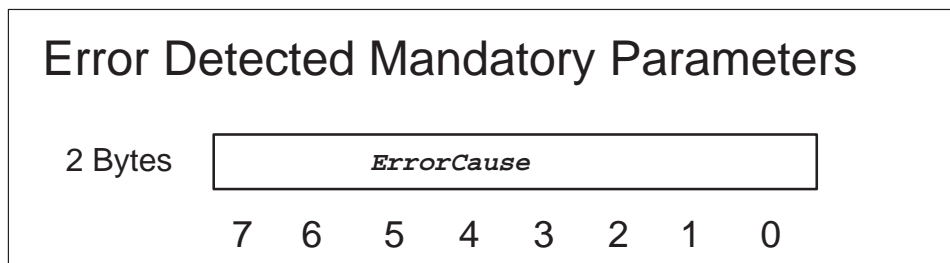
Parameter length: a minimum of 13 bytes and a maximum of 36 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS COLLECTED field consists of a minimum of four bytes and a maximum of 27 bytes and is the location which includes the TYPE: *DigitsCollected* parameter.

Error detected mandatory parameters

The mandatory parameters for a **Error_Detected** Event are as follows:



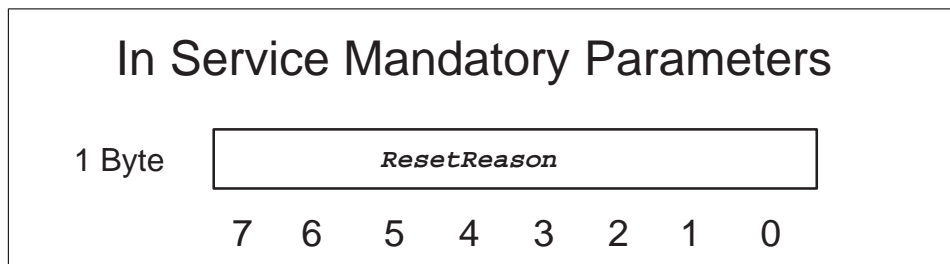
Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

In service mandatory parameters

The mandatory parameters for a **In_service** Event are as follows:



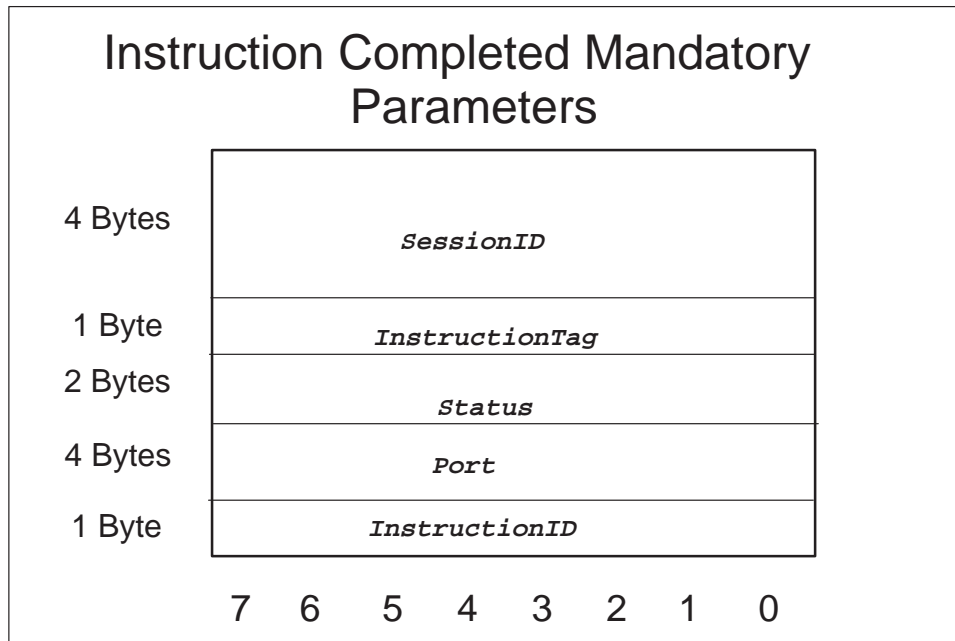
Parameter length: one byte

Parameter contents:

- The RESET REASON field consists of one byte and is the location which includes the TYPE: *ResetReason* parameter.

Instruction completed mandatory parameters

The mandatory parameters for a `Instruction_Completed` event are as follows:



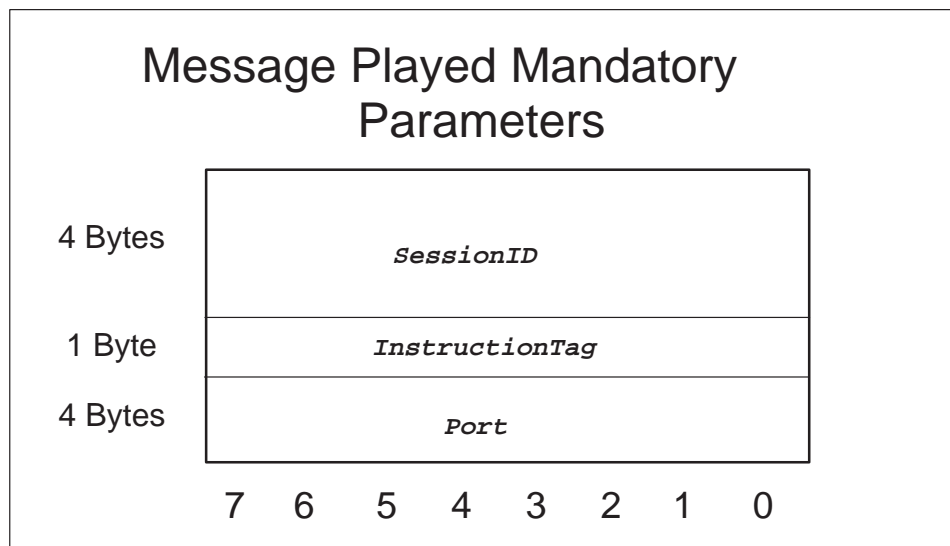
Parameter length: 12 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The INSTRUCTION ID field consists of one byte and is the location which includes the TYPE: *InstructionID* parameter.

Message played mandatory parameters

The mandatory parameters for a **Message_Played** Event are as follows:



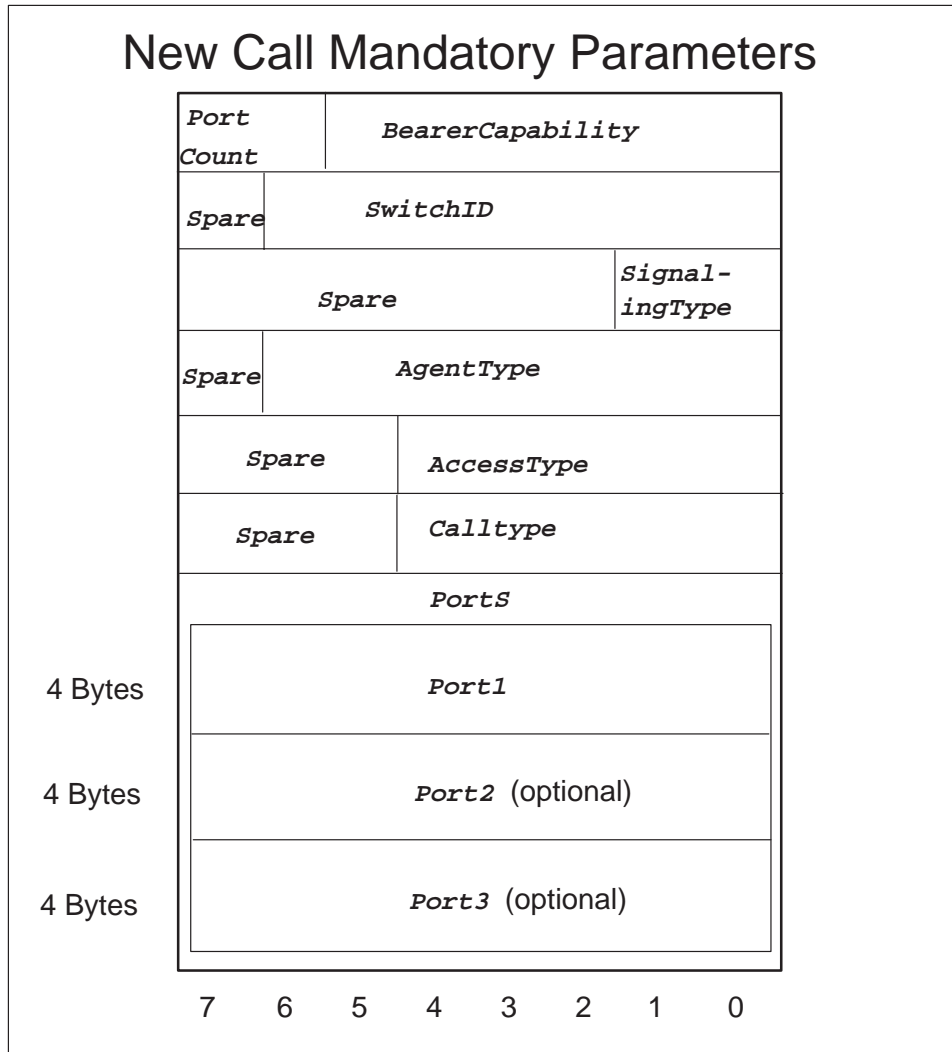
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call mandatory parameters

The mandatory parameters for a *New_Call* event are as follows:



Parameter length: minimum of 10 bytes and a maximum of 18 bytes

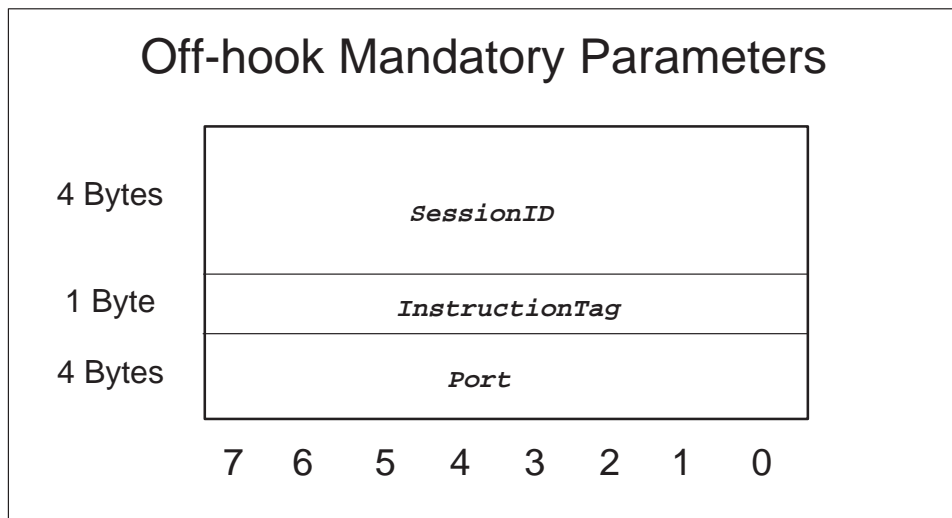
Parameter contents:

- The BEARER CAPABILITY field consists of six bits and is the location which includes the TYPE: *BearerCapability* parameter.
- The PORT COUNT field consists of two bits. The values include: 0 to 3.
- The SWITCH ID field consists of seven bits and is the location which includes the TYPE: *SwitchID* parameter.

- The SIGNALING TYPE field consists of one byte and is the location which includes the TYPE: *signalingType* parameter.
- The AGENT TYPE field consists of one byte and is the location which includes the TYPE: *AgentType* parameter.
- The ACCESS TYPE field consists of five bits and is the location which includes the TYPE: *AccessType* parameter.
- The CALL TYPE field consists of five bits and is the location which includes the TYPE: *Calltype* parameter.
- The PORTS consists of a minimum of 1 port to a maximum of 3 ports.
- The Port TYPE: *PortInfo* parameter.

Off-hook mandatory parameters

The mandatory parameters for an *Off_Hook* event are as follows:



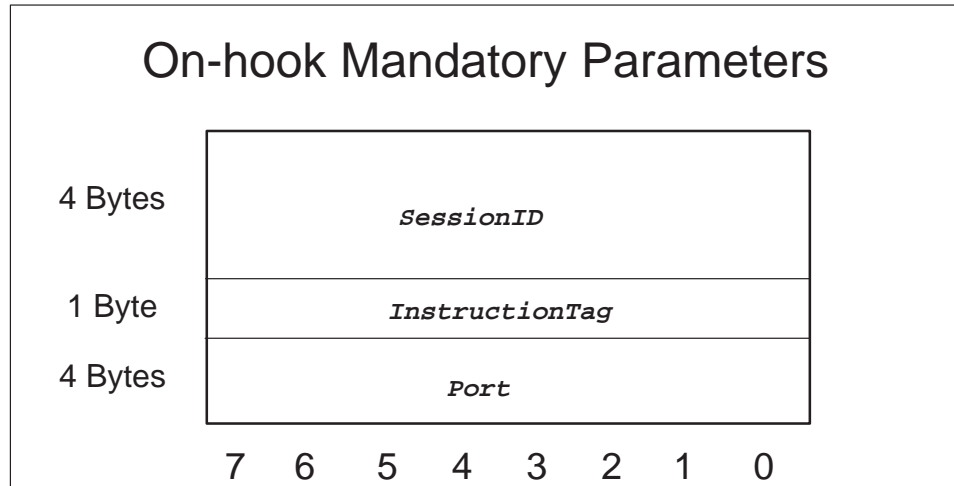
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

On-hook mandatory parameters

The mandatory parameters for an `On_Hook` Event are as follows:



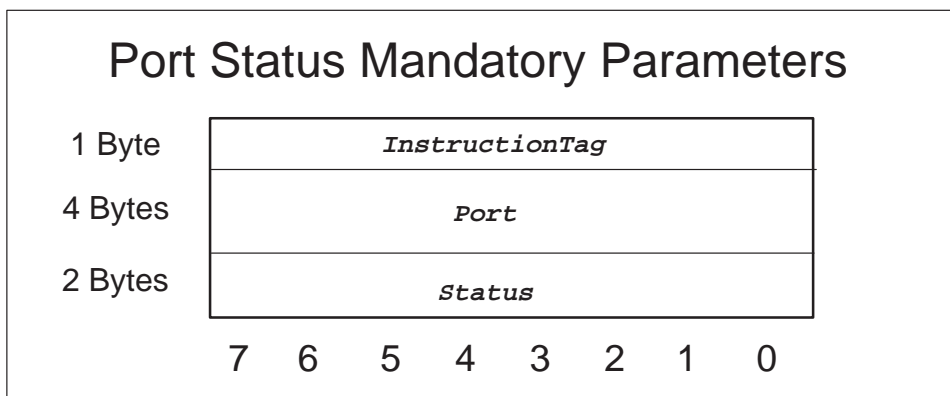
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` event are as follows:



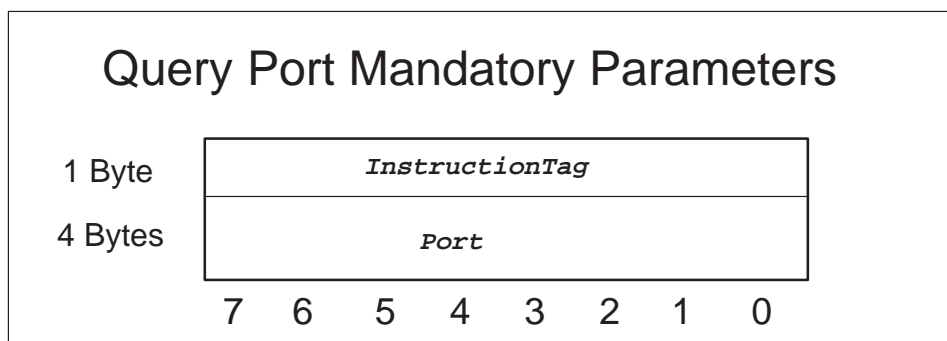
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` event are as follows:



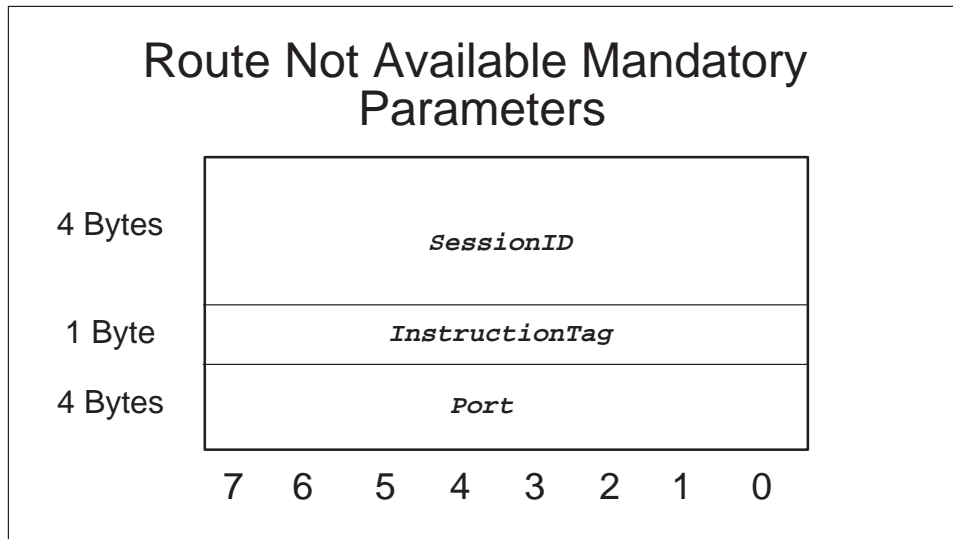
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route not available mandatory parameters

The mandatory parameters for a *Route_Not_Available* Event are as follows:



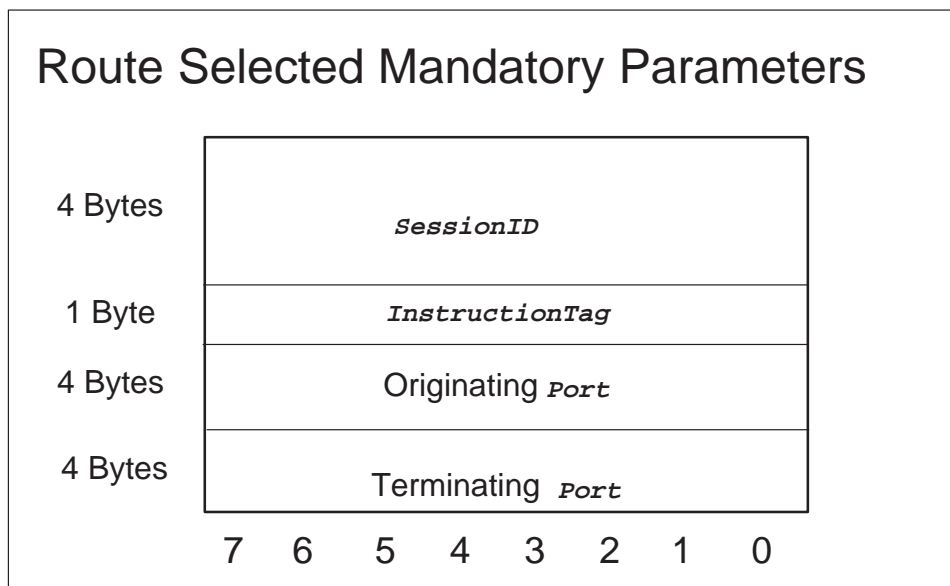
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route selected mandatory parameters

The mandatory parameters for a `Route_Selected` event are as follows:



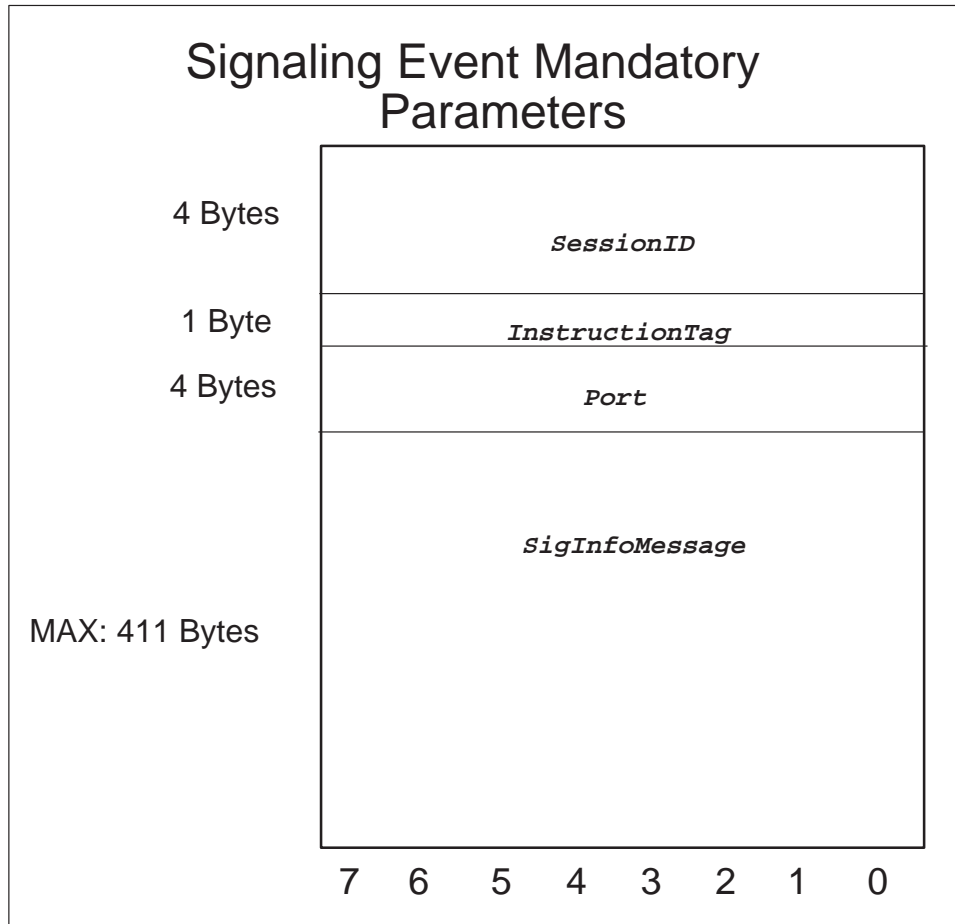
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The ORIGINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TERMINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Signaling event mandatory parameters

The mandatory parameters for a **signaling** Event are as follows:



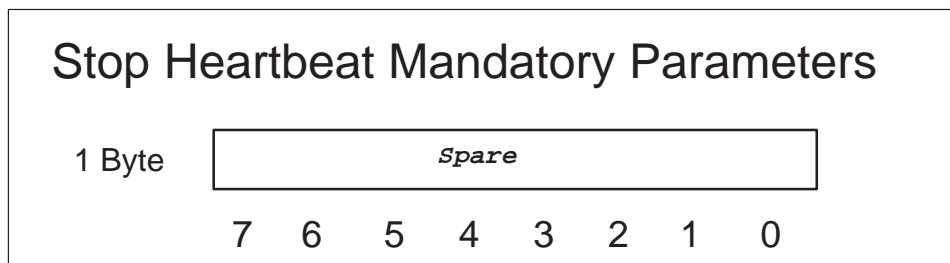
Parameter length: MAX 420 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The SIGINFO MSG field consists of a maximum of 411 bytes and is the location which includes the TYPE: *SignalingInfo* parameter.

Stop heartbeat mandatory parameters

The mandatory parameters for an `stop_Heartbeat` event are as follows:



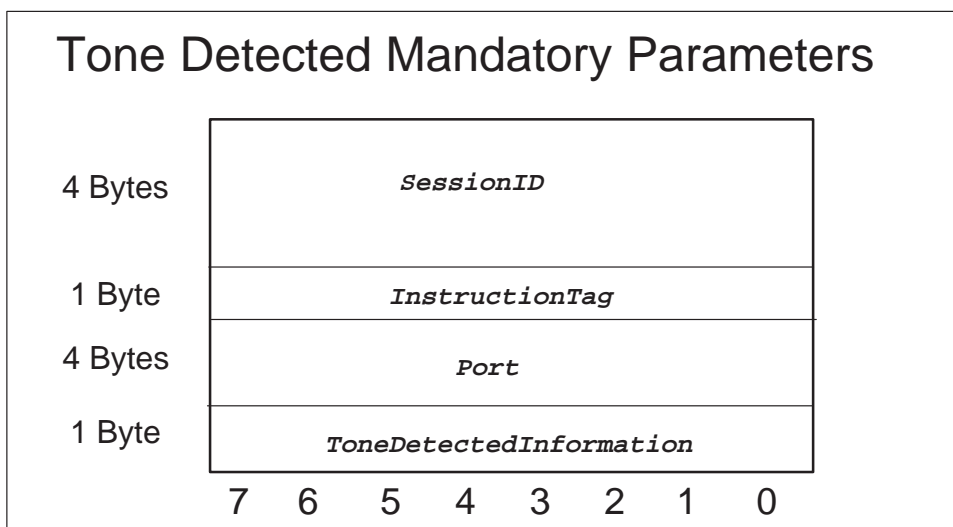
Parameter length: 1 byte

Parameter contents:

- The Spare as follows field consists of one byte and is the location which includes the To make message word aligned.

Tone detected mandatory parameters

The mandatory parameters for a `Tone_Detected` event are as follows:



Parameter length: 10 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TONE DETECTED INFORMATION field consists of one byte and is the location which includes the TYPE: *ToneDetected* parameter.

UCS PSN parameters version 3

This chapter contains a detailed description of the UCS PSN parameters, version 3. These PSN parameters may be a part of a PSN primitive, a PSN event notification, or both. The use of these parameters depend upon the location of the primitives and the event notifications.

PSN peer-to-peer application protocol

A new peer-to-peer application protocol has been created for the PSN platform. This protocol defines the primitives that are provided to allow the SCU to control calls on the PSN, and the event notifications which are reported to the SCU from the PSN. In addition, this protocol defines the parameters and their respective primitives and event notifications.

The peer-to-peer application protocol definition uses generic, functional references to data, rather than specific PSN data requirements. In addition, this protocol, including the primitive and event notification definitions, parameter definitions, and message flow definitions (with service implementation examples), is described later in this document.

PSN parameter definitions

The SCU sends instructions through the respective primitives to the PSN, which enables it to control the service call. In conjunction, the PSN notifies the SCU of the events that occur at the switch. The instructions and event notification messages contain one or more parameters with the appropriate information.

Each parameter has one or more bytes containing the parameter information. The division of parameter contents into fields, and the encoding of these fields, is covered in detail within this chapter.

The primitives or event notification messages in which the parameter may be a part of, is described earlier.

Tables 18-1 through 18-6 illustrates the primitives and event notifications, including the associated parameters. Each parameter is marked with an “M” for mandatory, an “O” for optional, or a “-” if the parameter is not associated with the primitive or event notification.

Table 18-1
Parameters for call control primitives (Part 1)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n i n f o
Parameters															
<i>AccessType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>BearerCapability</i>	-	-	O	-	-	-	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	O	O	O	O	O	O	O	O	O	O	O	M	O	O
<i>CallType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ControlInfo</i>	-	O	O	O	O	O	O	O	-	O	O	O	O	O	O
<i>DestinationTrunkGroup</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	M	-	-	-	-	-	-	-	-	M	-	-	-	-

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n I n f o
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>DigitstoOutpulse</i>	-	-	M	-	-	-	-	-	-	-	-	-	-	-	O
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M

Table 18-2
Parameters for call control primitives (Part 2)

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g n a l I n f o
Parameters															
<i>ISUPIndex</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>MessageInfo</i>	O	-	-	-	-	-	-	-	-	M	M	-	-	M	-
<i>MonitorMask</i>	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
<i>PortServiceInfo</i>	-	O	O	O	O	O	O	M	-	O	O	O	O	O	O
<i>PortStatus</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ServingTranslatio nScheme</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	B r i d g e	C o l l e c t - D i g i t s - & - R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w - C a l l - A c c e p t e d	N e w - C a l l - R e j e c t e d	P l a y - M e s s a g e	P P C D	R e c o n n e c t	S e t - B i l l i n g - R e c o r d	S t o p - M e s s a g e	T r a n s m i t - S i g I n f o
<i>SessionID</i>	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M
<i>SignalingInfo</i>	-	-	M / O	O	-	-	-	-	-	-	-	-	-	-	M
<i>SignalingType</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	O
<i>SwitchID</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>TimeOfDay</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note: The *signalingInfo* parameter is mandatory if the signaling type of the port is PRI.

Table 18-3
Parameters for event notifications (Part 1)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>AccessType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>CallType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	O	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	M	-	-	-	O	-	-	-	-	-	-
<i>DigitstoOutpulse</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	M	-
<i>ErrorCause</i>	-	M	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l l i n g - E v e n t	T o n e - D e t e c t e d
<i>FlowControlEncountered</i>	-	-	-	-	O	-	-	-	-	-	-
<i>InstructionID</i>	-	O	M	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	M	O	M	M	M	M	M	M	M	M	M

Table 18-4
Parameters for event notifications (Part 2)

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
Parameters											
<i>ISUPIndex</i>	-	-	-	-	O	-	-	-	-	-	-
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	O	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	M	O	M	M	M	M	M	M	M	M	M
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortStatus</i>	-	O	-	-	-	-	-	-	-	-	-
<i>ResetReason</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SessionID</i>	M	O	M	M	-	M	M	M	M	M	M
<i>SignalingInfo</i>	-	-	-	-	O	O	O	-	-	M	-
<i>SignalingType</i>	-	-	-	-	M	-	-	-	-	-	-
<i>SwitchID</i>	-	-	-	-	M	-	-	-	-	-	-

	D i g i t s - C o l l e c t e d	E r r o r - D e t e c t e d	I n s t r u c t i o n - C o m p l e t e d	M e s s a g e - P l a y e d	N e w - C a l l	O f f - h o o k	O n - h o o k	R o u t e - N o t - A v a i l a b l e	R o u t e - S e l e c t e d	S i g n a l i n g - E v e n t	T o n e - D e t e c t e d
<i>Timeofday</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	M

Table 18-5
Parameters for non-call related events and primitives (Part 1)

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
Parameters											
<i>AccessType</i>	-	-	-	-	-	-	-	-	-	-	-
<i>AgentDataInfo</i>	M	-	-	-	-	-	-	-	-	-	-
<i>AgentType</i>	-	-	-	-	-	-	-	-	-	-	-
<i>BillingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>Calltype</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InfoChangeReason</i>	M	-	-	-	-	-	-	-	-	-	-
<i>COTRequired</i>	-	-	-	-	-	-	-	-	-	-	-
<i>CRID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ControlInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DestinationTrunkGroup</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollection</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitsCollected</i>	-	-	-	-	-	-	-	-	-	-	-
<i>DigitstoOutpulse</i>	-	-	-	-	-	-	-	-	-	-	-

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
<i>DigitsOutpulsed</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ErrorCause</i>	-	-	-	-	-	-	-	-	-	-	-
<i>FlowControlInfo</i>	-	-	O	-	-	-	-	-	-	-	-
<i>FlowControlEncountered</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>InstructionTag</i>	-	M	M	-	-	M	M	M	M	M	-

Table 18-6
Parameters for non-call related events and primitives (Part 2)

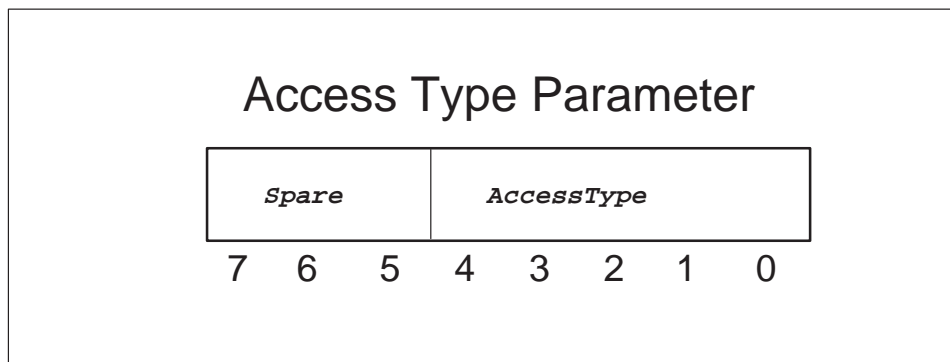
	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n - S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
Parameters											
<i>ISUPIndex</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MessageInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>MonitorMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>ParameterID</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PointInCall</i>	-	-	-	-	-	-	-	-	-	-	-
<i>PortInfo</i>	-	-	-	-	-	M	M	-	-	M	-
<i>PortServiceInfo</i>	-	-	-	-	-	-	-	-	M / 0	M	-
<i>PortStatus</i>	-	-	-	-	-	M	-	-	-	-	-
<i>ResetReason</i>	-	-	-	M	-	-	-	-	-	-	-
<i>SessionID</i>	-	-	-	-	-	O	O	-	-	-	-
<i>SignalingInfo</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SignalingType</i>	-	-	-	-	-	-	-	-	-	-	-

	A g e n t - D a t a	C u r r e n t - T i m e - o f - D a y	F l o w - C o n t r o l	I n - S e r v i c e	H e a r t b e a t	P o r t - S t a t u s	Q u e r y - P o r t	Q u e r y - T i m e - o f - D a y	R e s e t - S w i t c h	S e t - I P - A d d r e s s	S t o p - H e a r t b e a t
<i>SigInfoMask</i>	-	-	-	-	-	-	-	-	-	-	-
<i>SwitchID</i>	M	-	-	-	-	-	-	-	-	-	-
<i>Timeofday</i>	-	M	-	-	-	-	-	-	-	-	-
<i>ToneDetected</i>	-	-	-	-	-	-	-	-	-	-	-

Note: There is one mandatory PSI parameter called the Return IP Address parameter, which is used to send the **Instruction_Completed**. There are 0 to 20 optional PSI parameters called the IP Address to Reset parameters, which are used to do the actual resetting of calls.

Access type

This information is sent in the **New_Call** event notification and contains the originating trunk information. The only values supported for this parameter are FGD, DAL, and PRI.



The *AccessType* parameter is mandatory for a **New_Call** event.

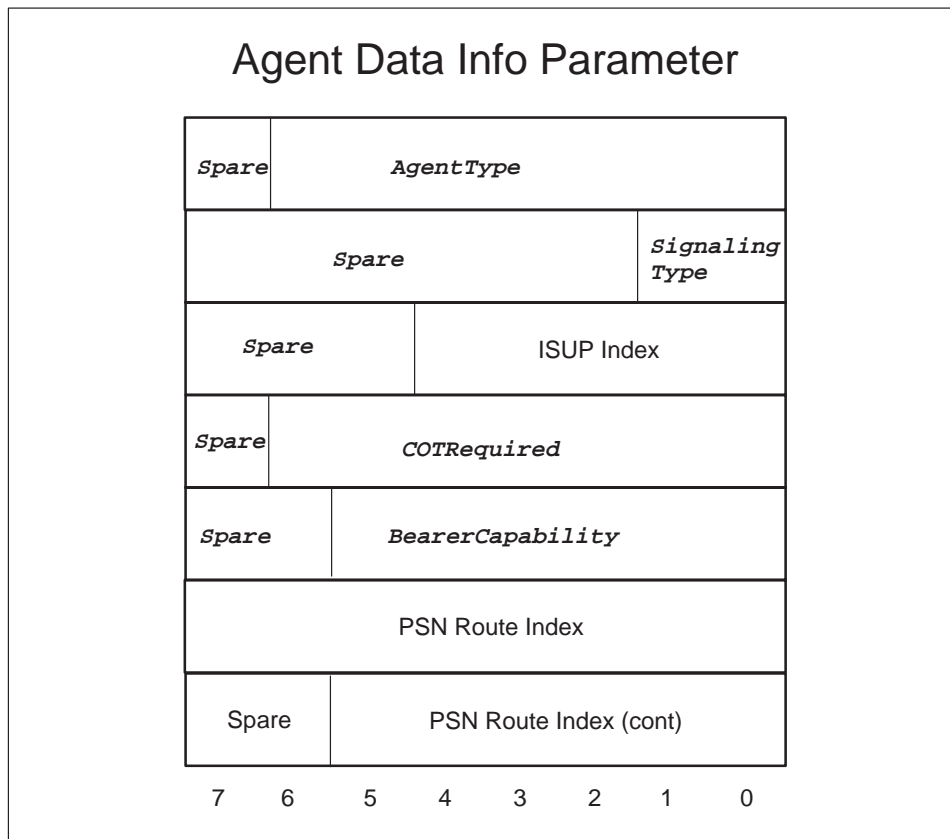
Parameter length: 1 byte

Parameter contents:

- The ACCESS TYPE field consists of three bits and the type of the port, which is encoded as follows:
 - 0 0 0 0: Unused (Spare)
 - 1 0 0 1: DAL 2-wire
 - 2 0 1 0: DAL 4-wire
 - 3 0 1 1: PTS FGD
 - 4 1 0 0: SS7 FGD
 - 5 1 0 1: PTS IMT
 - 6 1 1 0: SS7 IMT
 - 7 1 1 1: PRI

Agent data info

This parameter is only used in the **Agent_Data** event and the **New_Call** event.



Length: 7 bytes

Parameter contents:

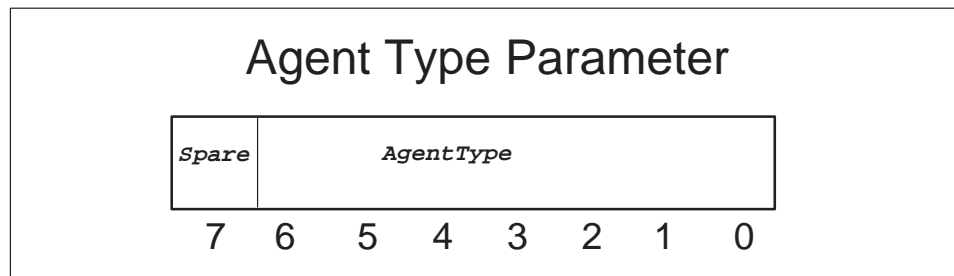
- The AGENT TYPE field consists of seven bits with *AgentType* parameter contents.
- The SIGNALING TYPE field consists of two bits with *SignalingType* parameter contents.
- The ISUP INDEX field consists of five bits with ISUP *Index* parameter contents.
- The COT REQUIRED PERCENT field consists of seven bits with *COTRequired* parameter contents.

- The PULSE TYPE field consists of two bits. This field is the pulse type of the port. Values include:
 0 0: Unused (Spare)
 0 1: MF
 1 0: DTMF
 1 1: Unused (Spare)
- The BEARER CAPABILITY field consists of six bits with *BearerCapability* parameter contents.
- The PSNROUTE INDEX consists of 14 bits and is the actual index into the table, PSNROUTE.

Note: This index is 0 when an *Agent_Data* event occurs informing the SCU of a change in only the switch ID. If the new switch ID and current switch ID differ, all other fields of the *AgentDataInfo* parameter should be ignored. At all other times (when no switch ID change occurs), this field has a value from 1 to 9999. When this occurs, all of the *AgentDataInfo* parameter fields may be considered valid.

Agent type parameter

This parameter is used in the *Agent_Data* event and in the *New_Call* event.



Optional parameter ID: 29

Length: 1 byte

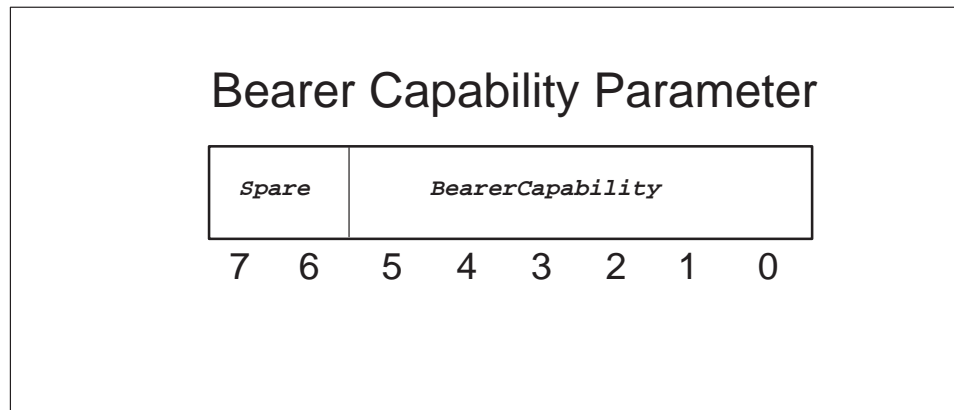
Parameter contents:

- The AGENT TYPE field consists of seven bits and allows the SCU to associate a generic trunk naming convention for each agent in the table, PSNROUTE. The values include:
 0 0 0 0 0 0 0: Unused
 1 0 0 0 0 0 1: DAL Trunk
 2 0 0 0 0 1 0: Unused
 3 0 0 0 0 1 1: ONAT Trunk
 4 0 0 0 1 0 0: EANT Trunk
 0 0 0 0 1 0 1 to 0 0 0 0 1 1 0: Unused

0 0000111: IMT Trunk
 1 0001000: Unused
 2 0001001: OP250 Trunk
 0001010 to 0001100: Unused
 0 0001101: PRA250 Trunk
 0001110 to 1111111: Unused

Bearer capability

The information below describes the *BearerCapability* parameter.



Optional parameter ID: 1

Parameter length: 6 bits

Parameter contents:

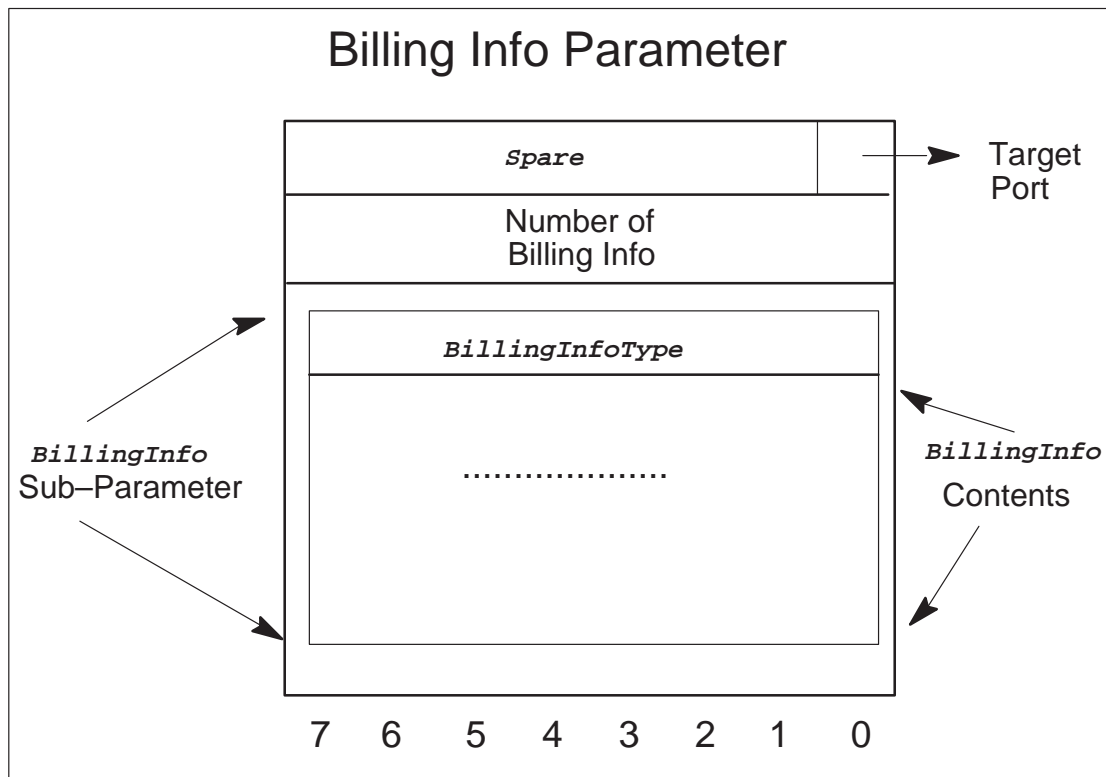
- The BEARER CAPABILITY field consists of six bits and indicates the speed and type of information that is to be carried by the call. The values include:

0 000000: Unused
 1 000001: Speech
 2 000010: 64kbyte Data
 3 000011: 64kbyte X25
 4 000100: 56kbyte Data
 5 000101: Data Unit
 6 000110: 64kbyte Restricted
 7 000111: 3.1 kHz
 8 001000: 7 kHz
 9 001001: Voice Data
 10 001010: 64kbyte Rate Data
 11 001011: 32kbyte Speech
 12 001100: Wideband

0 0 1 1 0 1 to 1 1 1 1 1 1: Spare values

Billing info

This parameter contains the billing information for the port in the service call. In addition, The *BillingInfo* parameter may be used to update multiple billing record fields for a port.



Optional parameter ID: 2

Parameter length: variable in size–Maximum: 12 bytes

Parameter contents:

- The TARGET PORT field is one byte and indicates to what port a primitive is being sent. This is necessary since some primitives (for example Connect) involve more than one port in a call.
 - 0 0 : Port_A This is the default value
 - 1 1 : Port_B Valid only when used with the Connect and/or Reconnect
 - 00000010 to 11111111 : Spare Values

- The NUMBER OF BILLING INFO field is one byte and determines the Number of Billing Info sub-parameters to follow. Each Number of Billing Info sub-parameter consists of the Billing Info Type and the Billing Info contents. The maximum number of Billing Info in Billing Info Parameter is one.

0 00000000 : Zero Number of Billing Info

1 00000001 : One Billing Info

00000010 to 11111111 : Spare Values

- The BILLING INFO TYPE field is one byte and indicates how to interpret the Billing Info contents. Each application may define its own set of types. As a result, the interpretation of the Billing Info contents depends upon the type.

0 00000000: Unused

1 00000001: CRID

00000010 to 11111111: Spare values

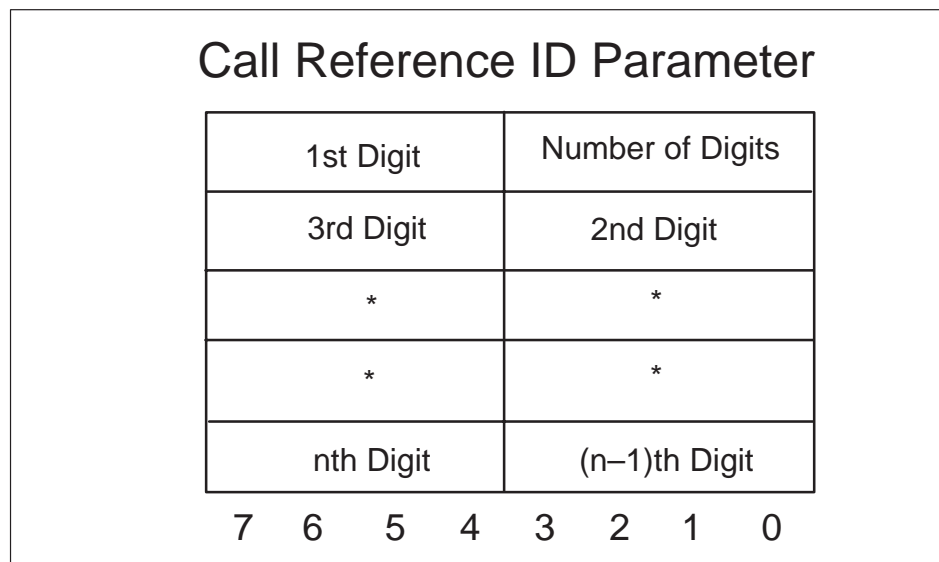
- The BILLING INFO CONTENTS field contains the billing information that is ultimately placed in the billing record.

This parameter supports the following billing type:

Call Reference ID (CRID)

Call reference ID (CRID)

This Parameter is used for the transport of the Call Reference Identifier. The CRID digits are encoded in a variable length digits field in TBCD format. The valid range of digits for the *CRID* parameter is 1 to 9 digits.



Optional parameter ID: 3

Parameter length: Variable in Size– Maximum of 5 bytes

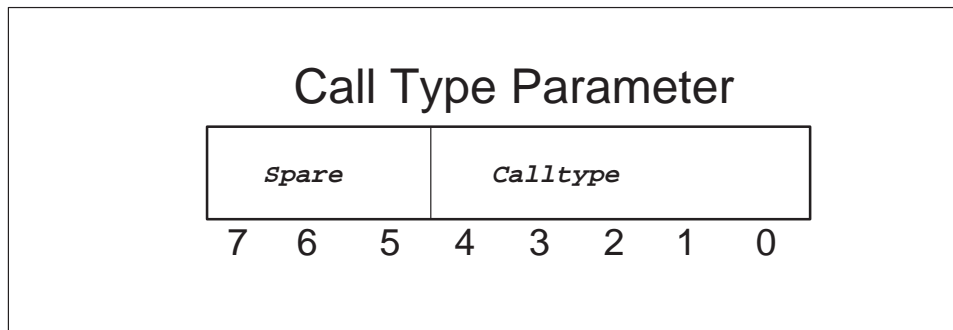
Parameter contents:

- The NUMBER OF DIGITS field consists of four bits and contains the number of CRID digits. The values include: 1 to 9.
- The 1ST DIGIT to 9TH DIGIT field consists of four bits (each digit) with all 9 digits in TBCD format, which are encoded as follows:

0 0 0 0: Filler
 1 0 0 0 1: Digit 1
 2 0 0 1 0: Digit 2
 3 0 0 1 1: Digit 3
 4 0 1 0 0: Digit 4
 5 0 1 0 1: Digit 5
 6 0 1 1 0: Digit 6
 7 0 1 1 1: Digit 7
 8 1 0 0 0: Digit 8
 9 1 0 0 1: Digit 9
 10 1 0 1 0: Digit 0
 11 1 0 1 1: *
 12 1 1 0 0: #
 13 1 1 0 1: D
 14 1 1 1 0: E
 15 1 1 1 1: F

Call type

This parameter contains the Call Type for a call when it is determined by the PSN that the call is to be controlled by the SCU.



Mandatory parameter for a **New_Call** event.

Parameter length: 1 byte

Parameter contents:

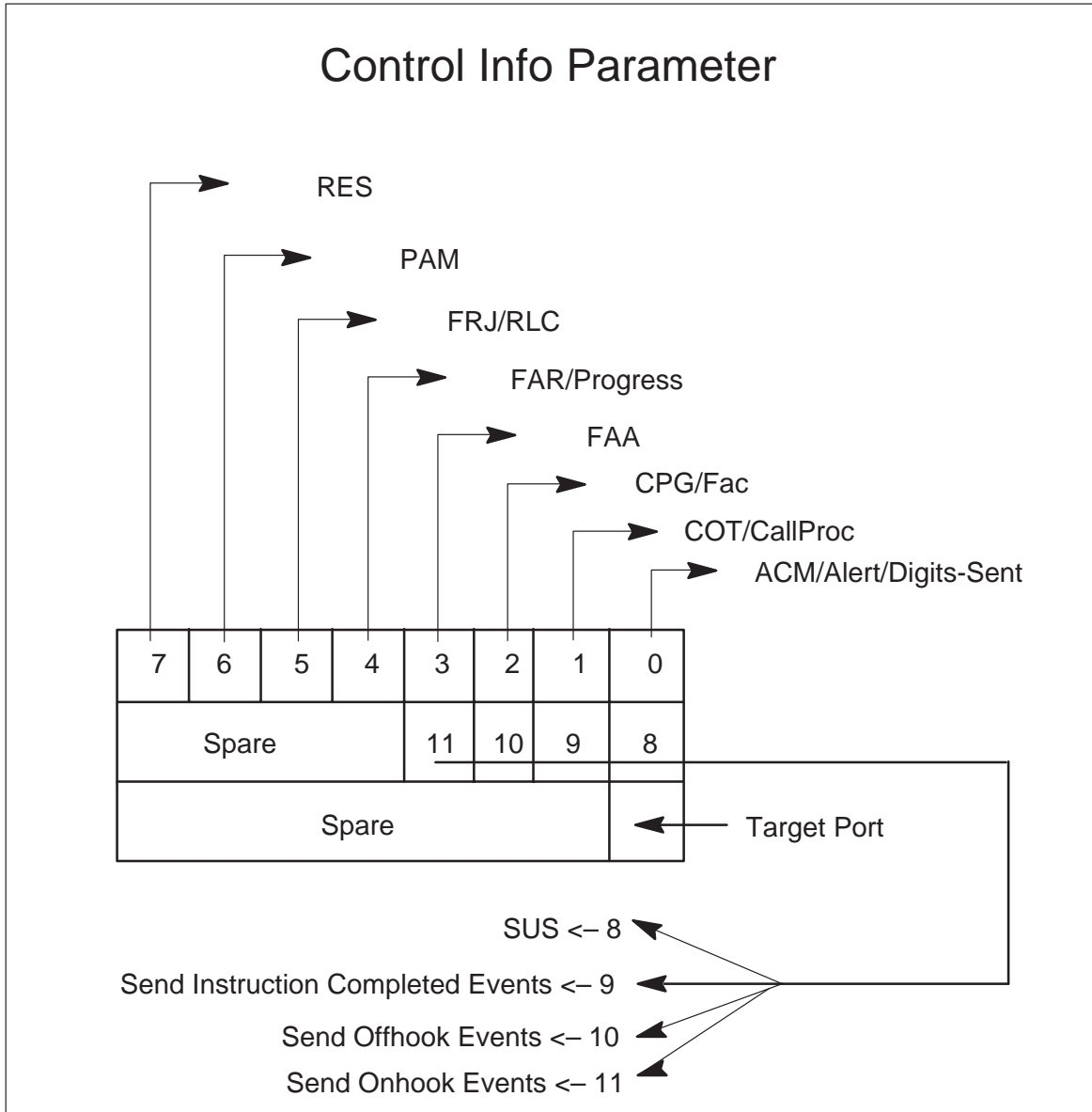
- The CALL TYPE field consists of 5 bits provides information on the type of call being made. The values include:

0 0 0 0 0: Undetermined *
1 0 0 0 1: Onnet *
2 0 0 0 1 0: Offnet *
3 0 0 0 1 1: Public Speed
4 0 0 1 0 0: Private Speed
5 0 0 1 0 1: Hotline Speed
6 0 0 1 1 0: N00*
7 0 0 1 1 1: Zero Plus – Onnet
8 0 1 0 0 0: Zero Plus – Offnet
9 0 1 0 0 1: INTOA *
0 1 0 1 0 to 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

Control info parameter

This parameter contains the action to be taken if the primitive is successfully processed.



Optional parameter ID: 4

Parameter length: 3 Bytes

Parameter contents:

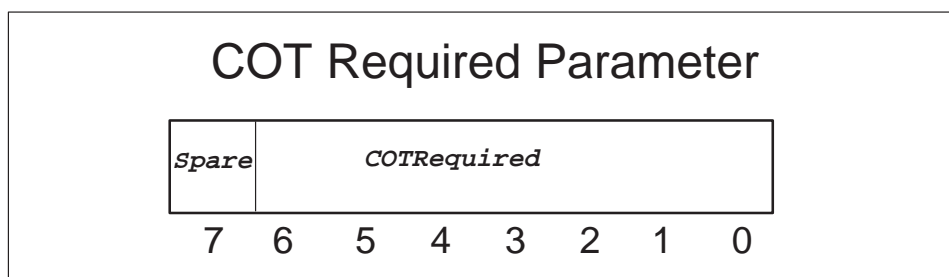
Note: The first nine bits are the contents of what was formerly the *SigInfoMask* parameter.

- ACM/Alert: 1 Bit
 - Values: 0 = do not send ACM (for SS7 port) or Alert (for PRI port) message to the SCU
 - 1 = send the ACM (for SS7 port) or Alert (for PRI port) to the SCU
- COT/CallProc: 1 Bit
 - Values: 0 = do not send COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU
 - 1 = send the COT (for SS7 port) or Call Proceeding (for PRI port) to the SCU
- CPG/FAC: 1 Bit
 - Values: 0 = do not send CPG (for SS7 port) or FAC (for PRI port) message to the SCU
 - 1 = send the CPG (for SS7 port) or FAC (for PRI port) to the SCU
- FAA: 1 Bit
 - Values: 0 = do not send FAA (for SS7 port) message to the SCU
 - 1 = send the FAA (for SS7 port) to the SCU
- FAR/Progress: 1 Bit
 - Values: 0 = do not send FAR (for SS7 port) or Progress (for PRI port) message to the SCU
 - 1 = send the FAR (for SS7 port) or Progress (for PRI port) to the SCU
- FRJ/RLC: 1 Bit
 - Values: 0 = do not send FRJ (for SS7 port) or RLC (for PRI port) message to the SCU
 - 1 = send the FRJ (for SS7 port) or RLC (for PRI port) to the SCU
- PAM: 1 Bit
 - Values: 0 = do not send PAM (for SS7 port) message to the SCU
 - 1 = send the PAM (received on the SS7 port) to the SCU
- RES: 1 Bit
 - Values: 0 = do not send RES (for SS7 port) message to the SCU
 - 1 = send the RES (received on the SS7 port) to the SCU

- SUS: 1 Bit
 - Values: 0 = do not send SUS (for SS7 port) message to the SCU
 - 1 = send the SUS (received on the SS7 port) to the SCU
- The SEND ONHOOK EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any **On_Hook** events that occur on the agent. The values include:
 - 0 (OFF) Do not send any **On_Hook** events.
 - 1 (ON) Send all **On_Hook** events.
- The SEND OFFHOOK EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any **Off_Hook** events that occur on the agent. The values include:
 - 0 (OFF) Do not send any **Off_Hook** events.
 - 1 (ON) Send all **Off_Hook** events.
- The SEND INSTRUCTION COMPLETED EVENTS field consists of one bit and is a boolean which indicates that the agent should, or should not, send any events that occur on the agent. The values include:
 - 0 (OFF) Do not send any **Instruction_Completed** events.
 - 1 (ON) Send all **Instruction_Completed** events.
- The TARGET PORT field is one byte and indicates to what port a primitive is being sent. This is necessary since some primitives (for example Connect) involve more than one port in a call.
 - 0 0 : Port_A This is the default value
 - 1 1 : Port_B Valid only when used with the Connect and/or Reconnect
 - 00000010 to 11111111 : Spare Values

COT required parameter

This parameter is used in the **Agent_Data** event and in the **New_Call** event.



Optional parameter ID: 30

Length: 1 byte

Parameter contents:

- The COT REQUIRED PERCENT field consists of seven bits and represents the percentage of the SS7 PSN agent's trunk members, which are datafilled to have COT testing performed on them. The range of this parameter is from 0, for no COT testing to 100, for COT testing on each member.

CRID Parameter

This parameter is used in the New Call Event.

CRID Parameter							
Spare				Count			
Spare				Spare			
2nd Digit				1st Digit			
4th Digit				3rd Digit			
6th Digit				5th Digit			
8th Digit				7th Digit			
Spare				9th Digit			
7	6	5	4	3	2	1	0

Optional parameter ID: 30

Parameter length: 7 bytes

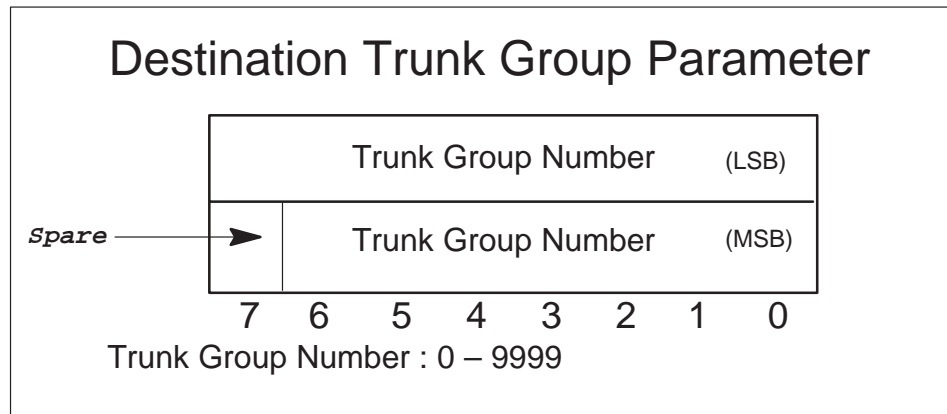
Parameter contents:

- The COUNT field indicates the number of actual CRID digits stored within the parameter.
- The 1ST DIGIT to 9TH DIGIT field consists of four bits (each digit) of n digits. These DTMF digits are encoded in the TBCD format as follows:

- 0 0 0 0 0: Filler
- 1 0 0 0 1: Digit 1
- 2 0 0 1 0: Digit 2
- 3 0 0 1 1: Digit 3
- 4 0 1 0 0: Digit 4
- 5 0 1 0 1: Digit 5
- 6 0 1 1 0: Digit 6
- 7 0 1 1 1: Digit 7
- 8 1 0 0 0: Digit 8
- 9 1 0 0 1: Digit 9
- 10 1 0 1 0: Digit 0
- 11 1 0 1 1: *
- 12 1 1 0 0: #
- 13 1 1 0 1: D
- 14 1 1 1 0: E
- 15 1 1 1 1: F

Destination trunk group

This parameter contains the external Trunk Group Number of the intended trunk. The call terminates to an idle member of this trunk group.



Optional parameter ID: 5

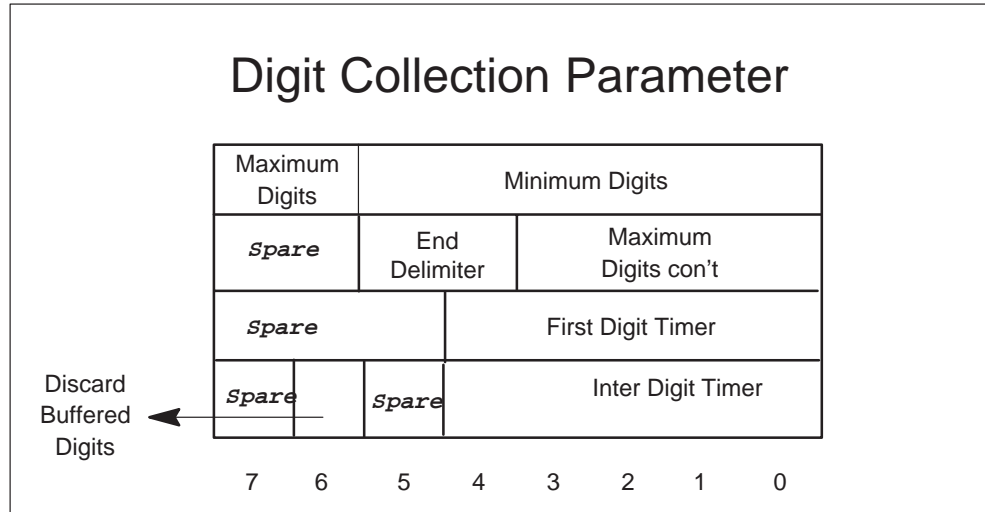
Parameter length: 2 bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the location of the external Trunk Group Number. The range is from 0 to 9999. The NIL_TRUNK_GROUP number is defined as 32,767.

Digit collection parameter

This parameter contains information required to collect digits on a specified port.



Optional parameter ID: 6

Parameter length: 4 bytes

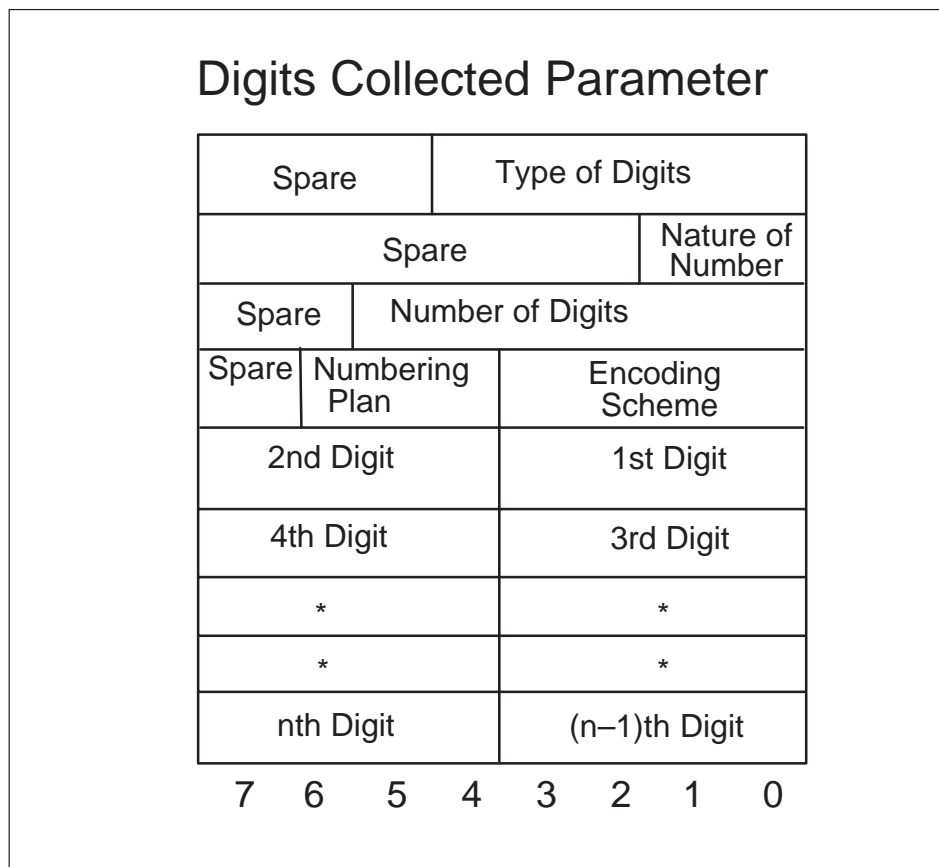
Parameter contents:

- The MINIMUM_DIGITS field consists of six bits and contains the minimum number of digits to collect. This value must be less than or equal to the MAXIMUM_DIGITS field. The values include the following (0 – 45):
 - 0 0 0 0 0 0: (Minimum Number of Digits to collect equals 0)
 - 0 0 0 0 0 1
 -
 - 1 0 1 1 0 1: (Minimum Number of Digits to collect equals 45)
 - 1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (spare)
- The MAXIMUM_DIGITS field consists of six bits and contains the Maximum Number of Digits to collect. The values include the following (0 – 45):
 - 0 0 0 0 0 0: (Maximum Number of Digits to collect equals 0)
 - 0 0 0 0 0 1
 -

- 1 0 1 1 0 1: (Maximum Number of Digits to collect equals 45)
- 1 0 1 1 1 0 to 1 1 1 1 1 1: Unused (spare)
- The END_DELIMITER field consists of two bits and identifies the End of Digits Delimiter. Upon detection of the End of Digits Delimiter, digit collection stops and the collected digits are reported. If the delimiter is dialed as the very first digit, then the digit collection stops immediately. The delimiter digit values include:
 - 0 0: No Delimiter Specified
 - 0 1: *
 - 1 0: #
 - 1 1: * and #
- The FIRST_DIGIT_TIMER field consists of five bits and specifies the time to wait until the first digit is dialed. The values include the following (2 – 30 seconds):
 - 0 0 0 0 0: Unused (spare)
 - 0 0 0 0 1: Unused (spare)
 - 0 0 0 1 0: Timer Value equals 2 second
 -
 - 1 1 1 1 0: Timer Value equals 30 seconds
 - 1 1 1 1 1: Unused (spare)
- The INTER_DIGIT_TIMER consists of five bits and specifies the Inter Digit Timer value, for example, the time to wait when collecting the second and subsequent digits. The values include the following (1 – 20 seconds):
 - 0 0 0 0 0: Unused (spare)
 - 0 0 0 0 1: Timer value equals 1 second
 -
 - 1 0 1 0 0: Timer Value equals 20 seconds
 - 1 0 1 0 1 to 11111: Unused (Spare)
- The DISCARD BUFFERED DIGITS field consists of one bit and is a bool, which if true, indicates that if the PSN had been buffering any digits prior to receiving this parameter, then it is to discard these digits and restart the digit collection process. Refer to Chapter “PSN finite state machine” for details on the buffering of digits.

Digits collected

This parameter contains the Digits Collected (or received) on the port or agent. The digits contained in this parameter are always in the TBCD format. Also included is COUNT, which specifies the number of digits included in this parameter.



Optional parameter ID: 7

Parameter length: Variable in size

Parameter contents:

Note: The contents of this parameter (including the Nature of Number, Numbering Plan, and Encoding Scheme fields) have been encoded in accordance with the following documents: TR-NWT-000317 Switching Systems Requirements for Call Control Using ISDNUP, TR-NWT-000394 Switching Systems Requirements for IEC Interconnection using ISDNUP, and TR-NWT-000444 Switching System Requirements Supporting ISDN Access Using the ISDNUP. However, there are some values marked “Private” which are application specific.

- The TYPE OF DIGITS field consists of five bits and contains the Type of Digits encoded as shown below. The Type of Digits is known when the collected digits are sent in the **New_Call** event notification. When this parameter is sent in the **Digit_Collected** event notification, a value of “unknown” is used.

0	0 0 0 0 0:	Unknown
1	0 0 0 0 1:	Called Party Address
2	0 0 0 1 0:	Calling Party Address (ANI)
3	0 0 0 1 1:	Caller Interaction
4	0 0 1 0 0:	Routing Number
5	0 0 1 0 1:	Billing Number
6	0 0 1 1 0:	Destination Number
7	0 0 1 1 1:	Local Access and Transport Area (LATA)
8	0 1 0 0 0:	Carrier Identification
9	0 1 0 0 1:	Referral Number (Private)
10	0 1 0 1 0:	True Billing Number (Private)
11	0 1 0 1 1:	Alternate Preferred Carrier (Private)
12	0 1 1 0 0:	Preferred INC (Private)
13	0 1 1 0 1:	Primary Preferred Carrier (Private)
14	0 1 1 1 0:	Personal Identification Number (PIN)
15	0 1 1 1 1:	Authorization Code (Private)
16	1 0 0 0 0:	TCM (Private)
17	1 0 0 0 1:	Second Alternate Preferred Carrier (Private)
18	1 0 0 1 0:	Business Customer ID (Private)
19	1 0 0 1 1:	Hop-off Office (Private)
20	1 0 1 0 0:	Outpulse Number (Private)
21	1 0 1 0 1:	Originating Station (DN)
22	1 0 1 1 0:	MCCS Card Number
23	1 0 1 1 1:	Account Code Number
24	1 1 0 0 0:	COSOVE Number*
25	1 1 0 0 1:	Generic Digits Number
26	1 1 0 1 0:	Dialed Digits Number
27	1 1 0 1 1:	Facility Code
28	1 1 1 0 0:	Country Code
29	1 1 1 0 1:	STS Digits
30	1 1 1 1 0:	OPart Digits
31	1 1 1 1 1:	TPart Digits

- The NATURE OF NUMBER field consists of two bits and is encoded as follows:

0	0 0:	Not Applicable
1	0 1:	International
2	1 0:	National
3	1 1:	Network Specific

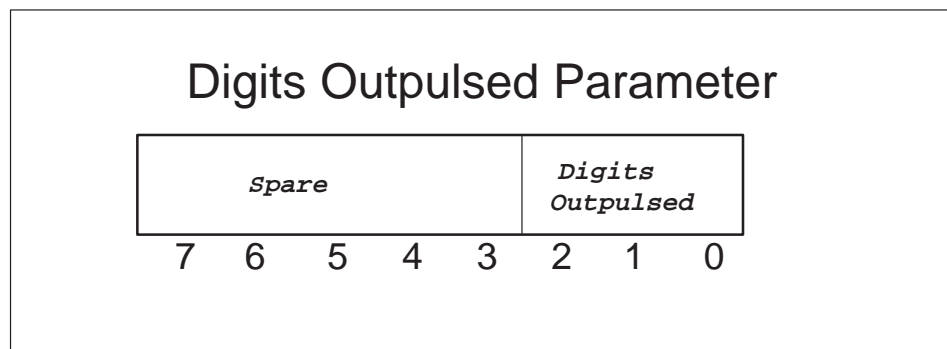
- The NUMBER OF DIGITS field is 6 bits and contains the number of digits that are sent in this parameter. This number may be as low as 0 and as high as 45.
- The ENCODING SCHEME field consists of four bits and contains the scheme used to encode the digits that are sent in this parameter. The following illustrates how the Encoding Scheme is encoded:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Binary Coded Decimal (BCD)
 - 0 0 1 0 to 1 1 0 1: Spare Values
 - 14 1 1 1 0: Telephony Binary Coded Decimal (TBCD)
 - 1 1 1 1: Spare
- The NUMBERING PLAN field consists of three bits. The following illustrates how the Numbering Plan is encoded:
 - 0 0 0 0: Unknown or Not Applicable
 - 1 0 0 1: ISDN Numbering Plan (E.164)
 - 2 0 1 0: Telephony Numbering Plan (E.163)
 - 3 0 1 1: Data Numbering Plan (X.121)
 - 4 1 0 0: Telex Numbering Plan (F.69)
 - 5 1 0 1: Maritime Mobile Numbering Plan (E.120, 211)
 - 6 1 1 0: Land Mobile Numbering Plan (E.212, 213)
 - 1 1 1 Spare Value
- The 1ST DIGIT to NTH DIGIT field consists of four bits per each digit and contains n digits. These DTMF digits may be encoded in the BCD format as follows:
 - 0 0 0 0: Digit 0
 - 1 0 0 1: Digit 1
 - 2 0 0 1 0: Digit 2
 - 3 0 0 1 1: Digit 3
 - 4 0 1 0 0: Digit 4
 - 5 0 1 0 1: Digit 5
 - 6 0 1 1 0: Digit 6
 - 7 0 1 1 1: Digit 7
 - 8 1 0 0 0: Digit 8
 - 9 1 0 0 1: Digit 9
 - 10 1 0 1 0: Filler
 - 11 1 0 1 1: *
 - 12 1 1 0 0: #
 - 13 1 1 0 1: D
 - 14 1 1 1 0: E
 - 15 1 1 1 1: F

Or the DTMF Digits may be encoded in the TBCD format as follows:

0 0 0 0 0: Filler
1 0 0 0 1: Digit 1
2 0 0 1 0: Digit 2
3 0 0 1 1: Digit 3
4 0 1 0 0: Digit 4
5 0 1 0 1: Digit 5
6 0 1 1 0: Digit 6
7 0 1 1 1: Digit 7
8 1 0 0 0: Digit 8
9 1 0 0 1: Digit 9
10 1 0 1 0: Digit 0
11 1 0 1 1: *
12 1 1 0 0: #
13 1 1 0 1: D
14 1 1 1 0: E
15 1 1 1 1: F

Digits outpulsed

This parameter indicates information on the digits that were outpulsed on a PTS trunk agent. This is especially useful in multi-stage outpulsing. This parameter is included in the **signaling** Event notification message to let the SCU know that all the digits were outpulsed on the trunk agency.



Optional parameter ID: 8

Parameter length: 1 byte

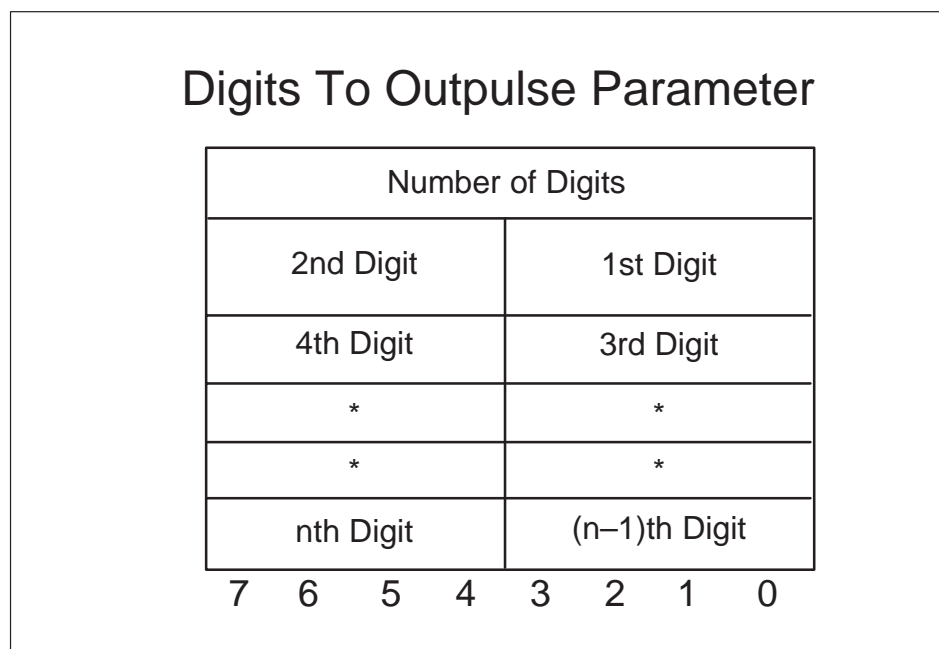
Parameter contents:

- The OUTPUTSED field consists of three bits and is encoded as follows:
 - 0 0 0: Unknown

- 0 0 1: All Streams Outputpulsed
- 0 1 0 to 1 1 1: Spare

Digits to outputpse

This parameter contains the digits to be outputpulsed on a PTS trunk agent.



Optional parameter ID: 9

Parameter length: Variable in size

Parameter contents:

- The NUMBER OF DIGITS consists of one byte and contains the number of digits that are to be outputpulsed on this port.

The maximum number of digits that may be outputpulsed is 23. The range for this field is from 1 to 23.

- The 1ST DIGIT to NTH DIGIT field consists of four bits per each digit and contains n digits.

DTMF digits are encoded as follows:

- 0 0 0 0: Filler
- 1 0 0 1: Digit 1

2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: *
12 1100: #
13 1101: D
14 1110: E
1111: F

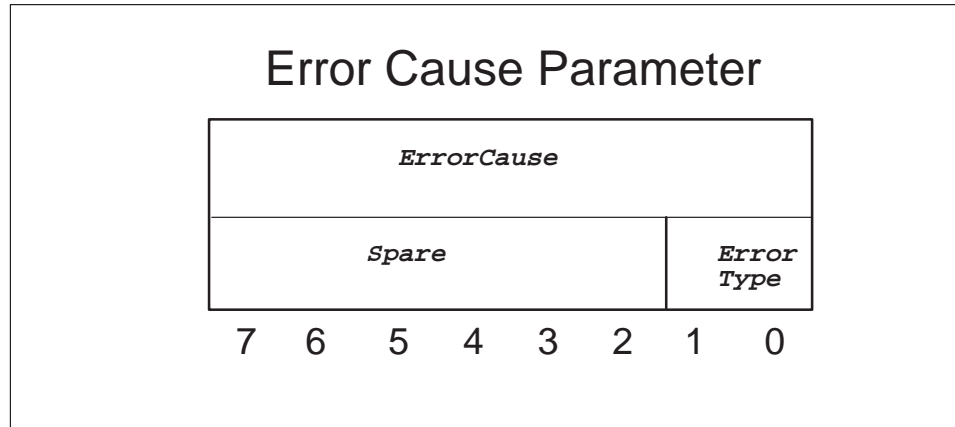
MF digits are encoded as follows:

0 0000: Filler
1 0001: Digit 1
2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: KP3 and ST3P
12 1100: KPP and STP
13 1101: KP and STKP
14 1110: KP2 and ST2P
1111: ST

If multiple “*DigitstoOutputpulse*” parameters are contained in a message, then multiple streams of digits are outputted on the agent or port with each stream contained in one parameter of type “*DigitstoOutputpulse*”.

Error cause

This parameter contains the cause of an error that is detected on the PSN.



Optional parameter ID: 10

Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of one byte and is used to represent the cause of the error. The values include:
 - 0 00000000: Nil Error Cause
 - 1 00000001: Header decode failure
 - 2 00000010: Bad macro tag
 - 3 00000011: Unrecognized primitive
 - 4 00000100: Missing mandatory parameter
 - 5 00000101: Mandatory parameter decode failure
 - 6 00000110: Optional parameter decode failure
 - 7 00000111: Parameter contents out of range
 - 8 00001000: Primitive userclass mismatch
 - 9 00001001: Maximum primitive exceeded
 - 10 00001010: Missing mandatory SigInfo parameter
 - 11 00001011: One or more agents in the primitive are not PSN agents
 - 12 00001100: Port not in table PSNROUTE
 - 13 00001101: Agent not supported
 - 14 00001110: Port down due to WARM restart
 - 15 00001111: Primitive invalid for current port state
 - 16 00010000: Unexpected message

17 00010001: STR not available (affects the **Monitor** primitive)
18 00010010: UTR not available
19 00010011: Conference circuit not available
20 00010100: No IDLE message
21 00010101: CCB not available
22 00010110: Primitive extension block not available
23 00010111: Scratchpad extension block not available
24 00011000: Software Resources unavailable
25 00011001: Message failure
26 00011010: Software error
27 00011011: Not minimum number ports to Bridge
28 00011100: Maximum ports to Bridge exceeded
29 00011101: Bearer Capability incompatible
30 00011110: Message index not in table PSNMSGIX
31 00011111: Unsupported signaling type
32 00100000: Duplicate message
33 00100001: Bad Agent state
34 00100010: Termination failure
35 00100011: Abnormal Exit
36 00100100: Message not playing
37 00100101: Tone duration unsupported
38 00100110: Prompt failure
39 00100111: Digit collection failure
40 00101000: Q764 protocol problem
41 00101001: Invalid duration gap
42 00101010: Unexpected FC message
43 00101011: Agent not in Table TRKGRP
44 00101100: BBF Not Possible

Note: The BBF Not Possible value is used to indicate that the BBF is not possible for this port based on its current agent/connection state. A port has to be connected to an agent and to be monitored for BBF. If it is held or bridged, BBF monitoring for the agent is not done.

- The ERROR TYPE field consists of two bits and contains the type of the error that is detected. The following illustrates how the Error Type field is encoded:
 - 00: Non Fatal Error
 - 01: Fatal Error

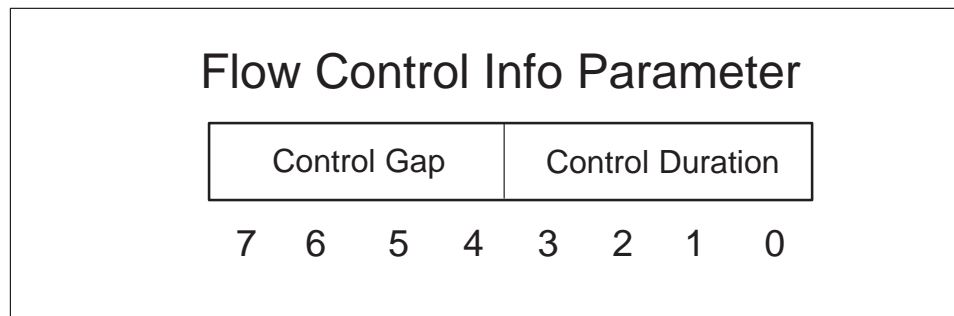
Fatal errors are defined as errors that are so severe that they do not allow normal call processing to proceed on this port.

If the error that is detected is non-fatal, then the PSN does not take any action other than report this error to the SCU. If the error that is detected is

fatal, then the PSN reports the error to the SCU. In addition, the PSN takes down the associated port when a fatal error is detected.

Flow control info parameter

This parameter consists of a control duration, which specifies the maximum amount of time the Flow Control Info is effective at the PSN. In addition, this parameter consists of a control gap, which specifies the maximum rate at which the **New_Call** event notifications may be sent to the SCU while the control is effective.



Optional parameter ID: 11

Parameter length: 1 byte

Parameter contents:

The CONTROL DURATION field consists of four bits and indicates the maximum amount of time that the Flow Control is effective at the PSN. The Control Duration is encoded as follows:

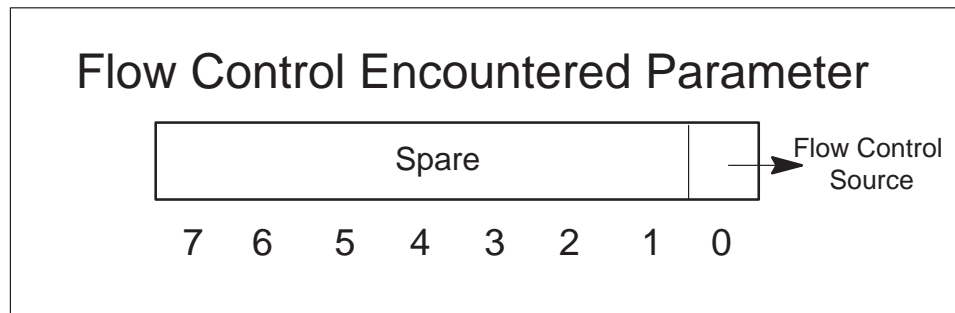
0	0 0 0 0:	Not Used
1	0 0 0 1:	1 Second
2	0 0 1 0:	2 Seconds
3	0 0 1 1:	4 Seconds
4	0 1 0 0:	8 Seconds
5	0 1 0 1:	16 Seconds
6	0 1 1 0:	32 Seconds
7	0 1 1 1:	64 Seconds
8	1 0 0 0:	128 Seconds
9	1 0 0 1:	256 Seconds
10	1 0 1 0:	512 Seconds
11	1 0 1 1:	1024 Seconds
12	1 1 0 0:	2048 Seconds
	1 1 0 1 – 1 1 1 1:	Spare Values

The CONTROL GAP field consists of four bits and indicates the maximum rate at which the **New_Call** event notifications may be sent to the SCU while the Flow Control is effective. The Control Gap is encoded as follows:

0	0 0 0 0:	Remove Gap Control
1	0 0 0 1:	1 1/10th of a Second
2	0 0 1 0:	3 1/10ths of a Second
3	0 0 1 1:	5 1/10ths of a Second
4	0 1 0 0:	1 Second
5	0 1 0 1:	2 Seconds
6	0 1 1 0:	5 Seconds
7	0 1 1 1:	10 Seconds
8	1 0 0 0:	15 Seconds
9	1 0 0 1:	30 Seconds
10	1 0 1 0:	50 Seconds
11	1 0 1 1:	80 Seconds
12	1 1 0 0:	120 Seconds
13	1 1 0 1:	300 Seconds
14	1 1 1 0:	600 Seconds
15	1 1 1 1:	Stop All Calls

Flow control encountered parameter

This parameter is sent within **New_Call** events, which are then sent to the SCU while the Flow Control is active. In addition, this parameter informs the SCU that the Flow Control is active and identifies the source that initiated the Flow Control.



Optional parameter ID: 12

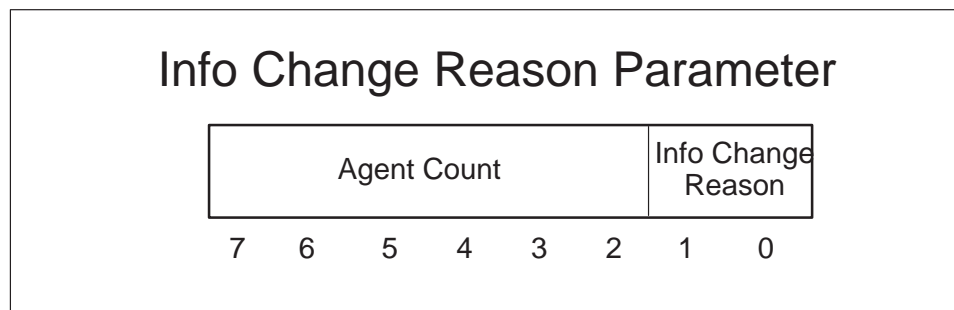
Parameter length: 1 byte

Parameter contents:

- The FLOW CONTROL SOURCE field consists of one bit and is the source that initiated the Flow Control. The Flow Control Source is encoded as follows:
 - 0: Flow Control initiated by the SCU
 - 1: Flow Control initiated by the PSN

Info change reason parameter

This parameter contains the Info Change Reason and Agent Count information used in **Agent_Data** event notifications.



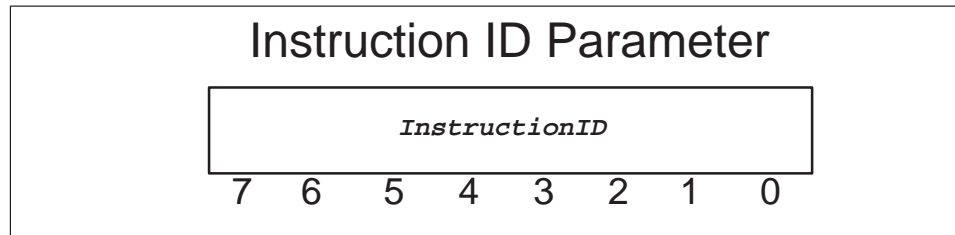
Parameter length: 1 byte

Parameter contents:

- The INFO CHANGE REASON field consists of two bits and indicates what type of **Agent_Data** event is occurring for the given message. The values include:
 - 0 0: Agent Deleted in table PSNROUTE
 - 0 1: Agent Added in table PSNROUTE
 - 1 0: Agent information Modified in table TRKGRP or TRKSGRP
- The AGENT COUNT field consists of six bits and indicates the number of **AgentDataInfo** parameters that exist in the given **Agent_Data** event message. The values include:
 - 0 0 0 0 1 – 1 1 1 1 1

Instruction ID

This parameter contains the Identifier of the primitive that is received from the SCU.



Optional parameter ID: 13

Parameter length: 1 byte

Parameter contents:

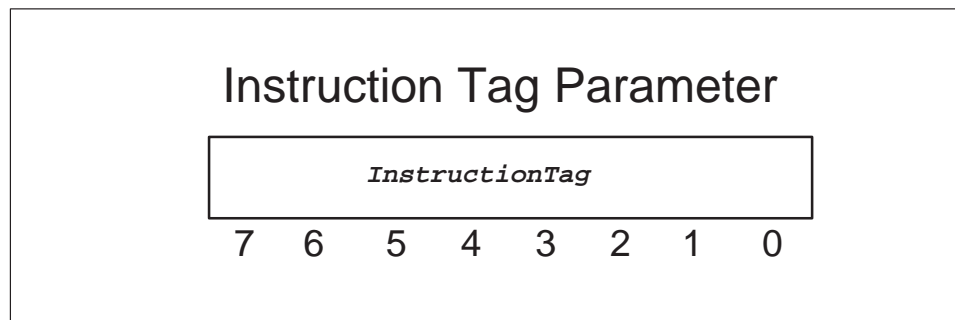
- The INSTRUCTION ID field consists of one byte which is used to represent the primitive instruction. The values include:

0	0 0 0 0 0 0 0 0	Unknown
1	0 0 0 0 0 0 0 1	Bridge
2	0 0 0 0 0 0 1 0	Collect Digits & Report
3	0 0 0 0 0 0 1 1	Connect
4	0 0 0 0 0 1 0 0	Disconnect
5	0 0 0 0 0 1 0 1	Hold
6	0 0 0 0 0 1 1 0	Monitor
7	0 0 0 0 0 1 1 1	Mute
8	0 0 0 0 1 0 0 0	New Call Accepted
9	0 0 0 0 1 0 0 1	New Call Rejected
10	0 0 0 0 1 0 1 0	Play Message
11	0 0 0 0 1 0 1 1	Play Prompt, Collect Digits & Report
12	0 0 0 0 1 1 0 0	Query Port
13	0 0 0 0 1 1 0 1	Reconnect
14	0 0 0 0 1 1 1 0	Reset Switch
15	0 0 0 0 1 1 1 1	Set Billing Record
16	0 0 0 1 0 0 0 0	Set IP Address
17	0 0 0 1 0 0 0 1	Stop Message
18	0 0 0 1 0 0 1 0	Transmit SigInfo
19	0 0 0 1 0 0 1 1	Heartbeat
20	0 0 0 1 0 1 0 0	Query Time of Day
21	0 0 0 1 0 1 0 1	Error Detected
22	0 0 0 1 0 1 1 0	Port Status
23	0 0 0 1 0 1 1 1	Flow Control
	0 0 0 1 1 0 0 0 – 1 1 1 1 1 1 1 1	Spare Values

Instruction tag

This parameter contains the tag associated with the instruction.

- For primitives that are sent from the SCU, the SCU generates an Instruction tag and then sends the Instruction tag to the PSN within this parameter. When a response for this primitive is generated by the PSN, the PSN includes this tag in the response, which enables the SCU to correlate the response with the primitive request that it had sent earlier.
- For asynchronous event notifications generated by the PSN, such as `On_Hook`, `Off_Hook` and `signaling` event, the Instruction tag is hard-coded to #01.
- For instructions that are generated by the PSN that require a reply from the SCU, such as New Call and Query Port from the Audit application, the PSN sends a nil Instruction tag.



Optional parameter ID: 14

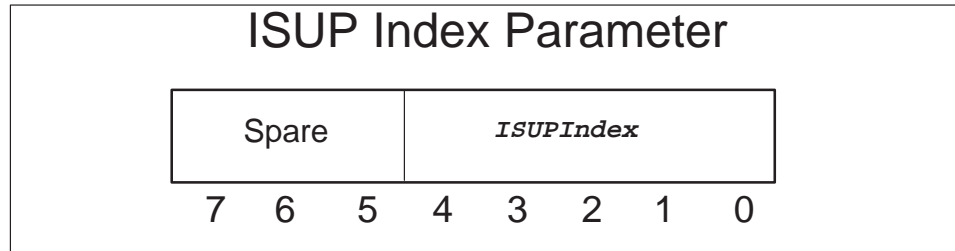
Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION TAG field consists of one byte, which is used to represent the tag of the instruction that is received from or sent to the SCU. The values include:
 - 0 0 0 0 0 0 0 0: Nil Instruction Tag
 - 0 0 0 0 0 0 0 1: Asynchronous Event Notification from the PSN.
 - 0 0 0 0 0 0 1 – 1 1 1 1 1 1 1 1: Valid Tags for Instructions from the SCU.

ISUP index

This parameter is used in the **Agent_Data** event and in the **New_Call** event.



Optional parameter ID: 31

Parameter length: 1 byte

Parameter contents:

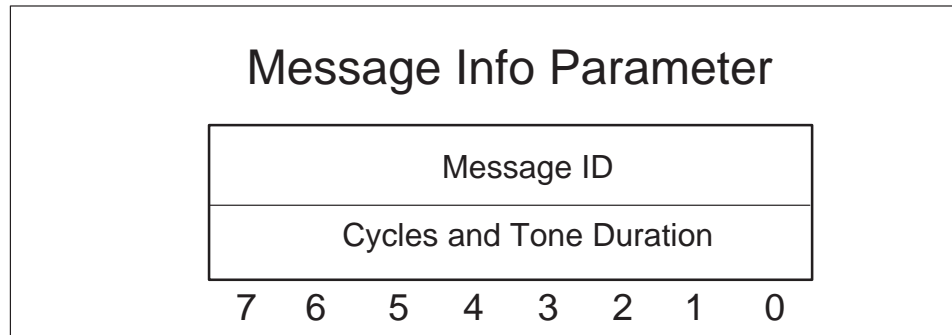
- The ISUP INDEX field consists of five bits and represents the type of facility that the SS7 agent connects to. Please note that this field is only valid for SS7 signaling types. The values include:

0 0 0 0 0: NILIDX
1 0 0 0 1: UCS2UCS
2 0 0 0 1 0: UCS2DEX8
3 0 0 0 1 1: UCS2EAE0
4 0 0 1 0 0: UCS2USP
5 0 0 1 0 1: UCS2MCI
6 0 0 1 1 0: UCSGITU
7 0 0 1 1 1: USP2USP
8 0 1 0 0 0: USP2UCS
9 0 1 0 0 1: UCSGWAY
0 1 0 1 0 – 1 1 1 1 1: Unused

Message info

This parameter contains the Message ID and the Cycles and Tone Duration information.

The Message ID is used to index into the new table PSNMSGIX to get either an index into the table ANNS (if the message ID corresponds to an announcement) or into the table TONES (if the message ID corresponds to a tone).



Optional parameter ID: 15

Parameter length: 2 bytes

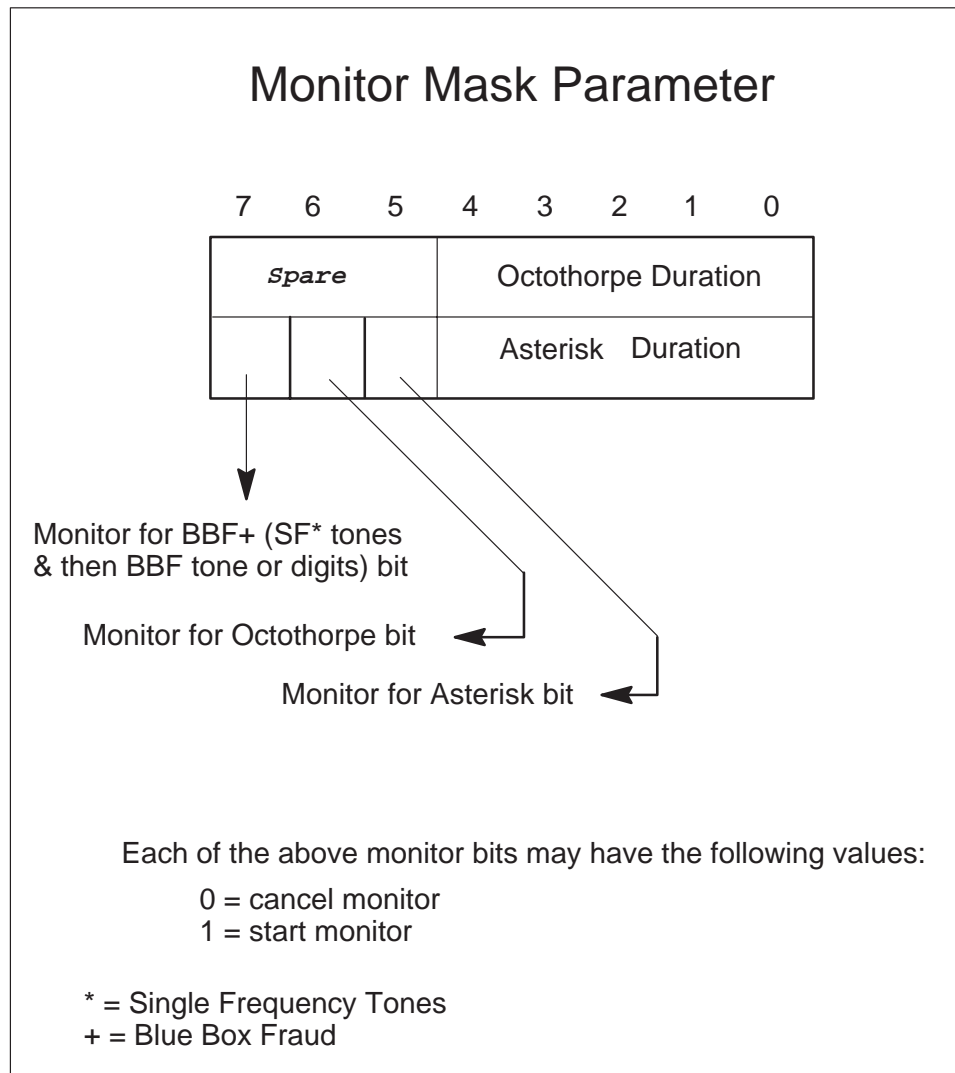
Parameter contents:

- The MESSAGE ID field consists of one byte with values of 1 to 255. The values are an index into the table PSNMSGIX. For the Message ID field, a value of 0 is invalid.
- The CYCLES field consists of one byte with a second byte of the parameter contents treated as the number of cycles in which to play the announcement, if the message ID field corresponds to an announcement. The values include:
 - 0 0 0 0 0 0 0 0: Play the Announcement indefinitely
 - 0 0 0 0 0 0 0 1: Play the Announcement for 1 cycle
 - 0 0 0 1 1 1 1 0: Play the Announcement for 30 cycles
 - 0 0 0 1 1 1 1 1 – 1 1 1 1 1 1 1 1: Play the Announcement indefinitely
- The TONE DURATION field consists of one byte with a second byte of the parameter contents treated as the tone duration value, if the message ID corresponds to a tone. The values include:
 - 0 0 0 0 0 0 0 0: Play the Tone forever
 - 0 0 0 0 0 0 0 1: Invalid
 - 0 0 0 0 0 0 1 0: Invalid
 - 0 0 0 0 0 0 1 1: Play the Tone for 3 seconds
 - 0 0 0 0 0 1 0 0: Play the Tone for 4 seconds
 - 1 1 1 1 1 1 1 1: Play the Tone for 255 seconds

Note: The values for the cycles or tone duration may be set to any value when the *MessageInfo* parameter is sent in the *Stop_Message* primitive. The *Stop_Message* primitive is only concerned with the Message ID (to stop the appropriate message on the PSN) in the *MessageInfo* parameter.

Monitor mask

This parameter is a bitmap of monitor values. Each bit in this bitmap indicates what digit or tone to monitor, or not to monitor, for a port. Also, each bit has two values: 0, which indicates to cancel the monitor, or 1, which indicates to start the monitor.



Optional parameter ID: 16

Parameter length: 2 bytes

Parameter contents:

- The **ASTERISK** and **OCTOTHORPE DURATION** fields consist of five bits each with the duration in 100 milliseconds for which the asterisk or octothorpe must be detected before it is reported as a valid tone or digit to the SCU. The values range from 5 to 30.

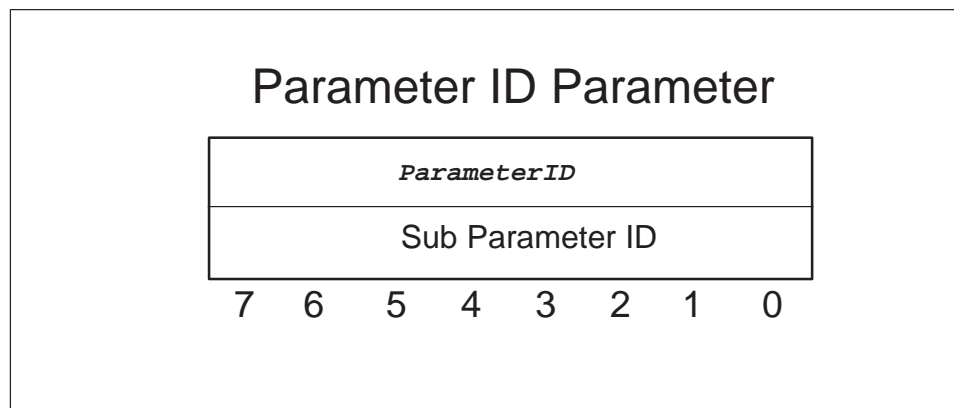
For example a duration of 5 for an Asterisk implies that the user has to hold down the asterisk for 500 msec before it is detected as a valid asterisk by the PSN and reported to the SCU.

Note: For **BBF**: the Single Frequency Tone duration, the signaling type (MF/DTMF), the Minimum Digits to Collect before a Blue Box Fraud is declared, and the Partial Dial Timer values are all determined from the table **BBTKSGRP**. Therefore, it is necessary to datafill the “terminating” port (where “terminating” indicates Party B of a **Connect** or a **Reconnect** primitive) in the table **BBTKSGRP**.

- The **TONE BITMAP** field consists of three bits and is a boolean for **BBF**, **Octothorpe**, and **Asterisk**. If the bool is true, then the port is monitored for the appropriate tone or digit. If the bool is false, then the appropriate tone or digit being monitored on that port is cancelled.

Parameter ID

This parameter contains the Identifier of the parameter, including the Identifier of the sub-parameter (in the event that the parameter is composed of sub parameters). The Identifier is returned to the SCU in case the decoding of the parameter resulted in an error.



Optional parameter ID: 17

Parameter length: 2 bytes

Parameter contents:

- The PARAMETER ID field consists of one byte, which is used to represent the parameter in error. The values include:
 - 0 00000000: Nil Error Cause
 - 1 00000001: Bearer Capability
 - 2 00000010: Billing Info
 - 3 00000011: Call Reference Identifier
 - 4 00000100: Control Info
 - 5 00000101: Destination Trunk Group
 - 6 00000110: Digit Collection
 - 7 00000111: Digits Collected
 - 8 00001000: Digits Outpulsed
 - 9 00001001: Digits To Outpulse
 - 10 00001010: Error Cause
 - 11 00001011: Flow Control Information
 - 12 00001100: Flow Control Encountered
 - 13 00001101: Instruction ID
 - 14 00001110: Instruction Tag
 - 15 00001111: Message Info
 - 16 00010000: Monitor Mask
 - 17 00010001: Parameter ID
 - 18 00010010: Port Count
 - 19 00010011: Port Info
 - 20 00010100: Port Service Information
 - 21 00010101: Port Status
 - 22 00010110: Reset Reason
 - 23 00010111: Session ID
 - 24 00011000: SigInfo Mask
 - 25 00011001: Signaling Info
 - 26 00011010: Switch ID
 - 27 00011011: Time of Day
 - 28 00011100: Tone Detected
 - 00011101 – 10000001: Spare Values
 - 0 10000010: UCS Point In Call
 - 1 10000011: UCS STS
 - 10000100 – 11111111: Spare Values
- The SUB PARAMETER ID field consists of one byte, which is used to represent the parameter in error.

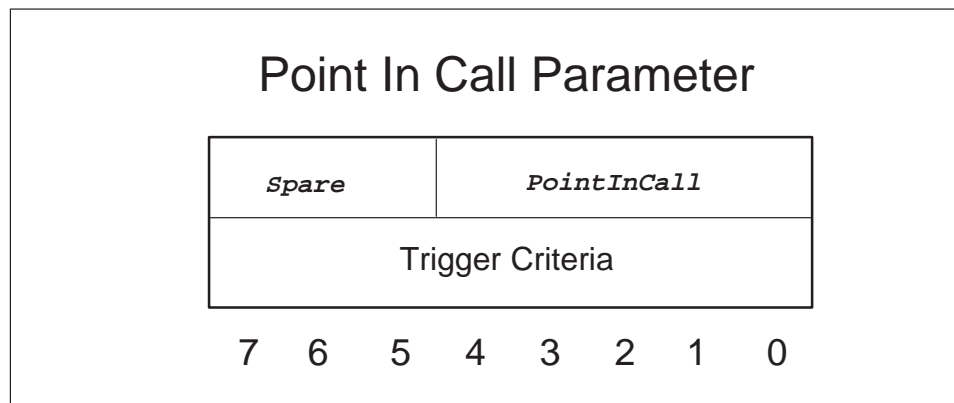
The values for PTS agents include:

 - 0 00000000: Unknown
 - 1 00000001: Message Type

2 00000010: Digits Outputed
 3 00000011: Digits to Outputed
 4 00000100: PTS Off-hook
 5 00000101: PTS On-hook
 00000110–11111111: Spare

Point in call

This parameter contains the Point in Call when all criteria are met and the PSN gives the SCU control over the call.



Optional parameter: 130

Parameter length: 2 byte

Parameter contents:

- The POINT IN CALL field consists of five bits and contains the Point in Call where the PSN determines that the call is a service call and should be controlled by the SCU. The Point in Call is encoded as follows:

0 00000: Not Used
 1 00001: Orig Null
 2 00010: Authorize Orig Attempt
 3 00011: Collect Information
 4 00100: Analyze Information*
 5 00101: Select Route
 6 00110: Authorize Call Setup
 7 00111: Send Call
 8 01000: Orig Alerting
 9 01001: Orig Active
 10 01010: Orig Suspended
 11 01011: Term Null

12 0 1 1 0 0: Authorize Termination
13 0 1 1 0 1: Select Facility
14 0 1 1 1 0: Present Call
15 0 1 1 1 1: Term Alerting
16 1 0 0 0 0: Term Active
17 1 0 0 0 1: Term Suspended
1 0 0 1 0 – 1 1 1 1 1: Spare Values

The values marked with an “*” are the only ones that are currently supported.

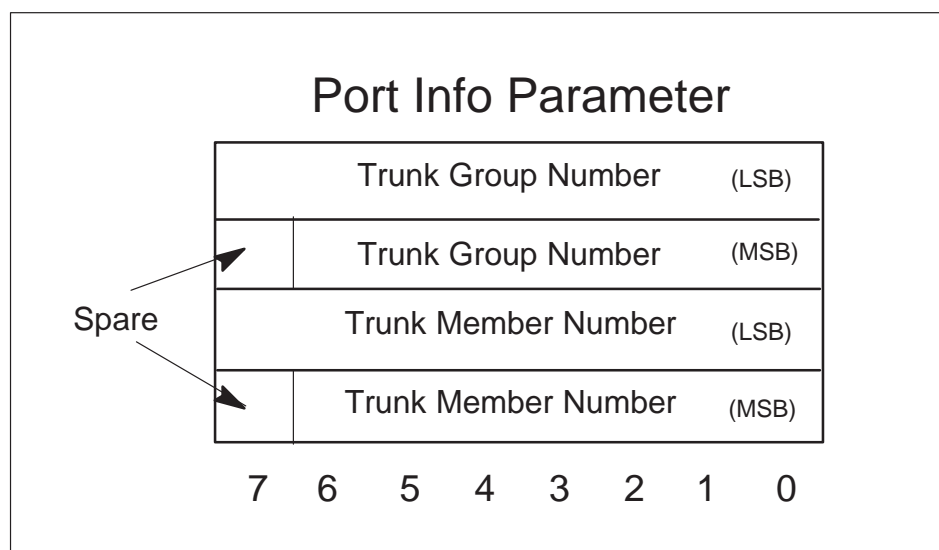
- The TRIGGER CRITERIA field consists of eight bits and is encoded as follows:

0 0 0 0 0 0 0 0: Feature Activator
1 0 0 0 0 0 0 1: Vertical Service Code
2 0 0 0 0 0 1 0: Customized Access
3 0 0 0 0 0 1 1: Customized Intercom*
4 0 0 0 0 1 0 0: NPA
5 0 0 0 0 1 0 1: NPA_NXX
6 0 0 0 0 1 1 0: NXX
7 0 0 0 0 1 1 1: NXX_XXXX
8 0 0 0 1 0 0 0: NPA_NXXXXXX*
9 0 0 0 1 0 0 1: Country_Code_NPA_NXXXXXX*
10 0 0 1 0 0 0 0: Off-hook Immed
11 0 0 1 1 0 0 0: Net Busy
12 0 0 1 1 0 1 1: Orig Called Party Busy
13 0 0 1 1 1 0 1: Orig No Answer
14 0 0 1 0 0 0 0: Orig Feature Activator
15 0 1 1 0 0 0 0: Channel Setup PRI CLID
16 0 1 1 0 0 0 1: Channel Setup PRI Addr
17 0 1 1 0 0 1 0: Channel Setup PRI N00
18 0 1 1 0 0 1 1: Channel Setup PRI Intl
19 0 1 1 0 1 0 0: Specific Digit String Info*
20 0 1 1 0 1 0 1: Specific Digit String ANI*
21 0 1 1 0 1 1 0: Specific Digit String N00*
22 0 1 1 0 1 1 1: Specific Digit String CIC
23 0 1 1 0 1 0 0 0: Shared Interoffice CIC
24 0 1 1 0 1 0 0 1: Shared Interoffice Info
25 0 1 1 0 1 0 1 0: Shared Interoffice ANI
26 0 1 1 0 1 0 1 1: Shared Interoffice Addr
27 0 1 1 0 1 1 0 0: Shared Interoffice N00
28 0 1 1 0 1 1 0 1: Shared Interoffice Intl
0 1 1 0 1 1 1 0 – 1 1 1 1 1 1 1 1: Unused (Spare)

The values marked with an “*” are the only ones that are currently supported.

Port info

This parameter contains the port information for an agent. The Port Info consists of the external Trunk Group Number and the Trunk Member Number.



Optional parameter ID: 19

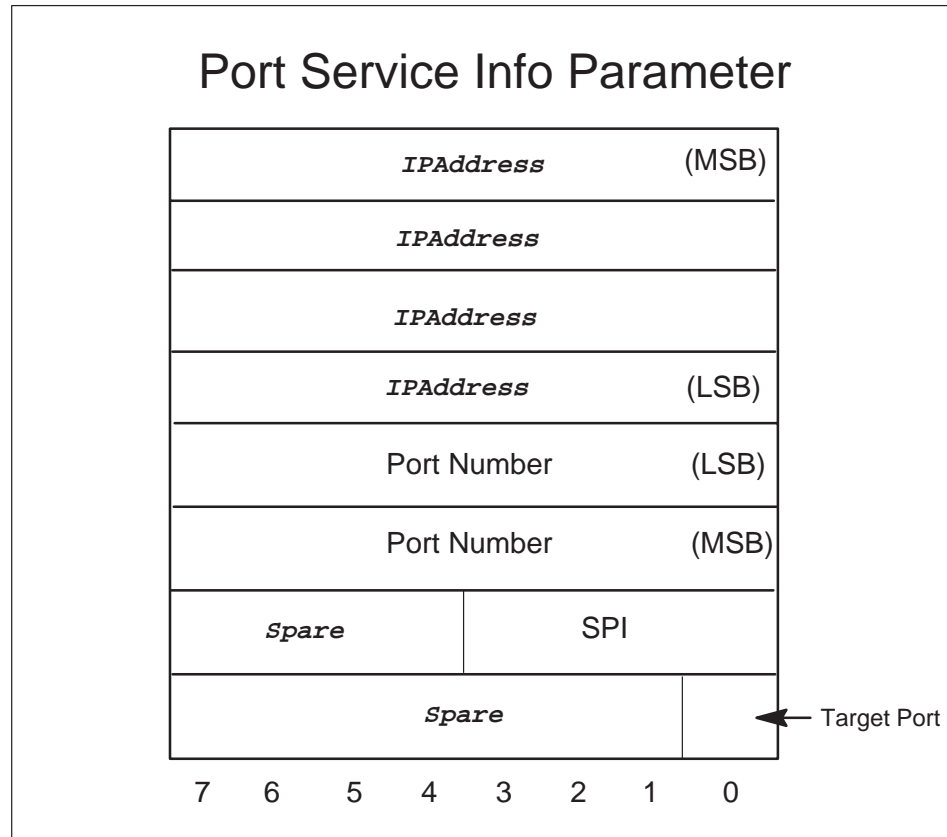
Parameter length: 4 Bytes

Parameter contents:

- The TRUNK GROUP NUMBER field consists of 15 bits and is the external Trunk Group Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_GROUP.
- The TRUNK MEMBER NUMBER field consists of 15 bits and contains the Trunk Member Number. The range is from 0 to 9999. A value of 32,767 indicates a NIL_TRUNK_MEMBER.

Port service info

This parameter contains the IP Address and the Service Programming Interface (SPI) information. The SCU returns the IP Address and the SPI to the PSN. Please note that the IP Address is the return address used to send the event notification messages back to the SCU (to the correct address and port). In addition, the SPI number is the version for the specified agent.



Optional parameter ID: 20

Parameter length: 8 bytes

Parameter contents:

- The IP ADDRESS field consists of four bytes and contains the IP Address sent by the SCU, which is eventually used to return the event notification messages.

For example, an IP Address of 47.122.64.153 stored as four bytes where byte 1 is 47, byte 2 is 122, byte 3 is 64, and byte 4 is 153. The `nil_ip_address` is 0.0.0.0.

- The PORT NUMBER field consists of two bytes and contains the transport layer Port Number associated with the IP Address. The nil_port_number is 0.

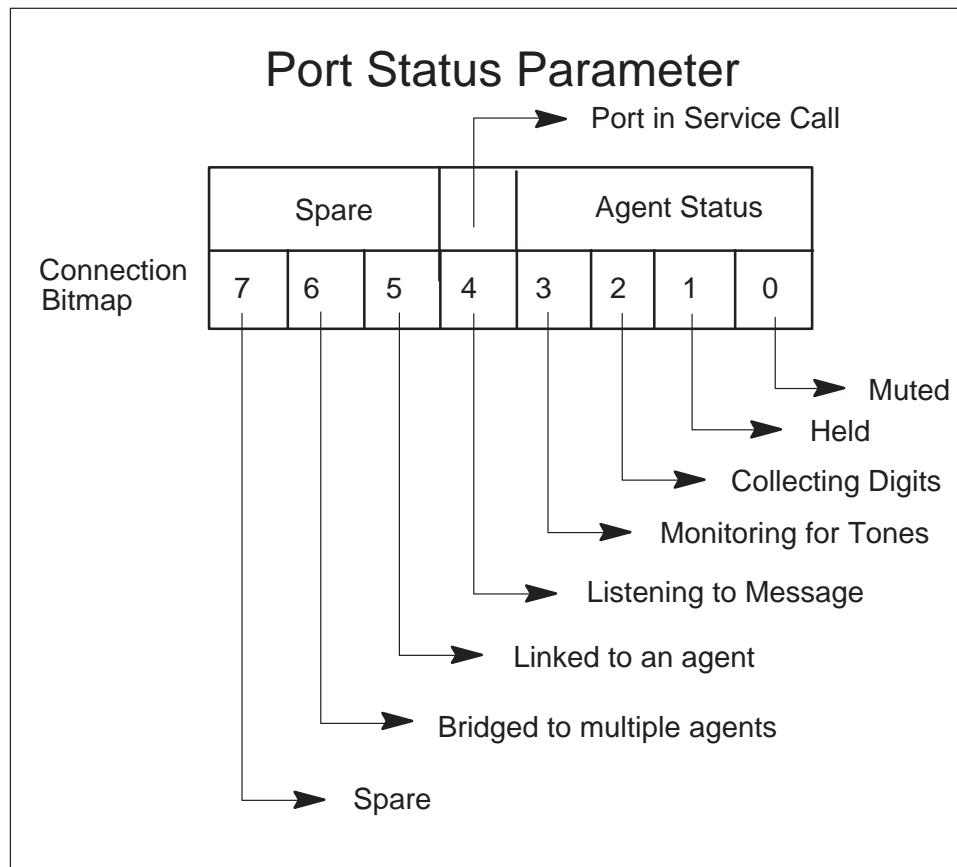
The following table illustrates how to interpret the different primitives for combination of the *IPAddress* and the Port Number in the *IPAddressInfo* parameter.

IP Address	Port Number	How is the IP Address Info used for in RESET SWITCH	How is the IP Address Info used in all the other Primitives
nil_ip_address	nil_port_number	Idle all ports that are currently serviced by the SCU (System Reset)	INVALID
NON nil_ip_address	nil_port_number	Idle all ports with the given IP Address and any port number (Shelf Reset)	INVALID
nil_ip_address	Non nil_port_number	INVALID	INVALID
Non nil_ip_address	Non nil_port_number	Idle ports with the appropriate IP Address Info (Service Reset)	<i>IPAddressInfo</i> parameter applies to a specific port – update the port's IP Address Info.

- The SPI field consists of four bits and contains the SPI version for the specified agent. The range is from 1 to 15.
- The TARGET PORT field is one byte and indicates to what port a primitive is being sent. This is necessary since some primitives (for example Connect) involve more than one port in a call.
 - 0 0 : Port_A This is the default value
 - 1 1 : Port_B Valid only when used with the Connect and/or Reconnect
 - 00000010 to 11111111 : Spare Values

Port status

This parameter contains the status of the port or agent as well as information which indicates if the port is currently controlled by the SCU.



Optional parameter ID: 21

Parameter length: 2 bytes

Parameter contents:

- The AGENT STATUS field consists of four bits and contains the Agent Status of the port, as illustrated below:

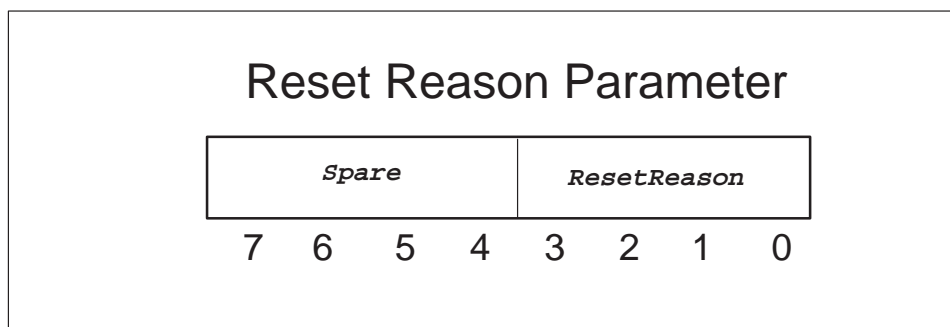
0 0 0 0 0: Idle
 1 0 0 0 1: Seized
 2 0 0 1 0: Answered
 3 0 0 1 1: ManBusy

4 0 0 1 0 0: Lockout
 5 0 0 1 0 1: System Busy
 6 0 0 1 1 0: PM Busy
 7 0 0 1 1 1: Unknown
 1 0 0 0 to 1 1 1 1: Spare Values

- The PORT IN SERVICE CALL field consists of one bit. The purpose of this field is to indicate if the port is currently a part of a service call or if the port is currently being serviced by the SCU.
 - 0: Port not a part of a service call
 - 1: Port is a part of a service call.
- The CONNECTION BITMAP field consists of six bits and is a bitmap, where each bit indicates the connection status of the agent. The bits are used to represent the connection states as specified below:
 - Bit 0: Muted
 - Values: 0 = port is unmuted
 - 1 = port is muted
 - Bit 1: Held
 - Values: 0 = port is not held
 - 1 = port is held
 - Bit 2: Collecting Digits
 - Values: 0 = port is not collecting digits
 - 1 = port is in the process of collecting digits
 - Bit 3: Monitoring for Tones
 - Values: 0 = port is not monitoring for tones
 - 1 = port is monitoring for tones
 - Bit 4: Listening to Message
 - Values: 0 = port is not listening to a message
 - 1 = port is listening to message (announcement and tones)
 - Bit 5: Linked to an Agent
 - Values: 0 = port is not linked to another agent
 - 1 = port is linked to another agent
 - Bit 6: Bridged to multiple agents
 - Values: 0 = port is not bridged to two or more agents
 - 1 = port is bridged to two or more agents

Reset reason

This parameter contains the reason why a restart occurred on the PSN. In addition, the Reset Reason indicates (to the SCU) the type of reset that is required at the SCU.



Optional parameter ID: 22

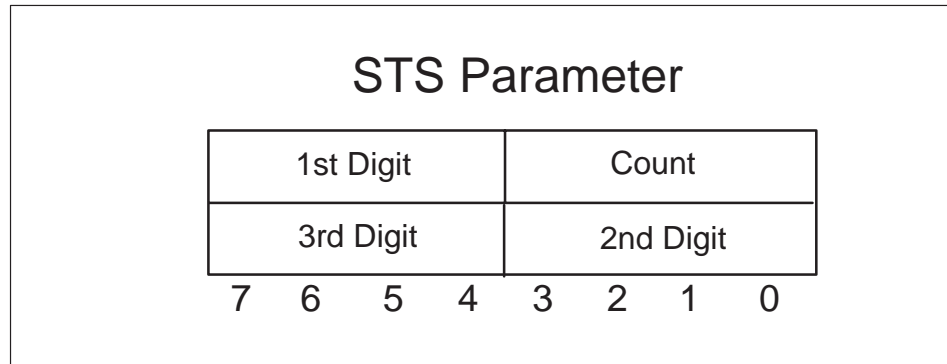
Parameter length: 1 byte

Parameter contents:

- The RESET REASON field consists of four bits and indicates the reason a restart occurred on the PSN. The values include:
 - 0 0 0 0: Unknown
 - 1 0 0 1: Warm Restart performed on the PSN
 - 2 0 0 1 0: Cold Restart performed on the PSN
 - 3 0 0 1 1: Reload Restart performed on the PSN
 - 4 0 1 0 0: PSN has just come into service
 - 5 0 1 0 1: Arbitrator Heartbeat has failed
 - 0 1 1 0 – 1 1 1 1: Spare Values

Serving translation scheme

This parameter contains the Serving Translation Scheme (STS) of the call. The switch sends this parameter to the SCU in the `New_Call` event notification message.



Optional parameter: 131

Parameter length: 2 bytes

Parameter contents:

- The COUNT field consists of four bits and contains the number of digits in the STS. The values include 1 to 3.
- The DIGITS 1, 2, and 3 field consists of four bits for each digit and contains each digit (1,2, and 3) in the TBCD format, which is encoded as follows:

```

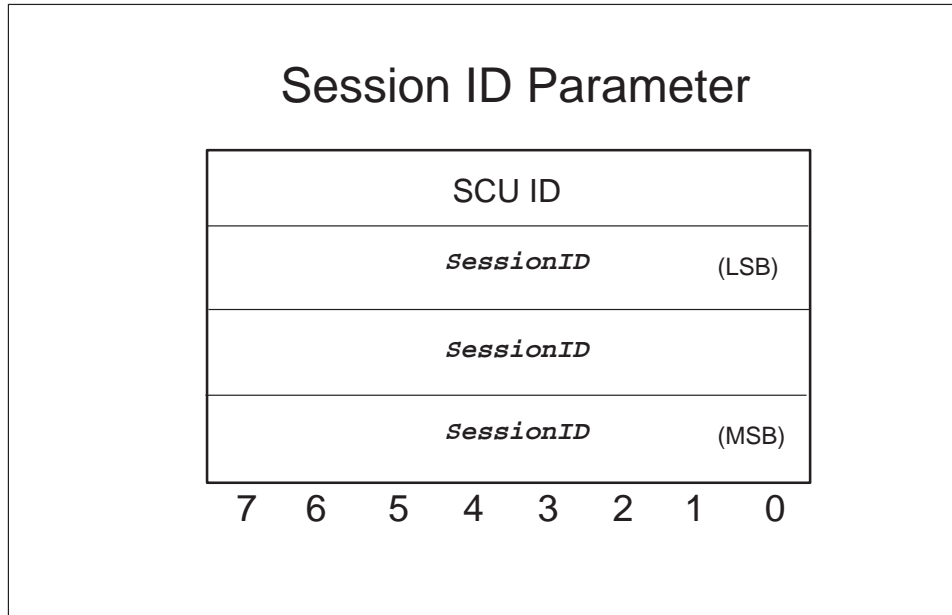
0 0000: Filler
1 0001: Digit 1
2 0010: Digit 2
3 0011: Digit 3
4 0100: Digit 4
5 0101: Digit 5
6 0110: Digit 6
7 0111: Digit 7
8 1000: Digit 8
9 1001: Digit 9
10 1010: Digit 0
11 1011: *
12 1100: #
13 1101: D
14 1110: E
15 1111: F

```

Session ID

This parameter contains the Session ID. The Session ID is generated by the SCU to associate the different ports involved in a service call.

Upon receipt of the *sessionID* parameter, the PSN does not perform any error checking. It merely re-transmits this value in the event notification message.



Optional parameter ID: 23

Parameter length: 4 Bytes

Parameter contents:

- The SCU ID field consists of one byte. The SCU ID field is a unique one byte ID generated by the SCU.
- The SESSION ID field consists of three bytes. The Session ID field is a unique 3 byte ID generated by the SCU. The values include:

000000000000000000000000: 0

to

111111111111111111111111111111: 16777215

Signaling info

This parameter contains the Signaling Information in the standard format that is applicable to the signaling type of the port. This parameter may be used to receive or send signaling information from or to the SCU.

Depending upon the primitive or event notification in which this parameter is sent or received, and the signaling type of the associated port, its contents may differ. In general, the contents contain parameters that are encoded in the standard format.

For PTS agents, there are no standards used. For SS7 agents, the parameters are encoded based on the TR444 and GR394, and for PRI agents, refer to Chapter UCS PRI, for more details.

Please refer to the primitives and event notification messages below for information on Signaling Info.

- **Connect** primitive from the SCU (receive use):

- The parameters below are sent in Signaling Info (on a PTS agent). At least one stream of digits (a *DigitstoOutputpulse* parameter) must be sent, with the capability of sending up to three streams. These parameters are encoded as described earlier in “Bearer capability” and “Digits to outputpulse.”

Bearer Capability	(Optional)
Digits to Outputpulse	(Mandatory)
Digits to Outputpulse	(Optional)
Digits to Outputpulse	(Optional)

- The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)
Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)
RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)
XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)

Originating Line Information	(Optional)
Supplementary Line Info	(Optional)
Transit Network Selection	(Optional)

- The following SETUP message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **Disconnect** primitive from the SCU (receive use):

- The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR39.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)

- The following DISC message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)

- The following REL message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- **Transmit_Siginfo** primitive from the SCU (receive use):
 - The following parameters are sent in Signaling Info (on a PTS agent). At least one stream of digits (a *DigitstoOutpulse* parameter) must be sent with the capability of sending up to three streams. The signaling parameters are encoded as described earlier in “Digits To Outpulse”.

Message Type	(Mandatory)
Digits to Outpulse	(Mandatory)
Digits to Outpulse	(Optional)
Digits to Outpulse	(Optional)

- The following information is sent (on a PTS agent) for a PTS On-hook. There is no Signaling Information present when the message type is PTS On-hook.

Message Type	(Mandatory)
--------------	-------------

- The following information is sent (on a PTS agent) for a PTS Off-hook. There is no Signaling Information present when the message type is PTS Off-hook.

Message Type	(Mandatory)
--------------	-------------

The Transmit SigInfo is used to send the IAM, ANM, CPG, FAA, FAR, FRJ, PAM, SUS, REL, and RES for an SS7 port.

The Transmit SigInfo is used to send the CONNECT, CALLPROC, FACILITY, RLC, REL, and DISC messages for a PRI port.

For the definition of the above SS7 and PRI messages, please refer to **signaling_Event** bullet in this section.

- **off_Hook** event notification to the SCU (send use):
 - The following ANM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Optional)
Call Reference	(Optional)
Carrier Selection	(Optional)
Internetwork Specific ANM	(Optional)
Intranetwork Specific ANM	(Optional)
Network Specific ANM	(Optional)
Operator Information	(Optional)
US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following CONNECT message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User to User Information	(Optional)

- **On_Hook** event notification to the SCU (send use):

- The following REL message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Cause Indicators	(Mandatory)
User to User Indicator	(Optional)
User to User Information	(Optional)

- The following RSC message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
--------------	-------------

- the following DISC message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Mandatory)
User to User Information	(Optional)

- the following REL message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- **New_Call** event notification to the SCU (send use):

- The following IAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Forward Call Indicator	(Mandatory)
Calling Party Category	(Mandatory)
User Service Information	(Mandatory)
Called Party Number	(Mandatory)
Nature of Connection Ind.	(Mandatory)
Calling Party Address	(Optional)
Carrier Identification	(Optional)

Carrier Selection Information	(Optional)
Channel Assignment Map	(Optional)
Charge Number	(Optional)
Authcode (GD)	(Optional)
Call Reference Identifier (GD)	(Optional)
Callid (GD)	(Optional)
CLLI Admin (GD)	(Optional)
IMT Info (GD)	(Optional)
RLT Treatment Code (GD)	(Optional)
Term SWID & TRKGRP (GD)	(Optional)
XFR Operator Queue (GD)	(Optional)
Network Information	(Optional)
Network Specific Facilities	(Optional)
Network Specific IAM	(Optional)
Operator Information	(Optional)
Operator Services Indicator	(Optional)
Originating Line Information	(Optional)
Supplementary Line Info	(Optional)
Transit Network Selection	(Optional)

- The following SETUP message parameters (on a PRI agent) are encoded as reported in the Chapter, “PRI messages”.

Protocol Discriminator	(Mandatory)
Message Type	(Mandatory)
Bearer Capability	(Mandatory)
Called Party Number	(Mandatory)
Business Group	(Optional)
Called Party Subaddress	(Optional)
Progress Indicator	(Optional)
Calling Party Number	(Optional)
Calling Party Subaddress	(Optional)
Display	(Optional)
Higher Layer Compatibility	(Optional)
Lower layer Compatibility	(Optional)
Network Specific Facility	(Optional)
Original Called Number	(Optional)
Transit Network Selection	(Optional)
User to User Information	(Optional)

- **signaling_Event** event notification to the SCU (send use):

- The following parameters are sent in Signaling Info (on a PTS agent). These parameters are encoded as described earlier in “Digits outpulsed.”

Digits Outpulsed Indicator	(Mandatory)
----------------------------	-------------

- The following ACM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Backward Call Indicator	(Mandatory)
Call Reference	(Optional)

US Network Parameter	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

— The following COT message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Continuity Indicators	(Mandatory)

— The following CPG message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Event Information	(Mandatory)
Backward Call Indicator	(Mandatory)
Party Information	(Optional)
Redirection Number	(Optional)
Supply end-to-end Inf. Req.	(Optional)
Supply end-to-end Inf. Resp.	(Optional)
User to User Indicator	(Optional)
User to User Information	(Optional)

— The following FAA message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)

— The following FAR message parameters (on an SS7 agent) are encoded as per the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)
Call Reference	(Optional)
Called Party Address	(Optional)
Calling Party Number	(Optional)
Charge Adjustment	(Optional)
Charge Number	(Optional)
Callid (GD)	(Optional)
Operator Information	(Optional)
Originating Line Info	(Optional)

— The following FRJ message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Facility Indicator	(Mandatory)

Cause Indicator	(Mandatory)
Call Reference	(Optional)

- The following PAM message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
RPM Message	(Mandatory)

- The following RES message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend and Resume Indicators	(Mandatory)

- The following SUS message parameters (on an SS7 agent) are encoded as indicated by the TR444 and the GR394.

Message Type	(Mandatory)
Suspend and Resume Indicators	(Mandatory)

- The following PROGRESS message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
Progress Indicator	(Mandatory)
Cause	(Optional)
User to User Information	(Optional)

- The following ALERT message parameters (on a PRI agent) are encoded as indicated in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)
User–User Information	(Optional)
Progress Indicator	(Optional)

- The following CALL PROCEEDING message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)

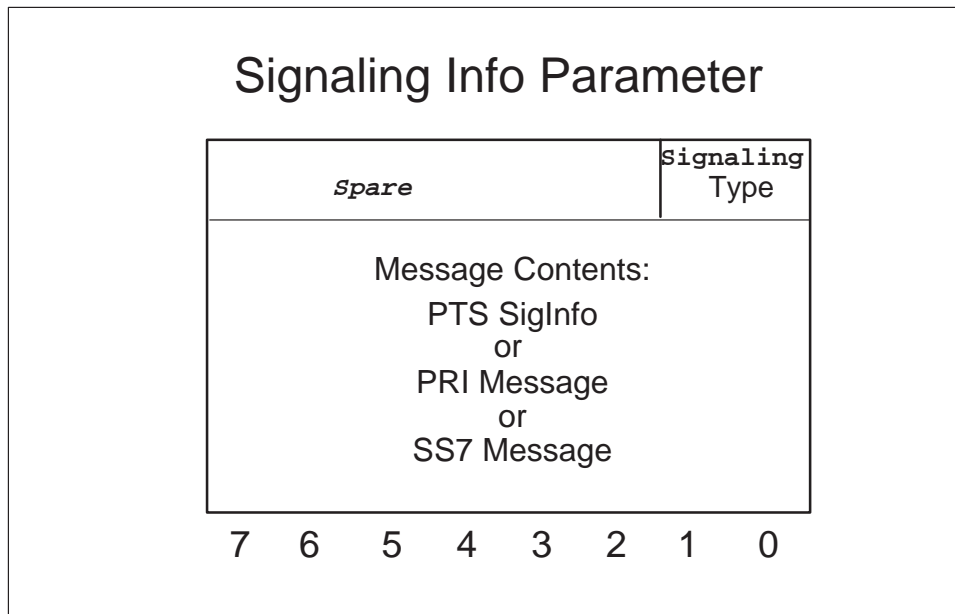
- The following FACILITY message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type	(Mandatory)
Protocol Discriminator	(Mandatory)

Called Party Number (Optional)
 Called Party Subaddress (Optional)

— The following REL Complete (RLC) message parameters (on a PRI agent) are encoded as recorded in the Chapter, “PRI messages”.

Message Type (Mandatory)
 User to User Information (Optional)



Optional parameter ID: 25

Parameter length: maximum of 412 bytes

Parameter contents:

- The Signaling TYPE field consists of two bits and is the field which indicates the Signaling TYPE.
- The MESSAGE CONTENTS field consists of a maximum of 411 bytes. In addition, this field possesses the message contents, which contain the parameters in the standard format listed earlier. The contents are encoded, depending upon the type of signaling used.

Primitive / Event Notification	PTS Messages	SS7 Messages	PRI Messages
Connect	Digits_To_Out pulse, Digits_To_Out pulse with Bearer_Capability	IAM	SETUP
Disconnect	Bearer_Capability	REL	DISC, REL
Transmit_Signaling_Event	Digits_To_Out pulse PTS_On-hook PTS_Off-hook	ACM, IAM, ANM, REL, CPG, FAA, FAR, FRJ, PAM, RES, and SUS	CONNECT, REL, DISC, ALERT, and FACility
New_Call		IAM	SETUP
Off_Hook		ANM	CONNECT
On_Hook		REL and RSC	DISC and REL
Signaling_Event	Digits_Outpulsed_Indicator	ACM, CPG, COT, FAA, FAR, FRJ, PAM, RES, and SUS	PROGRESS, ALERT, CALL PROCEEDING, FACILITY, RELEASE COMPLETE
—end—			

— PTS Message Contents:

- The PTS MESSAGE TYPE field consists of one byte and is the Message Type for a PTS message. The values include:
 - 0 00000000: Unknown
 - 1 00000001: Digits to Output
 - 2 00000010: Digits to Output with Bearer Capability
 - 3 00000011: PTS Off-Hook
 - 4 00000100: PTS On-Hook
 - 5 00000101: Digits Outpulsed
 - 00000110–11111111: Spare

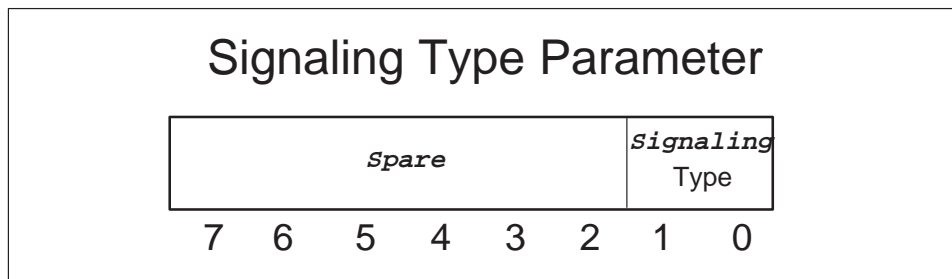
- The PTS SIGINFO MESSAGE AREA field consists of a maximum of 410 bytes and is the area that contains the SigInfo message; however, this field depends on the PTS Message Type.

The following list provides examples of the PTS Message Type contents, which appear in the PTS SigInfo Message Area field.

- Digits to Outpulse: the PTS SigInfo Message Area contains from 1 to 3 *DigitstoOutpulse* parameters (refer to “Digits to outpulse” for more information).
- Digits to Outpulse with Bearer Capability: the PTS SigInfo Message Area contains the *BearerCapability* parameter first in the area, (refer to “Bearer Capability” for more information) with the following bytes containing from 1 to 3 *DigitstoOutpulse* parameters (refer to “Digits to outpulse” for more information).
- PTS Off-Hook: no information appears in the PTS SigInfo Message Area. There is no signaling information associated with a PTS Off-Hook.
- PTS On-Hook: no information appears in the PTS SigInfo Message Area. There is no signaling information associated with a PTS On-Hook.
- Digits Outpulsed: PTS SigInfo Message Area contains the *DigitsOutpulsed* parameter (refer to “Digits outpulsed” for more information).
- SS7 Message Contents:
Use of the TR444 and the GR394 SS7 with ISDN support, define the contents of the SS7 Message Contents.
- PRI Message Contents:
SS7and PRI parameter contents are not re-defined in this document.

Signaling type

This parameter is used in the *Agent_Data* event and in the *New_Call* event.



Optional parameter ID: 32

Parameter length: 1 byte

Parameter contents:

- The Signaling TYPE field consists of two bits and represents the type of signaling the port is using. The values include:
 - 0 0: PTS (4 Wire)
 - 0 1: PTS (2 Wire)
 - 1 0: SS7
 - 1 1: PRI

SigInfo mask

This parameter allows the SCU to control which of the optional SS7 and PRI messages that are reported to the SCU in the `signaling_Event` event notification message.

For example, if the port is an SS7 port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ACM (Address Complete)
- COT (Continuity)
- CPG (Call Progress)
- FAA (Facility Activation), FAR (Facility Request), FRJ (Facility Reject)
- PAM (Pass Along Message)
- RES (Resume) and SUS (Suspend)

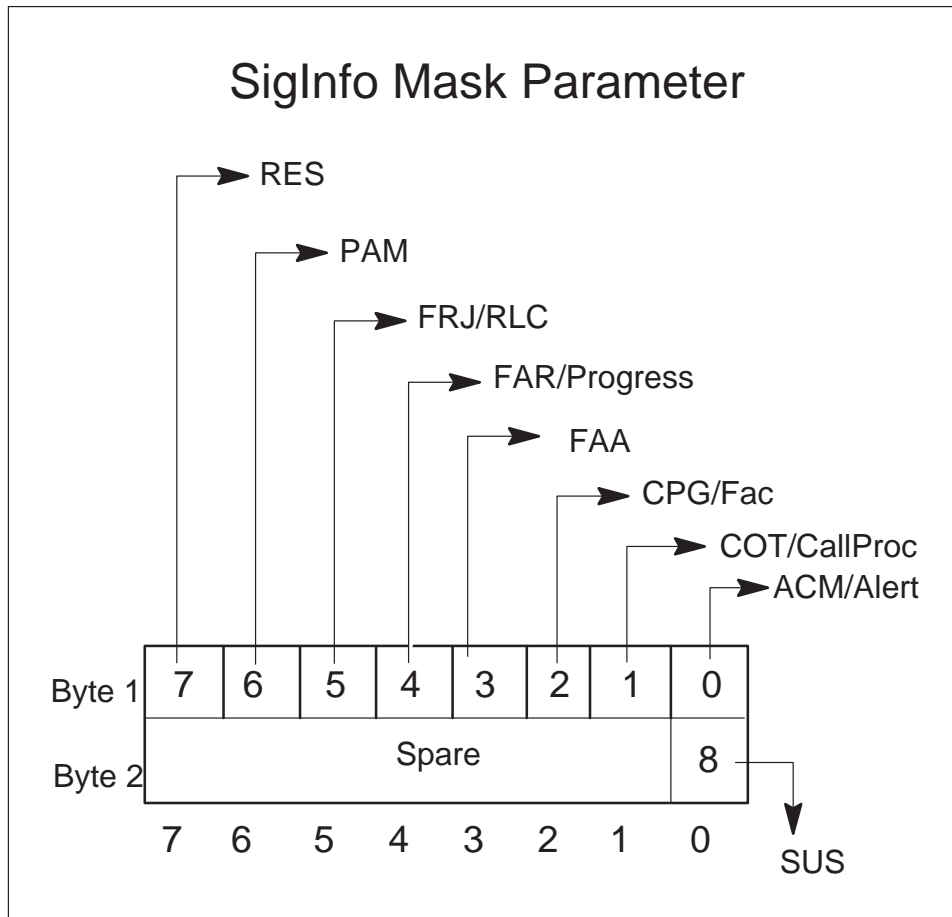
If the port is a PRI port, the SCU may choose to enable or disable the receipt of the following messages from the PSN:

- ALERT (Alerting Message)
- CALLPROC (Call Proceeding)
- FAC (Facility)
- RLC (Release Complete)
- PROG (Progress)

The messages that follow are mandatory. The receipt of these messages cannot be turned off by the SCU. In addition, these messages are always sent to the SCU. These messages include:

- ANM (Answer), REL (Release)/RSC (Reset Circuit) for SS7 ports

- CONNECT and REL (Release)/DISC (Disconnect) for PRI ports



Optional parameter ID: 24

Parameter length: 2 bytes

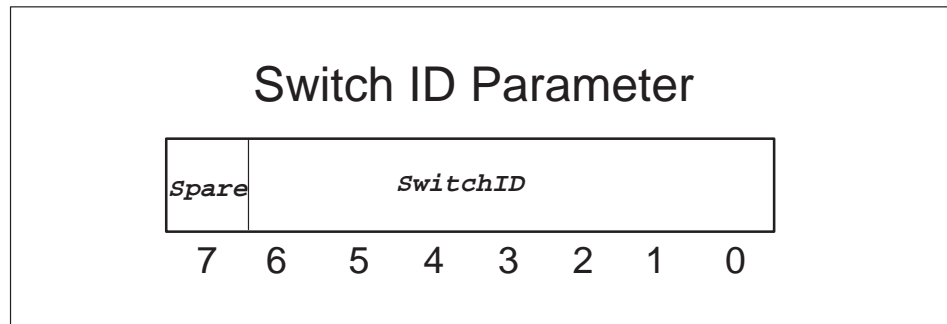
Parameter contents:

- The SIGINFO MASK BITMAP field consists of nine bits and is a bitmap, where each bit indicates the connection status of the agent. The bits are used to represent the connection states as specified below:
 - Bit 0: ACM/Alert
 - Values: 0 = do not send ACM (for SS7 port) or Alert (for PRI port) message to the SCU.

-
- 1 = send the ACM (for SS7 port) or Alert (for PRI port) to the SCU.
 - Bit 1: COT/CallProc
 - Values: 0 = do not send COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU
 - 1 = send the COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU
 - Bit 2: CPG/FAC
 - Values: 0 = do not send CPG (for SS7 port) or FAC (for PRI port) message to the SCU
 - 1 = send the CPG (for SS7 port) or FAC (for PRI port) to the SCU
 - Bit 3: FAA
 - Values: 0 = do not send FAA (for SS7 port) message to the SCU
 - 1 = send the FAA (for SS7 port) to the SCU
 - Bit 4: FAR/Progress
 - Values: 0 = do not send FAR (for SS7 port) or Progress (for PRI port) message to the SCU
 - 1 = send the FAR (for SS7 port) or Progress (for PRI port) to the SCU
 - Bit 5: FRJ/RLC
 - Values: 0 = do not send FRJ (for SS7 port) or RLC (for PRI port) message to the SCU
 - 1 = send the FRJ (for SS7 port) or RLC (for PRI port) to the SCU
 - Bit 6: PAM
 - Values: 0 = do not send PAM (for the SS7 port) to the SCU
 - 1 = send the PAM (received on the SS7 port) to the SCU
 - Bit 7: RES
 - Values: 0 = do not send RES (for the SS7 port) to the SCU
 - 1 = send the RES (received on the SS7 port) to the SCU
 - Bit 8: SUS
 - Values: 0 = do not send SUS (for the SS7 port) to the SCU
 - 1 = send the SUS (received on the SS7 port) to the SCU
 - Bit 9 – 15: Spare

Switch ID

This parameter contains the switch ID of the PSN.



Optional parameter ID: 26

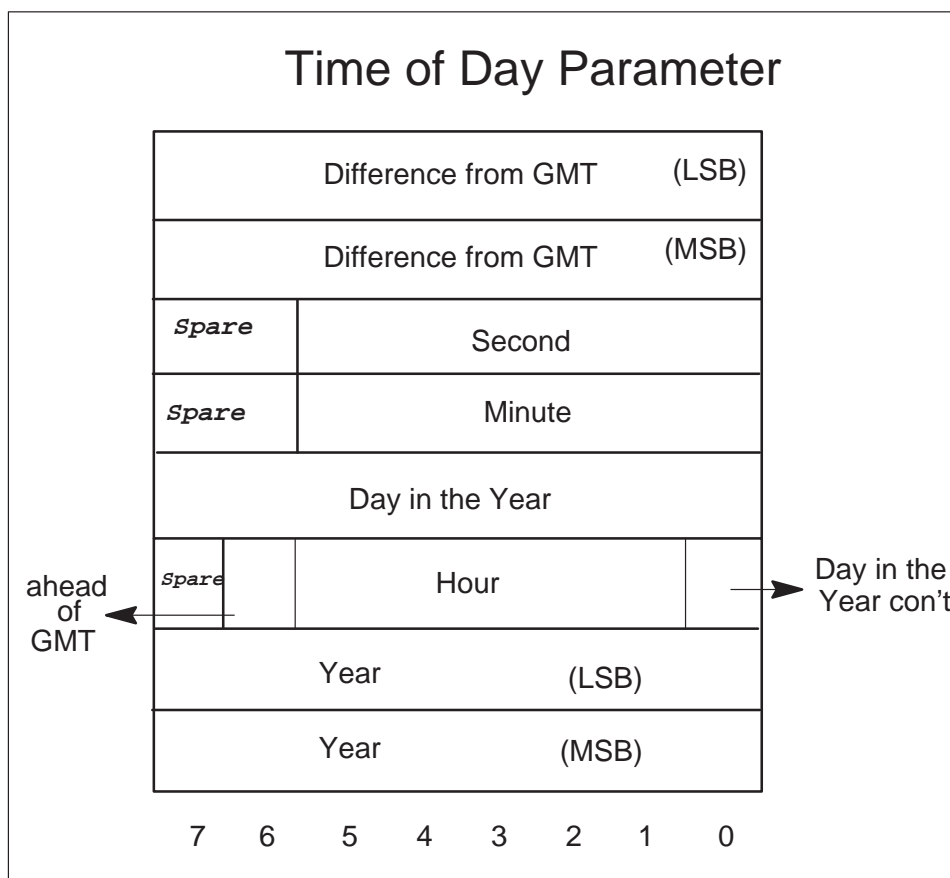
Parameter length: 1 byte

Parameter contents:

- The SWITCH ID field consists of seven bits and contains the switch ID with a range from 0 to 127.

Time of day

This parameter contains the time of the day and is returned in the `Current_Time_of_Day` event notification to the SCU.



Optional parameter ID: 27

Parameter length: 8 bytes

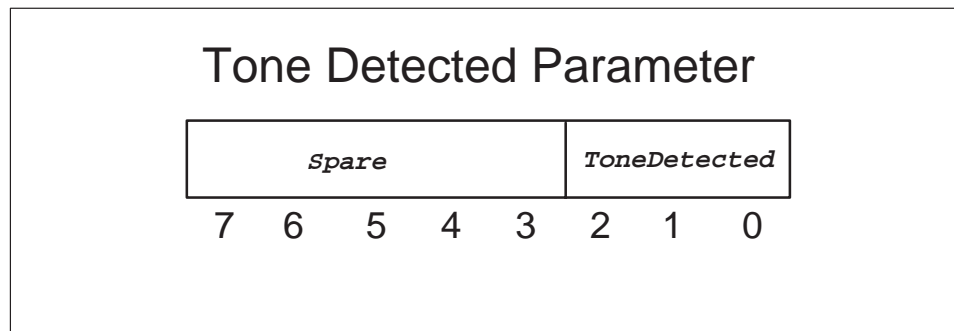
Parameter contents:

- The YEAR field consists of two bytes and contains the year.
- The DIFFERENCE FROM GMT field consists of two bytes and contains the time difference between the time specified by the “hour, minute and second” fields and the Greenwich Mean Time.

- The DAY IN THE YEAR field consists of nine bits and contains the day of the year. The values include 1 to 365.
- The MINUTE field consists of six bits and contains the minute. The values include 0 to 59 with the remaining as spare values.
- The AHEAD OF GMT field consists of one bit. When this field is true, it indicates that the switch time is AHEAD of the Greenwich Mean Time (GMT). The switch time is BEHIND the GMT if this field is set to false.
- The HOUR field consists of five bits and contains the hour. The values include 0 to 23 with the remaining as spare values.
- The SECOND field consists of six bits and contains the minute. The values include 0 to 59 with the remaining as spare values.

Tone detected

This parameter contains the tone that is detected on a given port or agent by the PSN.



Optional parameter ID: 28

Parameter length: 1 byte

Parameter contents:

- The TONE OR DIGIT DETECTED field consists of three bits and contains the tone or the digit that is detected, and is encoded as follows:
 - 0 0 0 0: Tone or Digit Monitoring Aborted
 - 1 0 0 1: Asterisk Detected
 - 2 0 1 0: Octothorpe Detected
 - 3 0 1 1: SF Tone Detected
 - 4 1 0 0: BBF Tone Detected
 - 5 1 0 1: BBF Digits Detected
 - 1 1 0 - 1 1 1: Unused (Spare)

Note: The BBF digits are not sent to the SCU by the PSN.

UCS PSN LOPER messages and parameters version 3

This chapter contains a detailed description of the UCS PSN messages and parameters Version 3 described in the Low Overhead Protocol Encoding Rule (LOPER) message format.

LOPER format

Each of the messages in the LOPER follow the same basic format and rules.

The following rules apply to LOPER:

- A message is byte aligned, meaning that each parameter starts on a new byte.
- The first byte in a message is considered to be the 0th index of the message, the second byte in the message is considered to be the 1st index of the message, and so on.
- Two bytes are used in describing the length of a message and parameter. Two bytes are also used to contain at which byte location in the message that the optional parameters begin in the message.
- A LOPER message can either contain one primitive, one event, or one macro which can contain up to five primitives.
- The SS7 signaling message of the SigInfo contents in an optional *SigInfo* parameter to be transmitted on a SS7 agent is not validated. Since the SS7 signaling message is to be built by the SCU in the TR444 BellCore standard, it is assumed that the SS7 signaling message has been correctly built and is sent onto the SS7 network without validation. Any error in the SS7 signaling message is caught by the network and handled accordingly by the SS7 network.

Messages sent from the SCU to the PSN

This section covers messages sent from the SCU to the PSN.

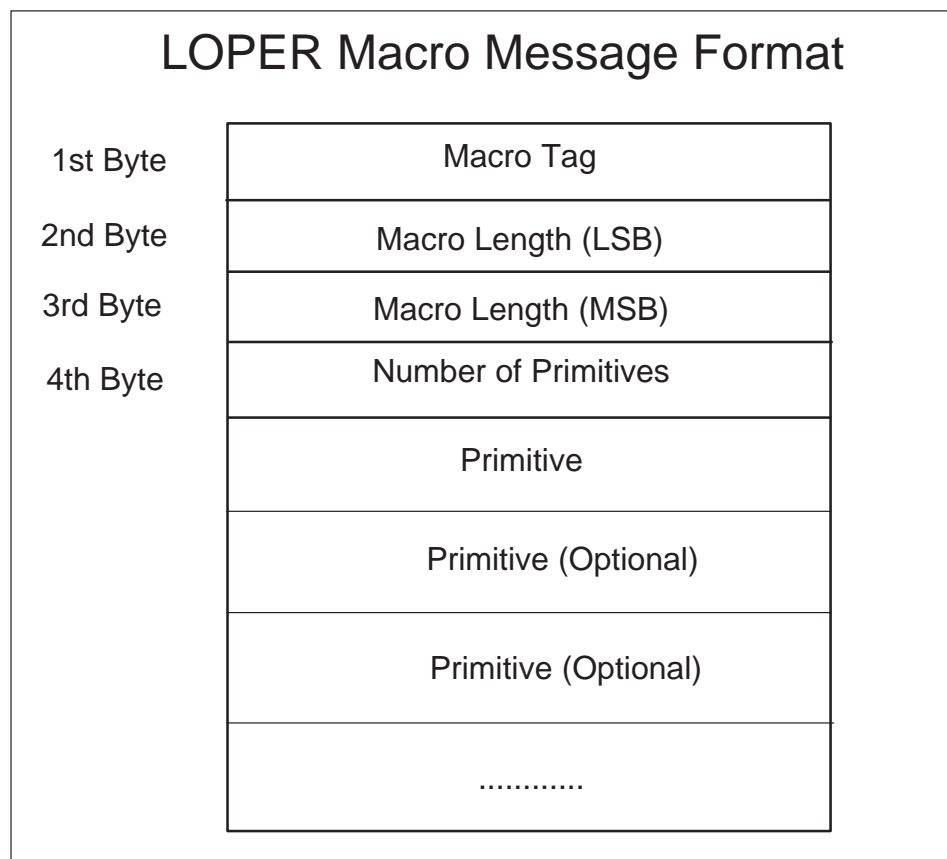
Macros

The first byte of a macro designates that the message is a macro. The second and third bytes indicate the length of the macro in the number of bytes that

comprise the length of the message from the Macro Tag to the end of the last primitive in the macro. The fourth byte identifies how many primitives there are in the primitive Area. The primitive area contains the individual primitives.

If there is an error in the decoding of one of the primitives found in the macro, that primitive is dropped and the decoding process is ceased since the rest of the macro is considered to be corrupted. Any primitive decoded before the error is still processed.

LOPER macro message format



- The MACRO TAG field consists of one byte and is the location which is used to identify the primitive in the message. The values include:
 0 0 0 0 0 0 0 0 : Macro Tag

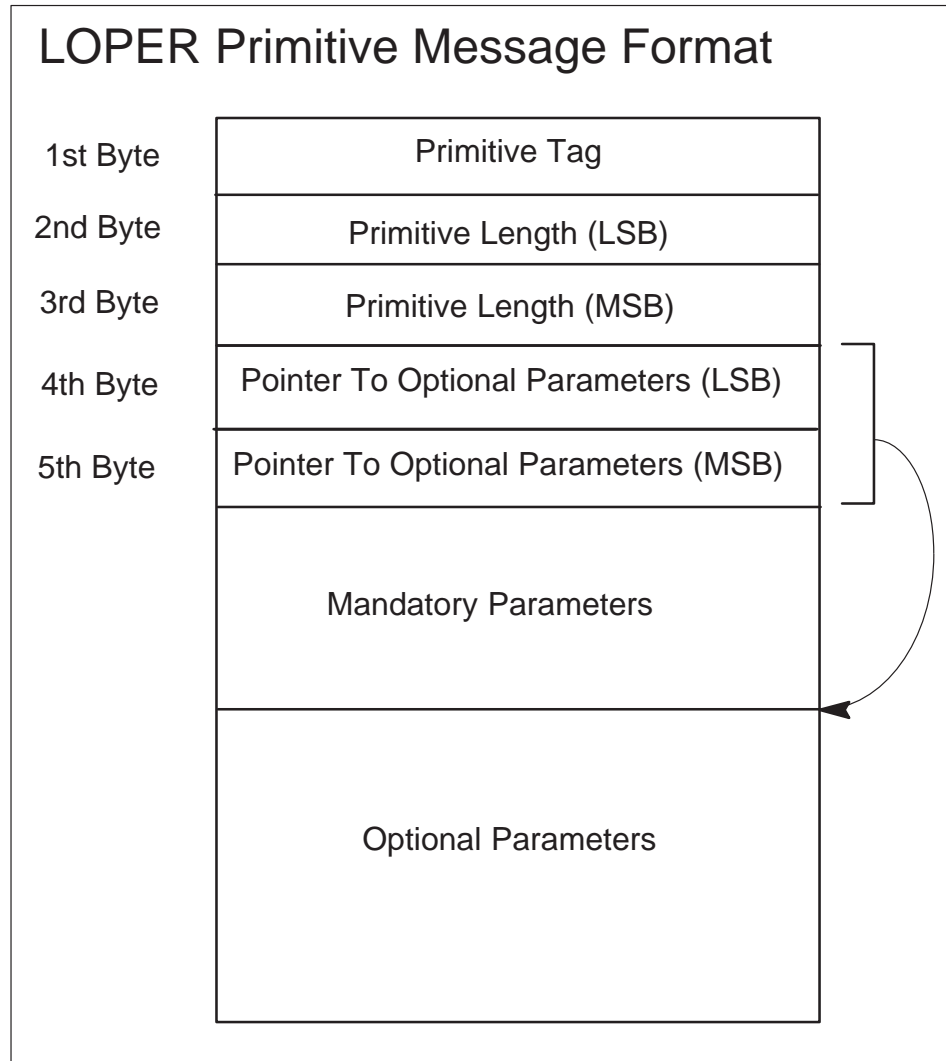
Primitives

The first byte is the primitive tag and identifies the primitive. The second and third bytes indicate the length of the primitive beginning with the primitive tag to the end of the mandatory parameters or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters. The contents of the mandatory parameter depend on the primitive and event and the customer using LOPER. Refer to Figure 19-1.

Figure 19-1
LOPER primitive message format

- The PRIMITIVE TAG field consists of one byte and is a byte that is used to identify the primitive in the message. The values include:

0	00000000	: Nil Primitive Tag
1	00000001	: Bridge
2	00000010	: Collect Digits
3	00000011	: Connect
4	00000100	: Disconnect
5	00000101	: Hold
6	00000110	: Monitor
7	00000111	: Mute
8	00001000	: RESERVED FOR INTERNAL PSN USE
9	00001001	: New Call Accepted
10	00001010	: New Call Rejected
11	00001011	: Play Message
12	00001100	: PPCD
13	00001101	: Query Port
14	00001110	: Reconnect
15	00001111	: Set Billing Record
16	00010000	: Stop Message
17	00010001	: Transmit Signaling Info
18	00010010	: Error Detected
19	00010011	: Heartbeat
20	00010100	: Port Status
21	00010101	: Query TOD
22	00010110	: Reset Switch
23	00010111	: Set IP Address
24	00011000	: Flow Control
	00011001 to 11111111	: Spare

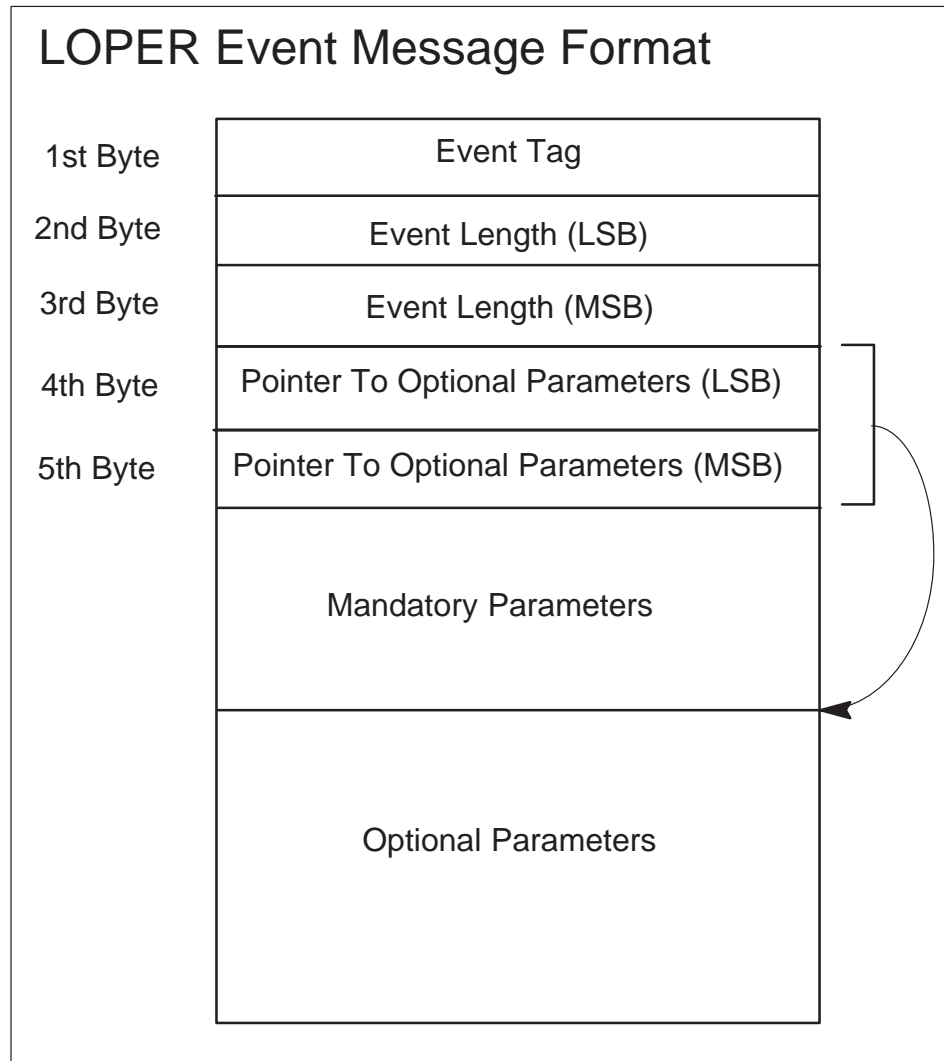


Messages sent from the PSN to the SCU

Events

The first byte is the Event Tag and identifies the event. The second and third bytes indicate the length of the event, beginning with the Event Tag to the end of the mandatory parameters, or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the mandatory parameters. The contents of the mandatory parameters depend on the primitive and event, and the customer using LOPER. Refer to the following figure.

LOPER event message format



- The EVENT TAG field consists of one byte and is a byte that is used to identify the event in the message. The values include:

```

0 00000000 : Current TOD
1 00000001 : Digits Collected
2 00000010 : Error Detected
3 00000011 : In Service
4 00000100 : Instruction Completed
5 00000101 : Message Played
6 00000110 : New Call
7 00000111 : Off Hook
8 00001000 : On Hook
9 00001001 : Port Status

```

10 0 0 0 0 1 0 1 0 : Query Port
11 0 0 0 0 1 0 1 1 : Route Not Available
12 0 0 0 0 1 1 0 0 : Route Selected
13 0 0 0 0 1 1 0 1 : Signaling Event
14 0 0 0 0 1 1 1 0 : Tone Detected
15 0 0 0 0 1 1 1 1 : Agent Data
16 0 0 0 1 0 0 0 0 : Stop Heartbeat
0 0 0 1 0 0 0 1 to 1 1 1 1 1 1 1 1 : Spare

Parameters

LOPER divides parameters into two classes, mandatory and optional. Mandatory parameters contain the information required by the primitive and event in order to be processed. Optional parameters contain the information not required by the primitive and event but add extra information for processing. For a mapping of which parameters are mandatory and/or optional for primitives and events, refer to the tables within Chapter “PRI messages.”

Mandatory parameters

Mandatory parameters in LOPER are customized for each primitive and event. The sixth byte in the primitive and event is always the beginning of the mandatory parameters for primitives and events. There is no need for a mandatory parameter tag. Mandatory parameters are identified by the primitive and event ID and can only be found in their corresponding primitive and event, such as a *Connect* mandatory parameter can only be found in a *Connect* primitive. Also, there is no need for a length indicator for the mandatory parameters. For example, since the pointer to the optional parameters designates the end of the mandatory parameters, and if there are no optional parameters, then the length of the primitive and event is the end of the mandatory parameters. The contents of mandatory parameters is covered in the Chapter “PSN parameters Version 1.”

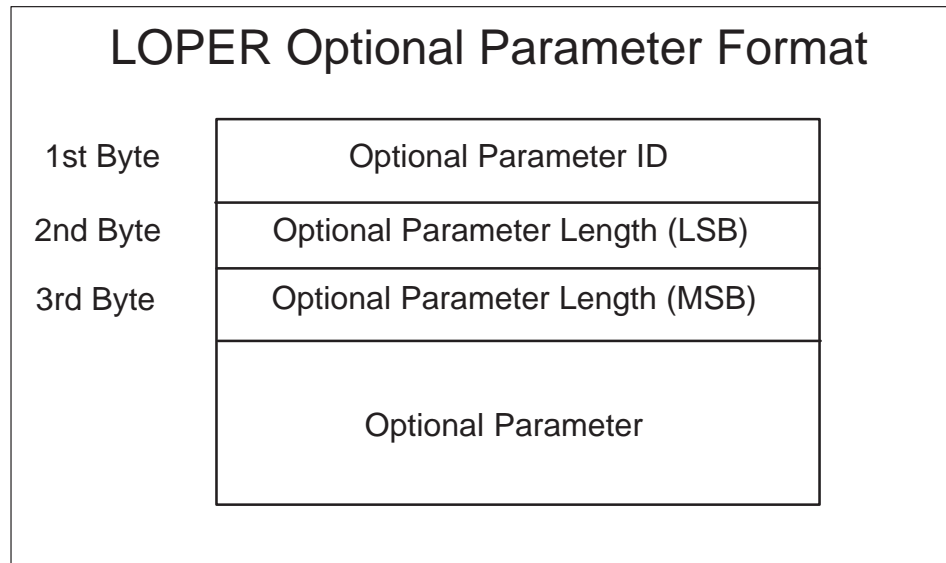
If there is an error in the decoding of the mandatory parameters, then entire message is dropped, the OM PSN_ERPS:DECODEFL is incremented, the logs PSN202 and PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause HEADER_DECODE_FAILURE_EC.

If there is an error in the validating of the mandatory parameters values, then the entire message is dropped, the OM PSN_ERPS:MANDPDEF is incremented, the logs PSN202 and PSN212 are produced, and an **Error_Detected** event is produced with the Error Cause PARAM_CONTENTS_OUT_OF_RANGE_EC.

Optional parameters

The first byte of an optional parameter is the optional parameter tag, which identifies the optional parameter. The second and third bytes indicate the length of the optional parameter in bytes, starting with the beginning of the optional parameter ID, to the end of the optional parameter. The contents of the optional parameter depends on the specific parameter and the customer using LOPER. The contents of the optional parameters are defined in PSN PARMS. Refer to the following figure.

If a LOPER message is received with an unknown optional parameter ID, the unknown optional parameter is skipped and the decoding of the rest of the optional parameters in the message continues. If there is an error in the decoding of a known optional parameter, the optional parameter is dropped and the decoding process is ceased. As a result, the reset of the message is considered to be corrupted, and the OM PSN_ERPS:OPPRMDEF is incremented, the logs PSN202 and PSN212 are produced, and the primitive, along with the optional parameters, are decoded before the error is processed.



- The OPTIONAL parameter ID field consists of one byte and is the location which is used to represent the parameter. The values include:
 - 0 00000000 : Unknown
 - 1 00000001 : Bearer Capability
 - 2 00000010 : Billing Info
 - 3 00000011 : Call Reference Identifier
 - 4 00000100 : Control Info
 - 5 00000101 : Destination Trunk Group

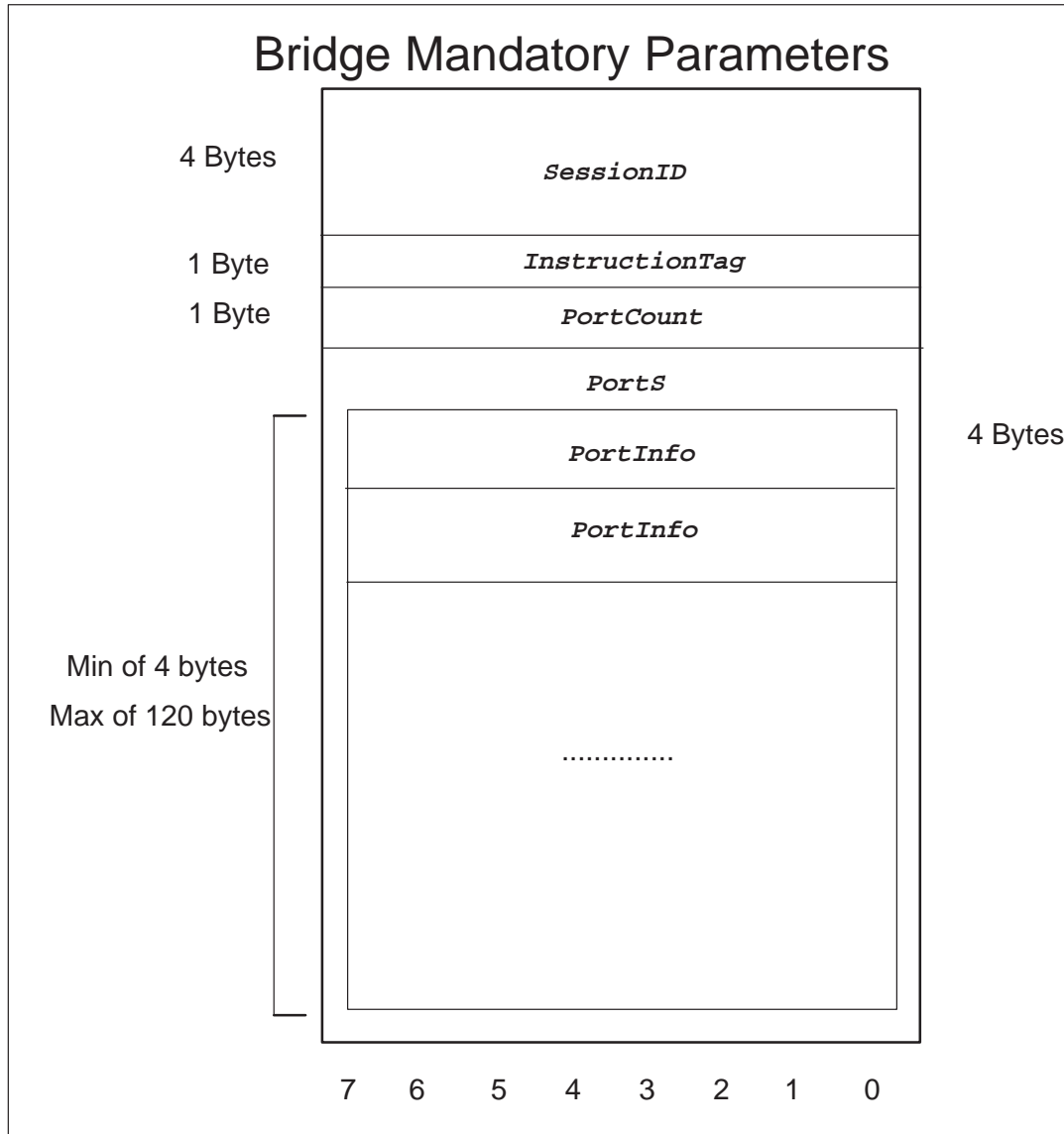
6 00000110 : Digit Collection
7 00000111 : Digits Collected
8 00001000 : Digits Outpulsed
9 00001001 : Digits To Outpulse
10 00001010 : Error Cause
11 00001011 : Flow Control Information
12 00001100 : Flow Control Encountered
13 00001101 : Instruction ID
14 00001110 : Instruction Tag
15 00001111 : Message Info
16 00010000 : Monitor Mask
17 00010001 : Parameter ID
18 00010010 : Port Count
19 00010011 : Port Info
20 00010100 : Port Service Information
21 00010101 : Port Status
22 00010110 : Reset Reason
23 00010111 : Session ID
24 00011000 : SigInfo Mask
25 00011001 : Signaling Info
26 00011010 : Switch ID
27 00011011 : Time of Day
28 00011100 : Tone Detected
29 00011101 : Agent Type
30 00011110 : COT Required
31 00011111 : ISUP Index
32 00100000 : Signaling Type
33 00100001 to 10000001 : Spare Values
130 10000010 : UCS Point In Call
131 10000011 : UCS STS
10000100 to 11111111 : Spare Values

PSN LOPER mandatory parameters for primitives

The following mandatory parameters are used to build primitives. Their description includes the length of the parameter as well as the type of each field in the parameter.

Bridge mandatory parameters

The mandatory parameters for a **Bridge** primitive are as follows:



Parameter length: minimum of 10 bytes and a maximum of 126 bytes

Parameter contents:

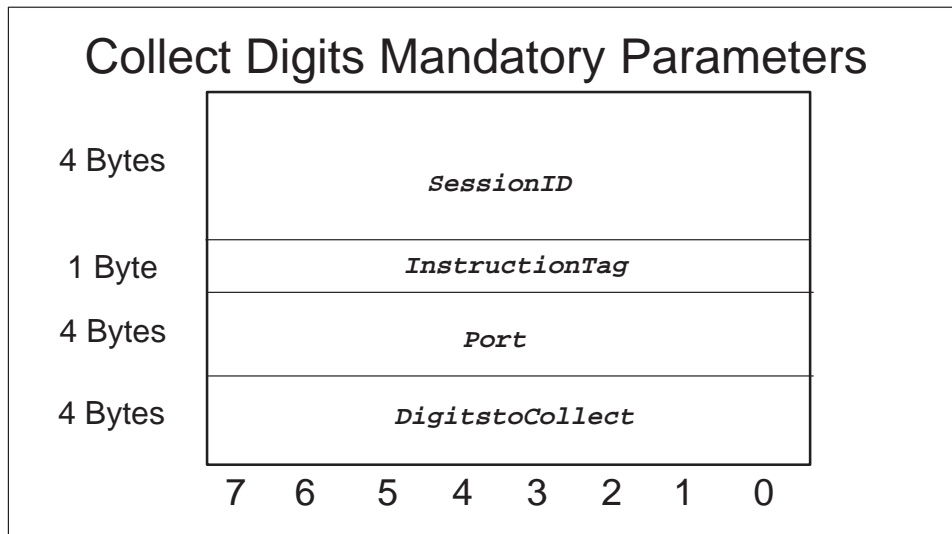
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The PORT COUNT field consists of one byte and is the location which includes the TYPE: Count of 1 to 30.
- The PORTS field consists of a minimum of four bytes and a maximum of 120 bytes. Also, the PORTS field consists of a minimum of one port to a maximum of 30 ports.

The Port TYPE: *PortInfo* parameter.

Collect digits mandatory parameters

The mandatory parameters for a `Collect_Digits` primitive.



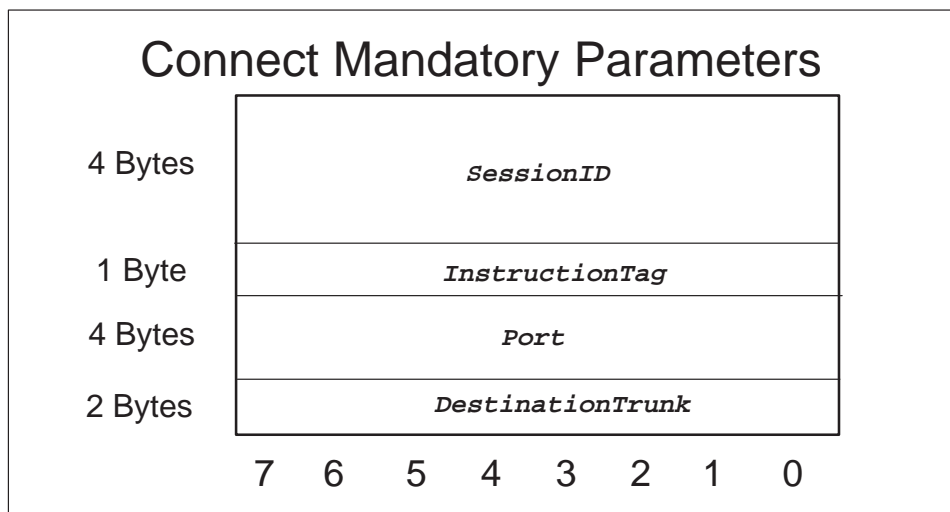
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitCollection* parameter.

Connect mandatory parameters

The mandatory parameters for a `Connect` primitive are as follows:



Parameter length: 11 bytes

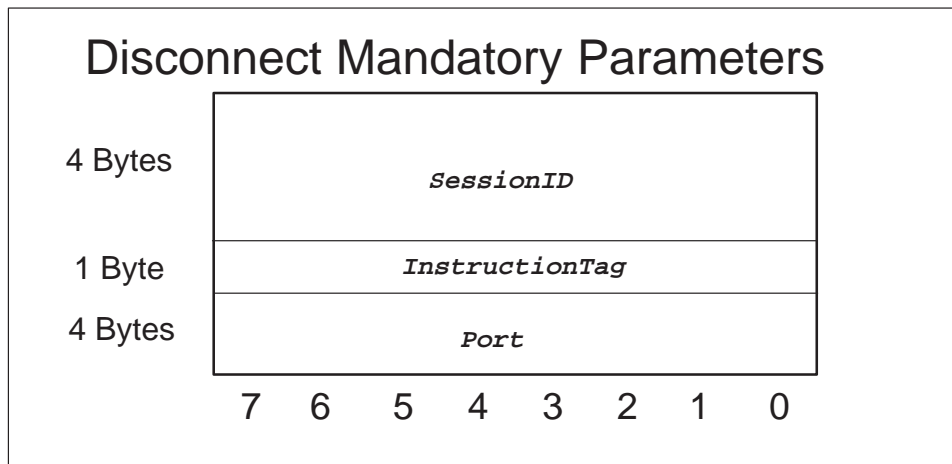
Instruction

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DESTINATION TRUNK field consists of two bytes and is the location which includes the TYPE: *Destinationtrunkgroup* parameter.

Disconnect mandatory parameters

The mandatory parameters for a **Disconnect** primitive are as follows:



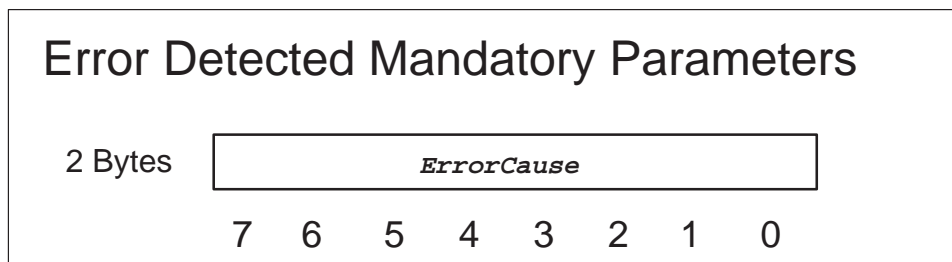
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Error detected mandatory parameters

The mandatory parameters for a **Error_Detected** primitive are as follows:



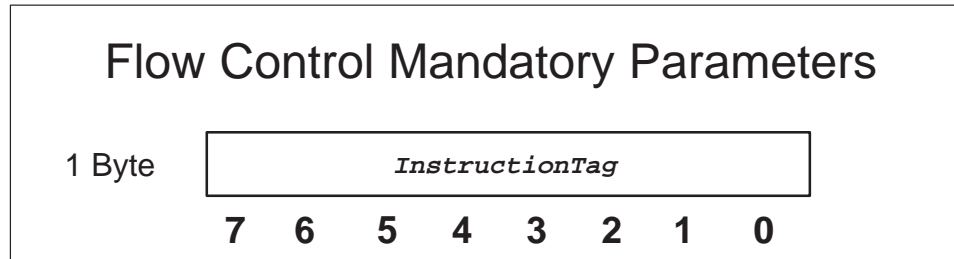
Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

Flow control mandatory parameters

The mandatory parameters for a `Flow_Control` primitive are as follows:



Parameter length: 1 byte

Parameter contents:

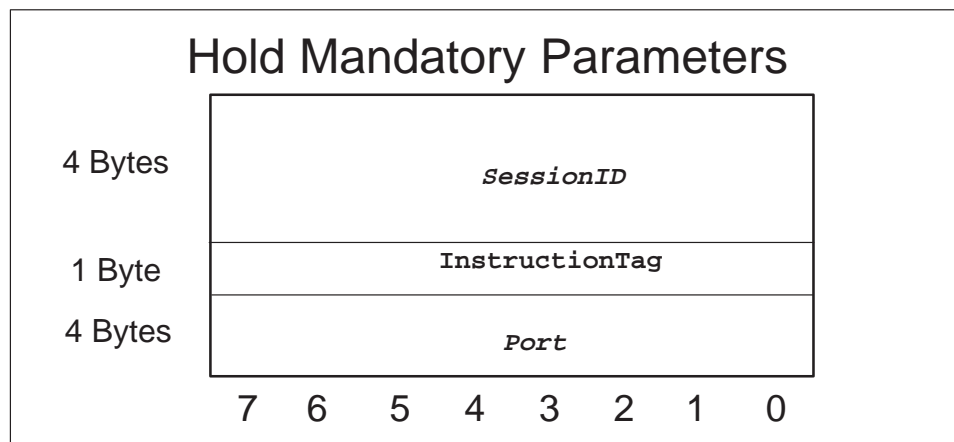
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Heartbeat mandatory parameters

The `Heartbeat` primitive does not contain any information in the primitive. A `Heartbeat` primitive message contains only what is mandatory for a LOPER primitive which is the primitive tag, the primitive length, and the optional pointer, totaling five bytes.

Hold mandatory parameters

The mandatory parameters for a `Hold` primitive are as follows:



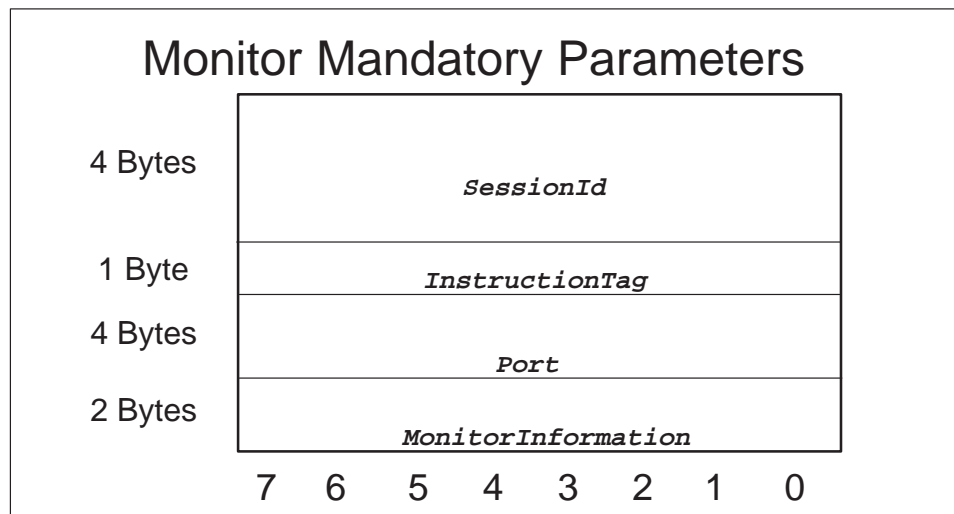
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Monitor mandatory parameters

The mandatory parameters for a **Monitor** primitive are as follows:



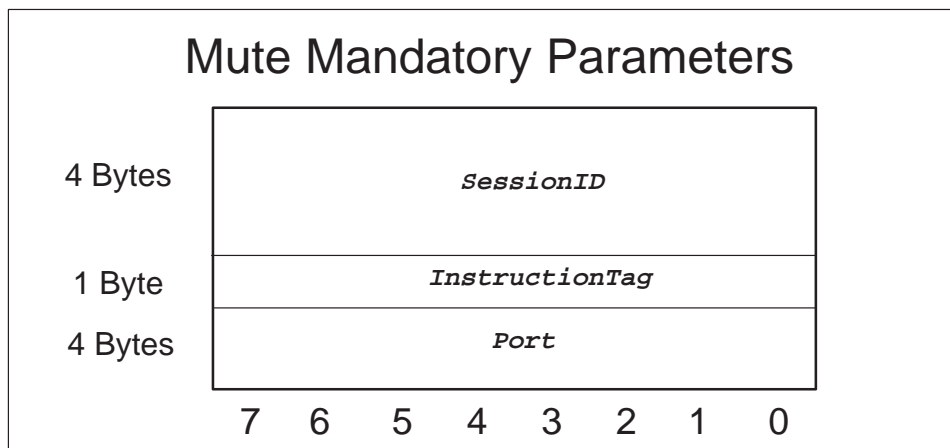
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MONITOR INFORMATION field consists of two bytes and is the location which includes the TYPE: *MonitorMask* parameter.

Mute mandatory parameters

The mandatory parameters for a `Mute` primitive are as follows:



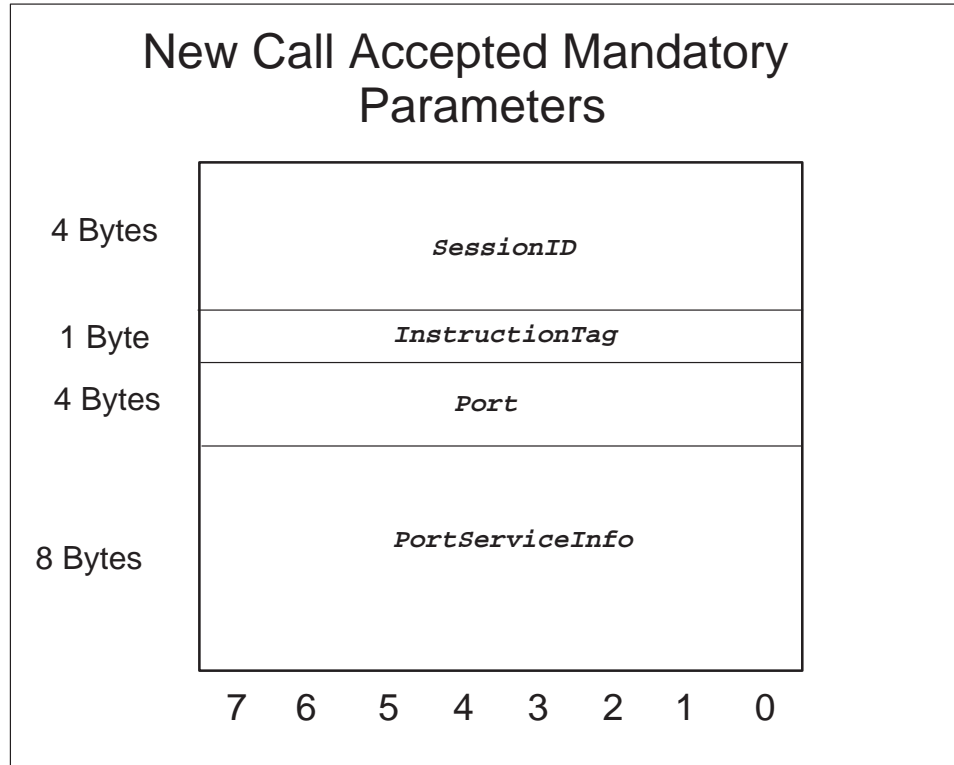
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call accepted mandatory parameters

The mandatory parameters for a `New_Call_Accepted` primitive are as follows:



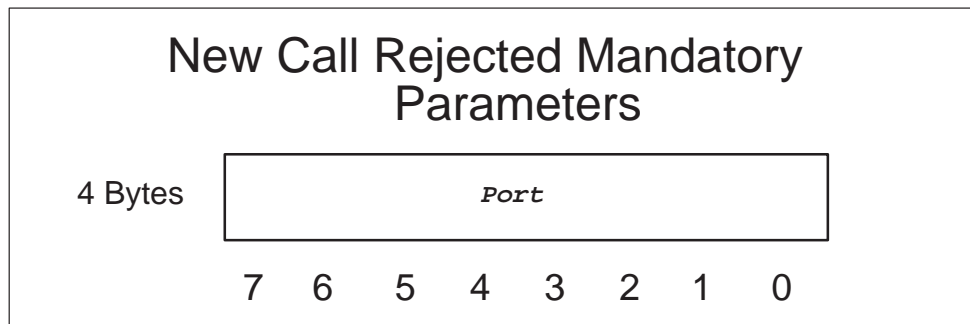
Parameter length: 17 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT SERVICE INFO field consists of 8 bytes and is the location which includes the TYPE: *PortServiceInfo* parameter.

New call rejected mandatory parameters

The mandatory parameters for a `New_Call_Rejected` primitive are as follows:



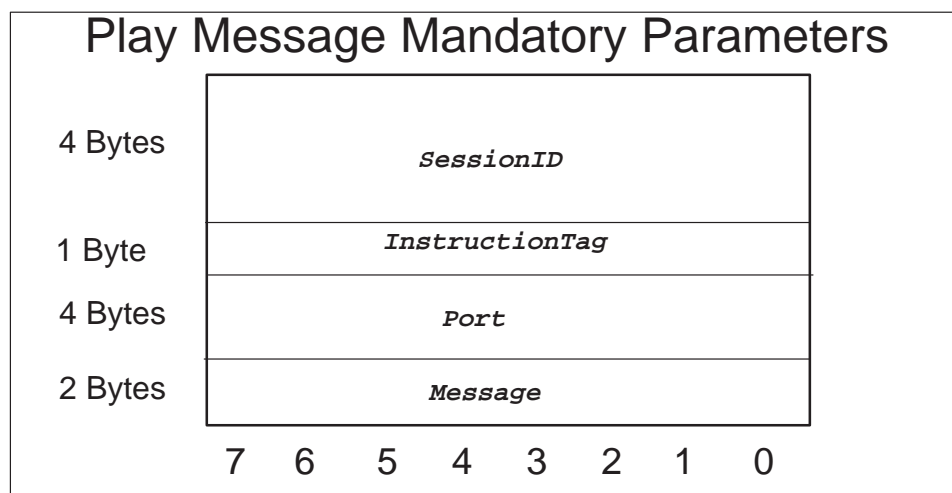
Parameter length: 4 bytes

Parameter contents:

- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Play message mandatory parameters

The mandatory parameters for a `Play_Message` primitive are as follows:



Parameter length: 11 bytes

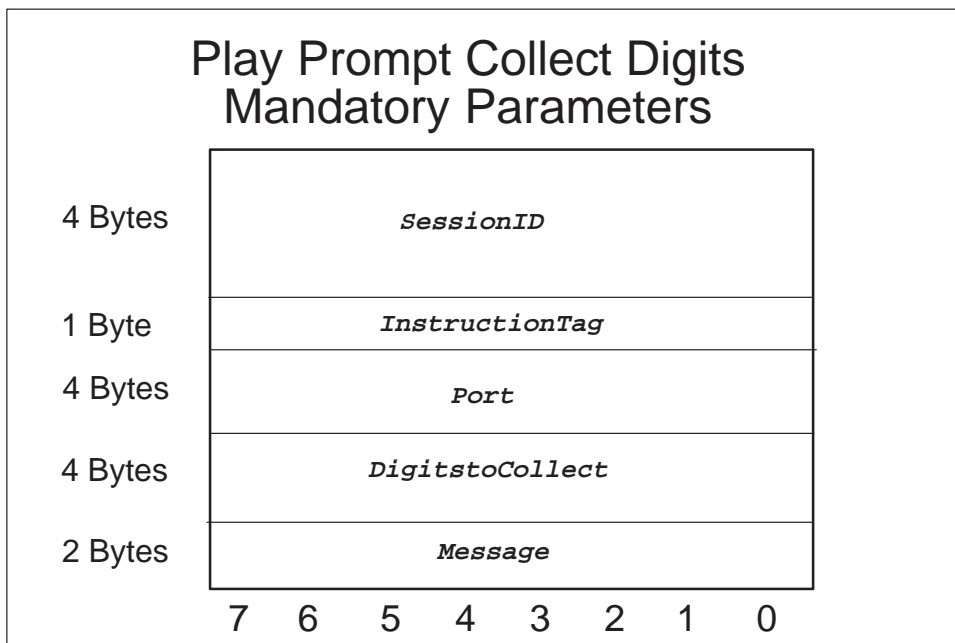
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Play prompt collect digits mandatory parameters

The mandatory parameters for a `Play_Prompt_Collect_Digits_&_Report` primitive are as follows:



Parameter length: 15 bytes

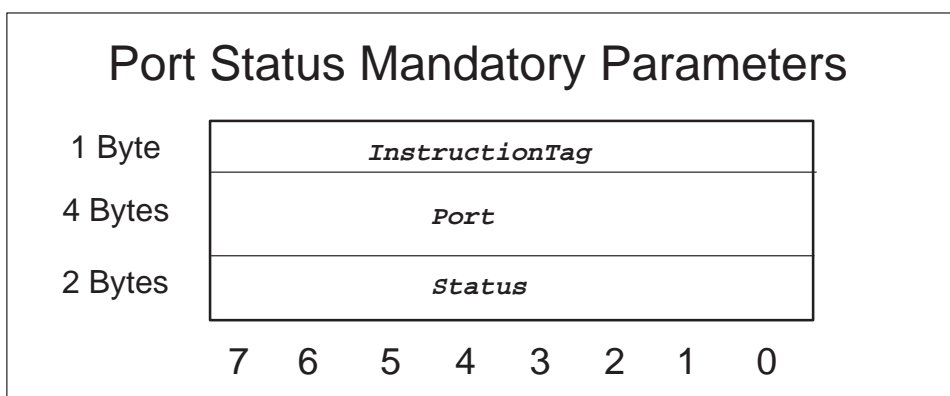
Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The DIGITS TO COLLECT field consists of four bytes and is the location which includes the TYPE: *DigitCollection* parameter.

- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` primitive are as follows:



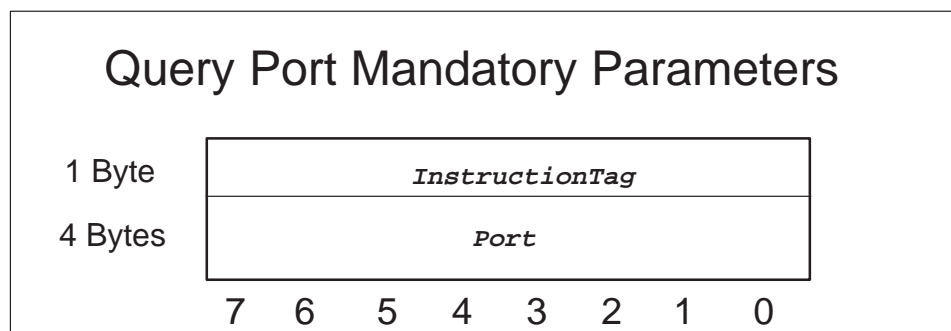
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` primitive are as follows:



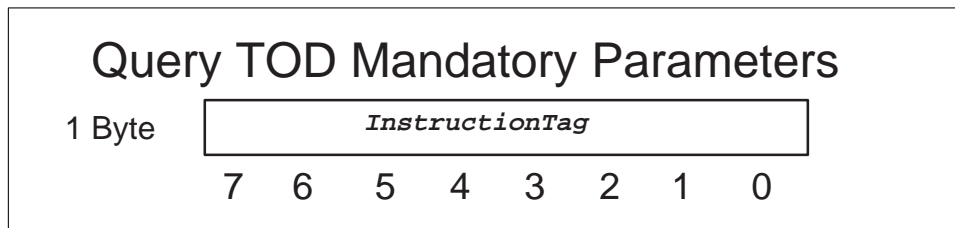
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Query time of day (TOD) mandatory parameters

The mandatory parameters for a `query_TOD` primitive are as follows:



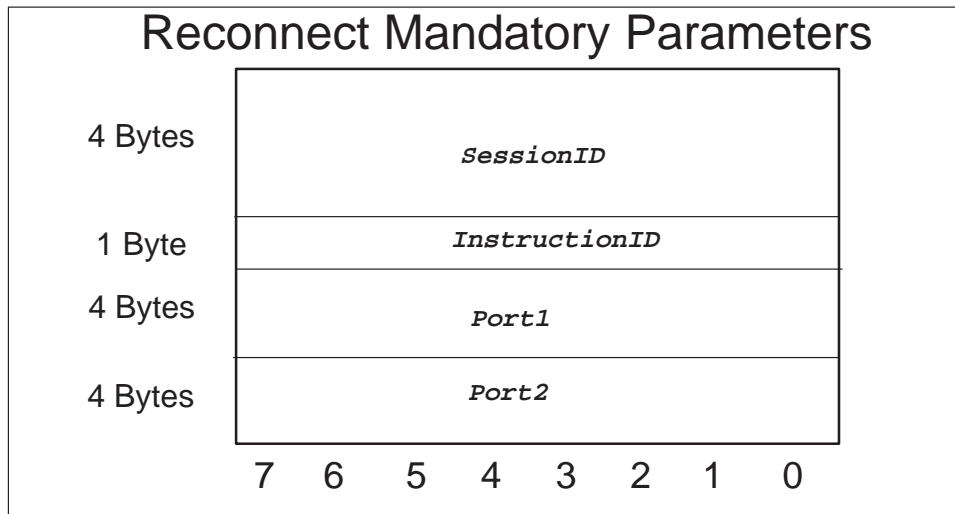
Parameter length: 1 byte

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Reconnect mandatory parameters

The mandatory parameters for a `Reconnect` primitive are as follows:



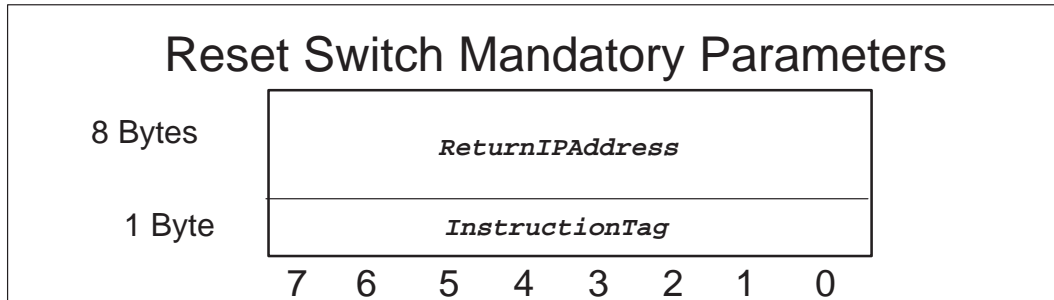
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT 1 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The PORT 2 field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Reset switch mandatory parameters

The mandatory parameters for a `Reset_Switch` primitive are as follows:



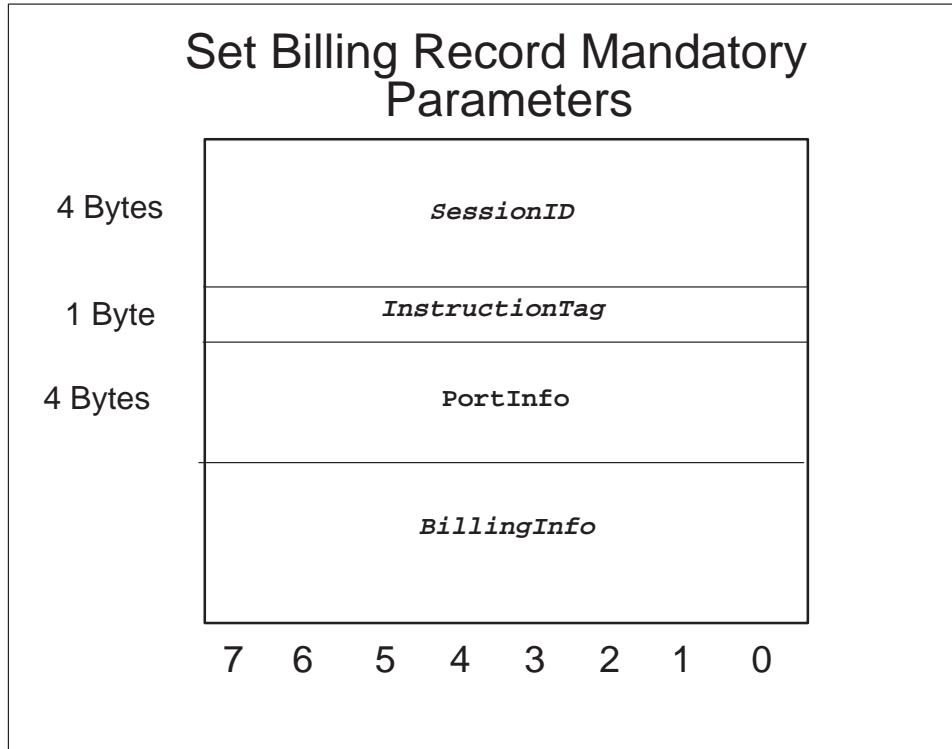
Parameter length: 9 bytes

Parameter contents:

- The RETURN IP ADDRESS field consists of 8 bytes and is the location which includes the TYPE: *PortService* parameter.
Note: There is one mandatory PSI parameter called the Return IP Address parameter, which is used to send the **Instruction_Completed**. There are 0 to 20 optional PSI parameters called the IP Address to Reset parameters, which are used to do the actual resetting of calls.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Set billing record mandatory parameters

The mandatory parameters for a `Set_Billing_Record` primitive are as follows:



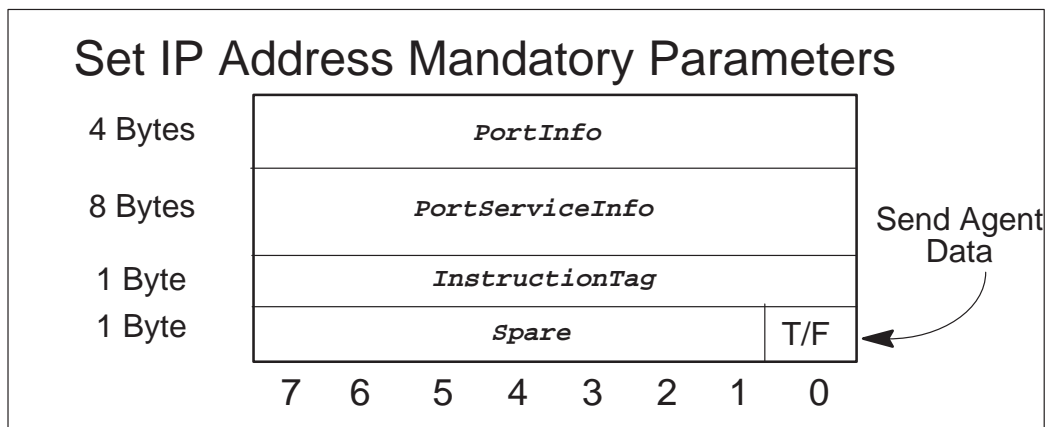
Parameter length: variable in size

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The Port Info field consists of four bytes and is the location which includes the TYPE *PortInfo* parameter.
- The BILLING INFORMATION is variable in size and is the location which includes the TYPE: *BillingInfo* parameter.

Set IP address mandatory parameters

The mandatory parameters for a `Set_IP_Address` primitive are as follows:



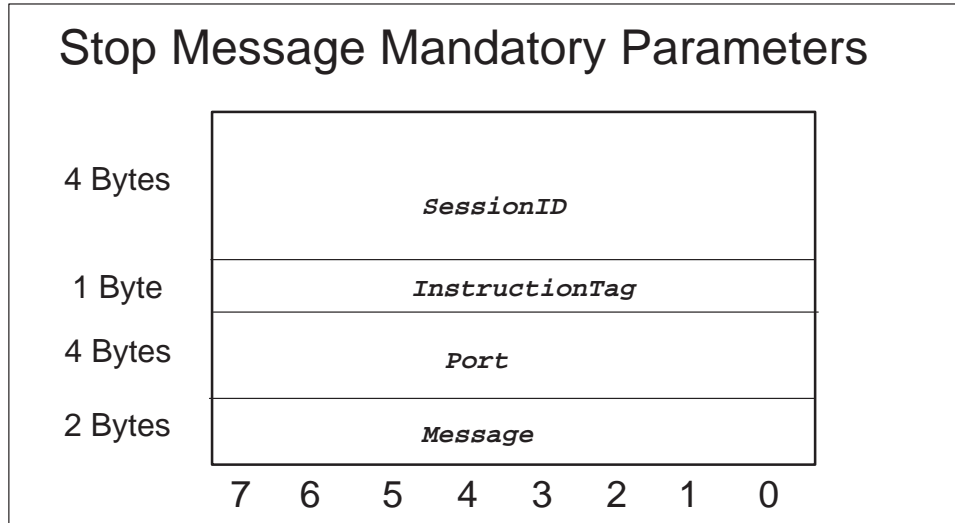
Parameter length: 14 bytes

Parameter contents:

- The PORT field consists of 4 bytes and is the location which includes the TYPE *Port* parameter.
- The PORT SERVICE INFO field consists of 8 bytes and is the location which includes the TYPE: *PortServiceInfo* parameter.
- The SEND AGENT DATA field consists of 1 bit and is the location which includes the TYPE: Boolean (TRUE = 1, FALSE = 0).
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.

Stop message mandatory parameters

The mandatory parameters for a `Stop_Message` primitive are as follows:



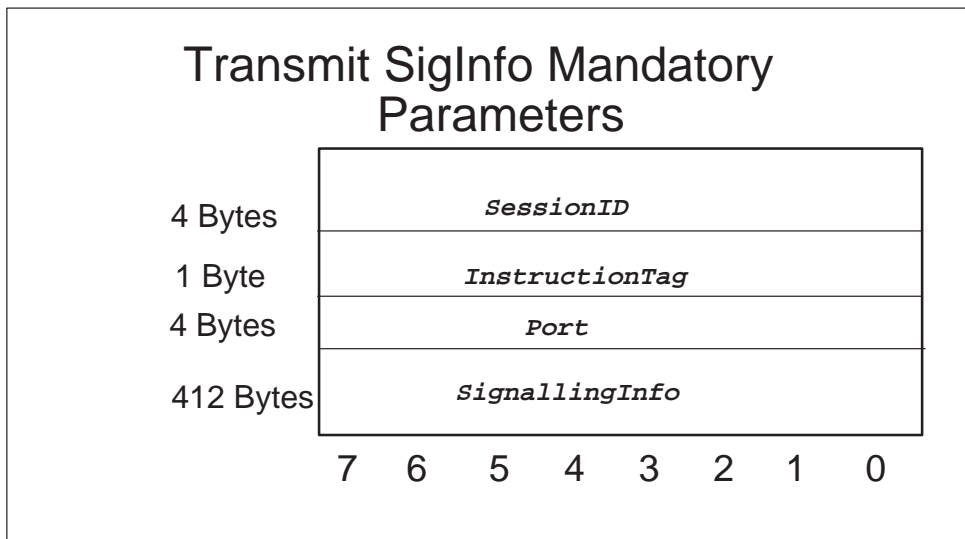
Parameter length: 11 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The MESSAGE field consists of two bytes and is the location which includes the TYPE: *MessageInfo* parameter.

Transmit siginfo mandatory parameters

The mandatory parameters for a `Transmit_siginfo` primitive are as follows:



Parameter length: 421 bytes

Parameter contents:

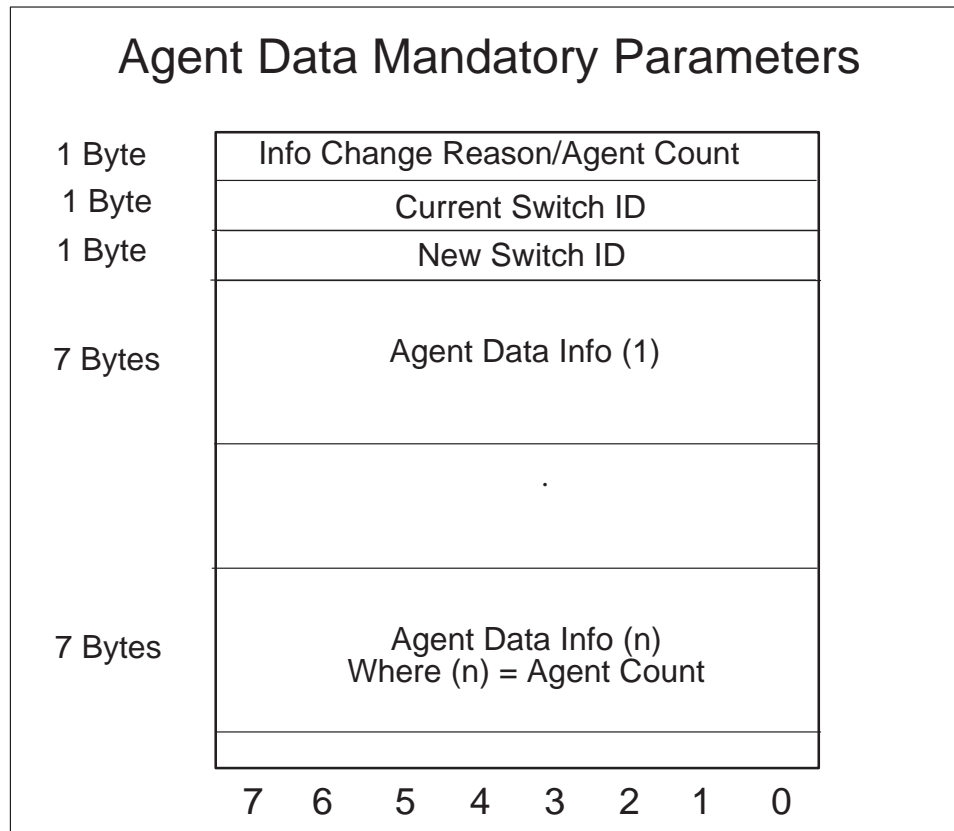
- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The SIGNALLING INFO field consists of 412 Bytes and is the location which includes the TYPE: *SignalingInfo* parameter.

PSN LOPER mandatory parameters for events

The following mandatory parameters are used to build event notifications. Their description includes the parameter length as well as the type of each field in the parameter.

Agent data mandatory parameters

The mandatory parameters for an `Agent_Data` Event are as follows:



Parameter length: a minimum of 10 bytes and a maximum of 444 bytes

Parameter contents:

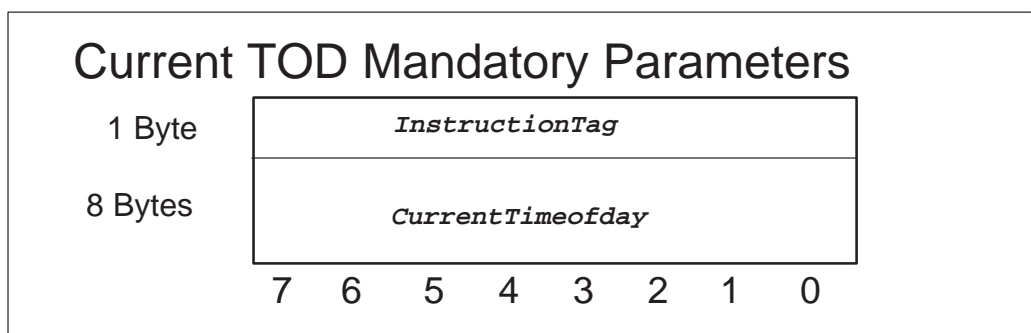
- The INFO CHANGE REASON/AGENT COUNT field consists of one byte and is the location which includes the TYPE: *InfoChangeReason* parameter.
- The CURRENT SWITCH ID field consists of one byte and is the location which includes the TYPE: *SwitchID* parameter.
- The NEW SWITCH ID field consists of one byte and is the location which includes the TYPE: *SwitchID* parameter.

- The AGENT DATA INFO 1 field consists of seven bytes and is the location which includes the TYPE: *AgentDataInfo* parameter.
- The AGENT DATA INFO n field consists of seven bytes and is the location which includes the TYPE: *AgentDataInfo* parameter.

Note: The number of *AgentDataInfo* parms must equal the Agent Count Value in the first byte.

Current time of day (TOD) mandatory parameters

The mandatory parameters for a `Current_TOD` Event are as follows:



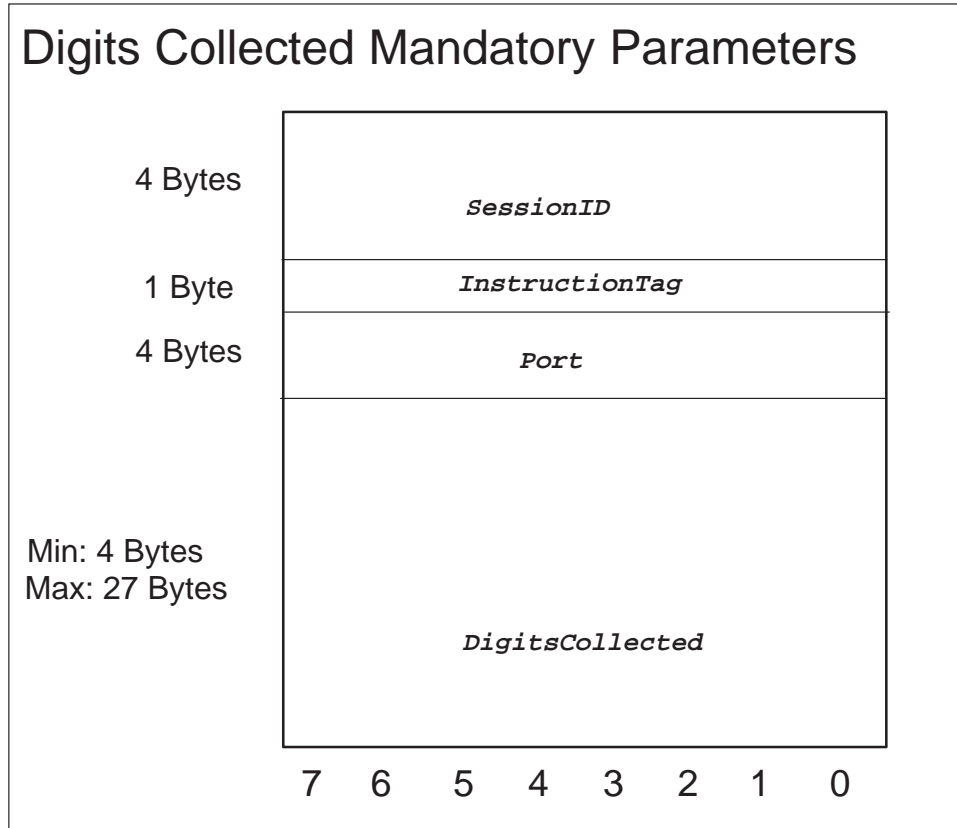
Parameter length: 9 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The CURRENT TIME OF DAY field consists of eight bytes and is the location which includes the TYPE: *Timeofday* parameter.

Digits collected mandatory parameters

The mandatory parameters for a `Digits_Collected` Event are as follows:



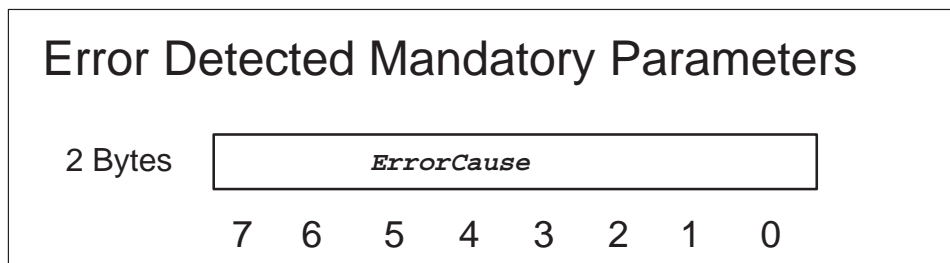
Parameter length: a minimum of 13 bytes and a maximum of 36 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: `SessionID` parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: `InstructionTagID` parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: `PortInfo` parameter.
- The DIGITS COLLECTED field consists of a minimum of four bytes and a maximum of 27 bytes and is the location which includes the TYPE: `DigitsCollected` parameter.

Error detected mandatory parameters

The mandatory parameters for a **Error_Detected** Event are as follows:



Parameter length: 2 bytes

Parameter contents:

- The ERROR CAUSE field consists of two bytes and is the location which includes the TYPE: *ErrorCause* parameter.

In service mandatory parameters

The mandatory parameters for a **In_service** Event are as follows:



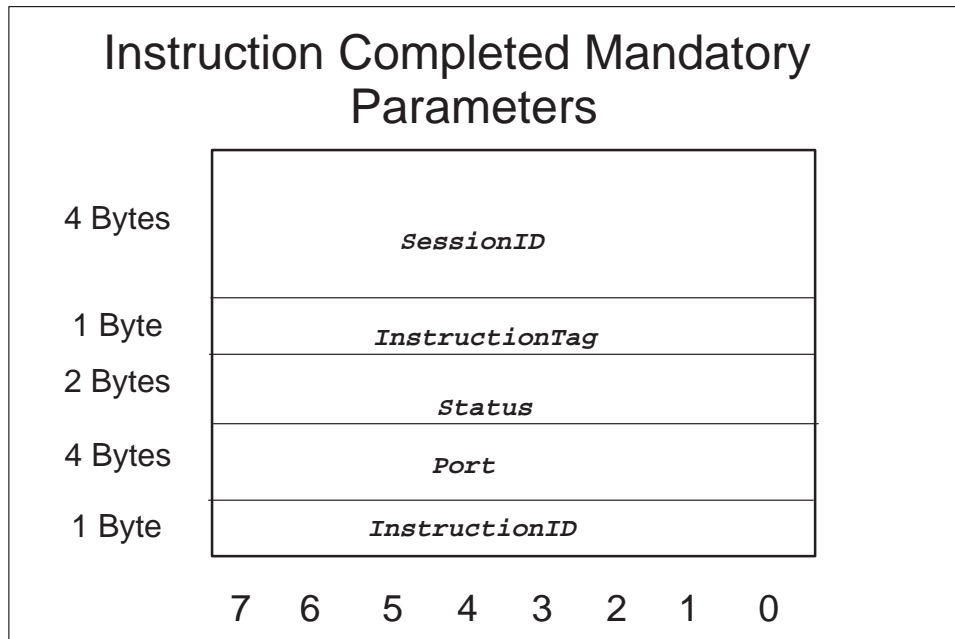
Parameter length: one byte

Parameter contents:

- The RESET REASON field consists of one byte and is the location which includes the TYPE: *ResetReason* parameter.

Instruction completed mandatory parameters

The mandatory parameters for a `Instruction_Completed` event are as follows:



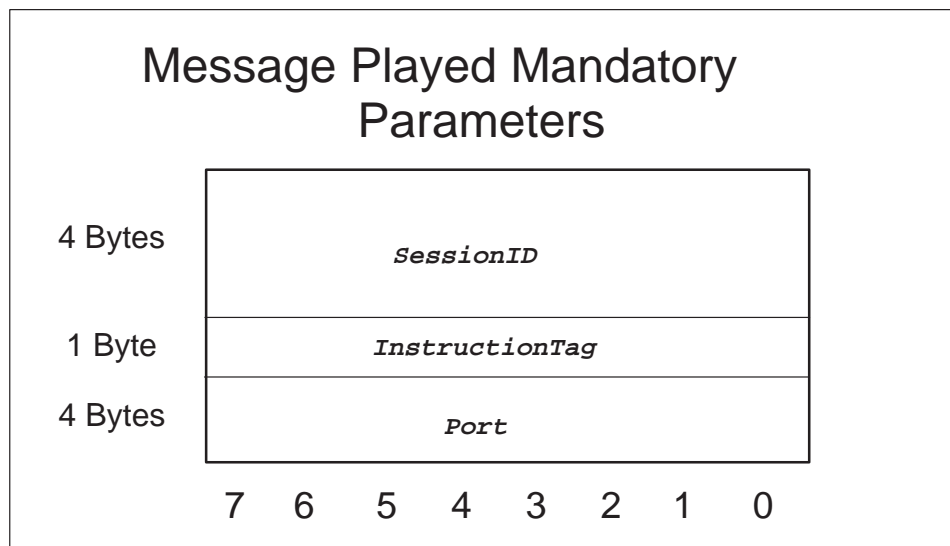
Parameter length: 12 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTag* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The INSTRUCTION ID field consists of one byte and is the location which includes the TYPE: *InstructionID* parameter.

Message played mandatory parameters

The mandatory parameters for a **Message_Played** Event are as follows:



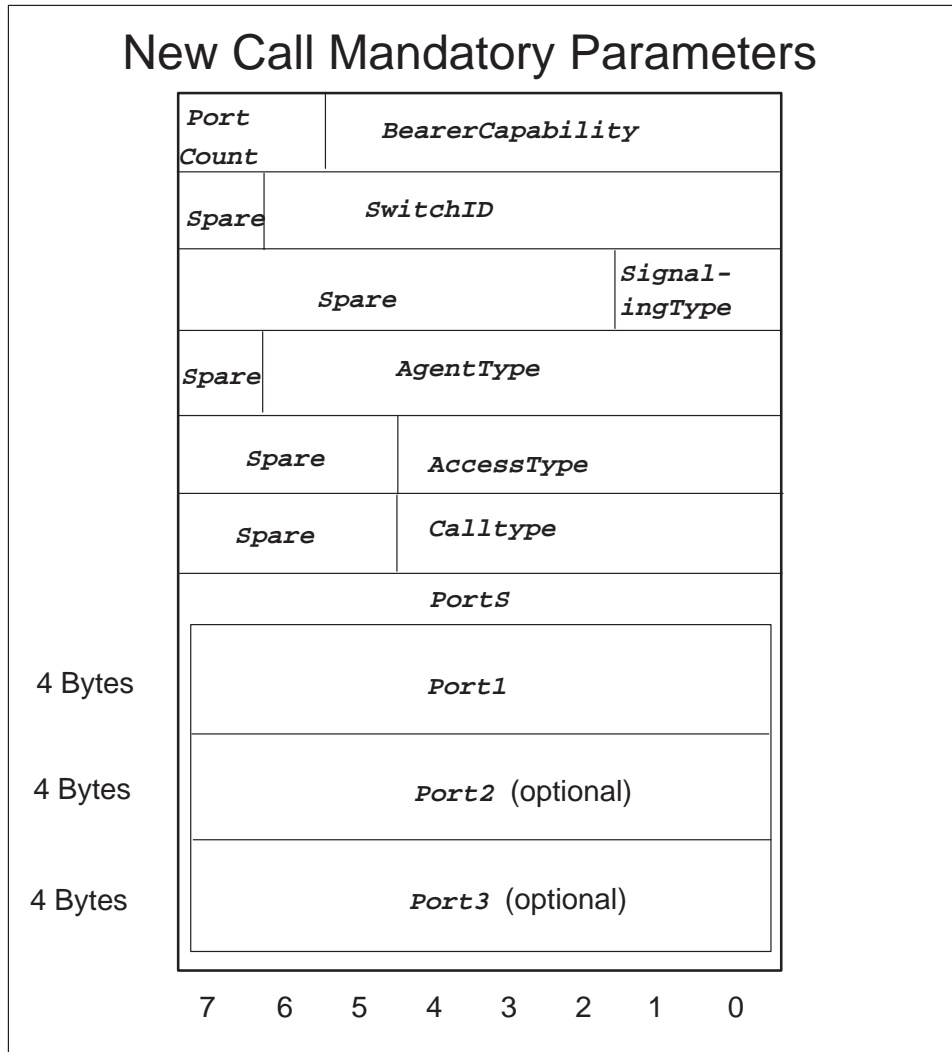
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

New call mandatory parameters

The mandatory parameters for a **New_Call** event are as follows:



Parameter length: minimum of 10 bytes and a maximum of 18 bytes

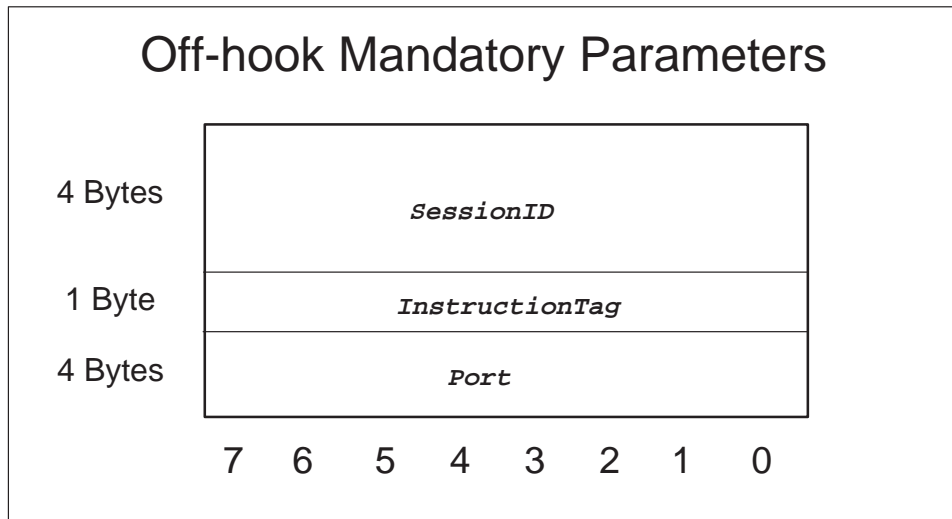
Parameter contents:

- The BEARER CAPABILITY field consists of six bits and is the location which includes the TYPE: *BearerCapability* parameter.
- The PORT COUNT field consists of two bits. The values include: 0 to 3.
- The SWITCH ID field consists of seven bits and is the location which includes the TYPE: *SwitchID* parameter.

- The SIGNALING TYPE field consists of one byte and is the location which includes the TYPE: *signalingType* parameter.
- The AGENT TYPE field consists of one byte and is the location which includes the TYPE: *AgentType* parameter.
- The ACCESS TYPE field consists of five bits and is the location which includes the TYPE: *AccessType* parameter.
- The CALL TYPE field consists of five bits and is the location which includes the TYPE: *Calltype* parameter.
- The PORTS consists of a minimum of 1 port to a maximum of 3 ports.
- The Port TYPE: *PortInfo* parameter.

Off-hook mandatory parameters

The mandatory parameters for an *Off_Hook* event are as follows:



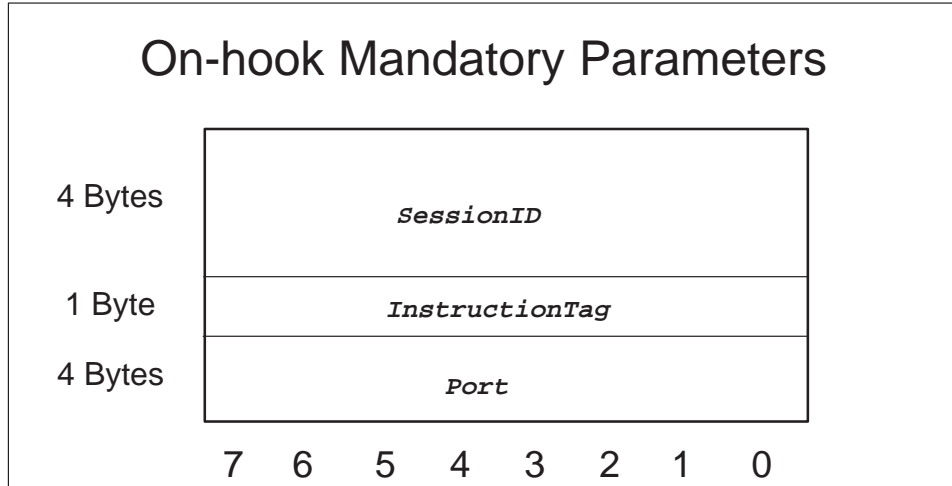
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

On-hook mandatory parameters

The mandatory parameters for an `On_Hook` Event are as follows:



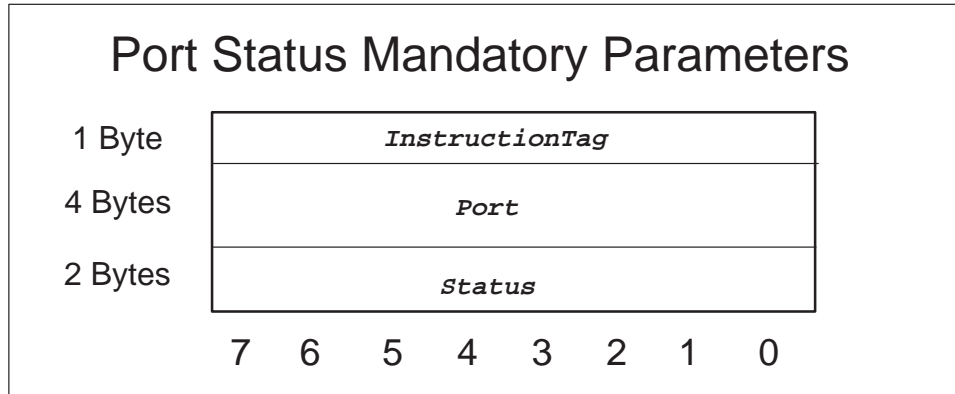
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Port status mandatory parameters

The mandatory parameters for a `Port_Status` event are as follows:



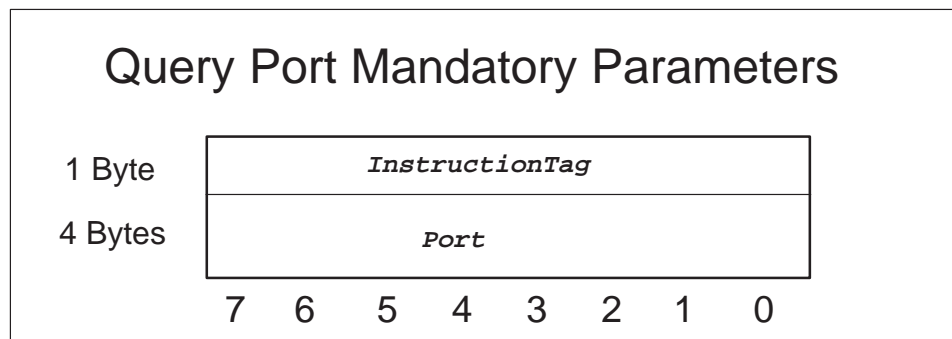
Parameter length: 7 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The STATUS field consists of two bytes and is the location which includes the TYPE: *PortStatus* parameter.

Query port mandatory parameters

The mandatory parameters for a `Query_Port` event are as follows:



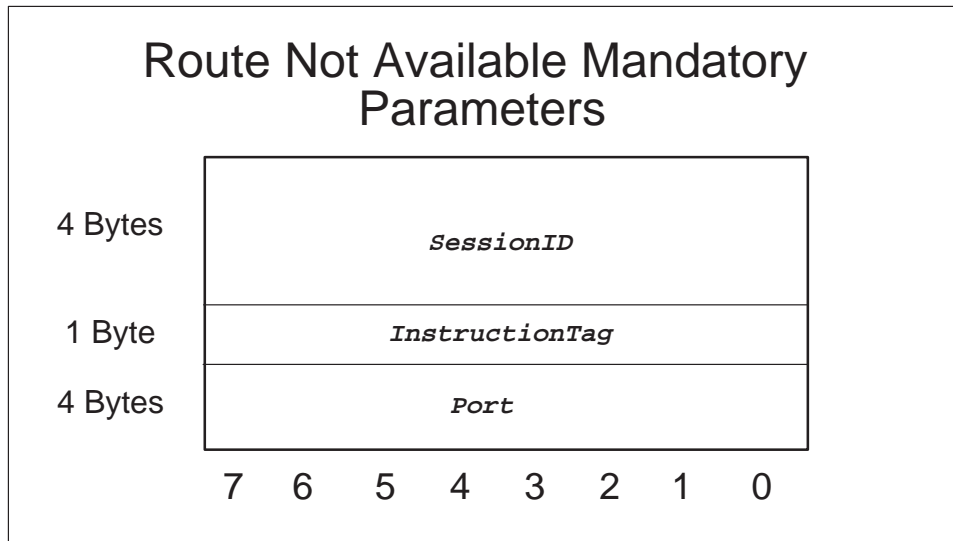
Parameter length: 5 bytes

Parameter contents:

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route not available mandatory parameters

The mandatory parameters for a *Route_Not_Available* Event are as follows:



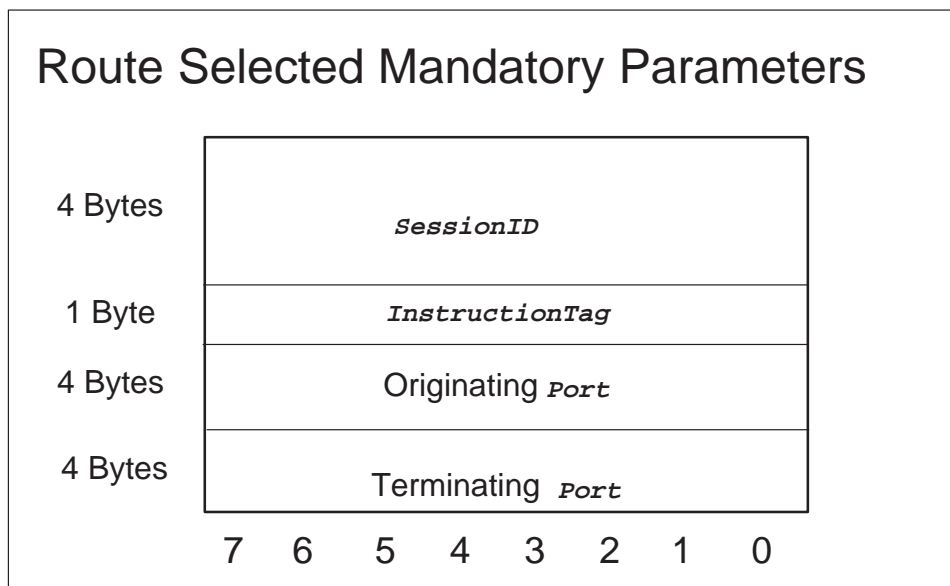
Parameter length: 9 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Route selected mandatory parameters

The mandatory parameters for a `Route_Selected` event are as follows:



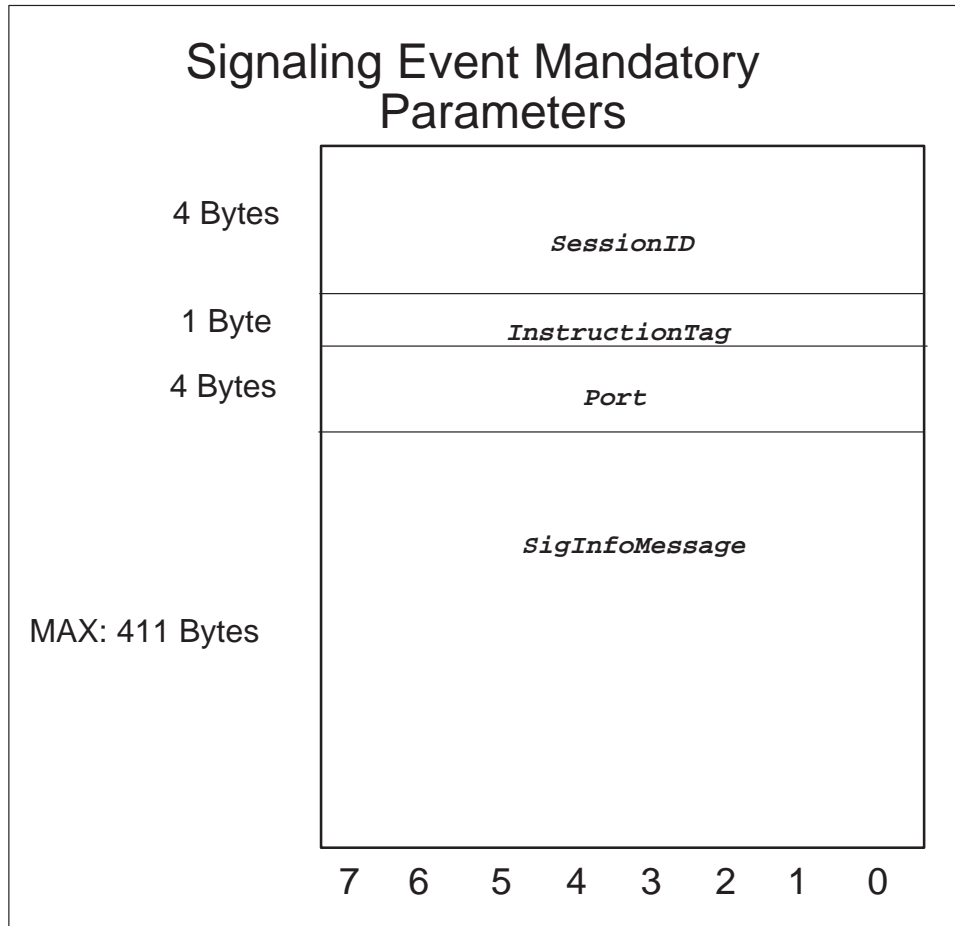
Parameter length: 13 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The ORIGINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TERMINATING PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.

Signaling event mandatory parameters

The mandatory parameters for a **signaling** Event are as follows:



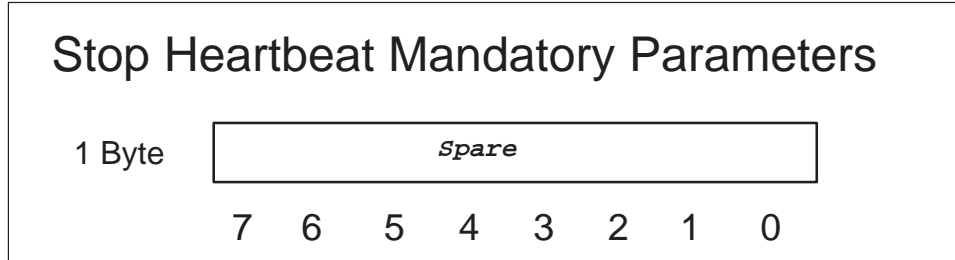
Parameter length: MAX 420 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *sessionID* parameter.
- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The SIGINFO MSG field consists of a maximum of 411 bytes and is the location which includes the TYPE: *signalingInfo* parameter.

Stop heartbeat mandatory parameters

The mandatory parameters for an `stop_Heartbeat` event are as follows:



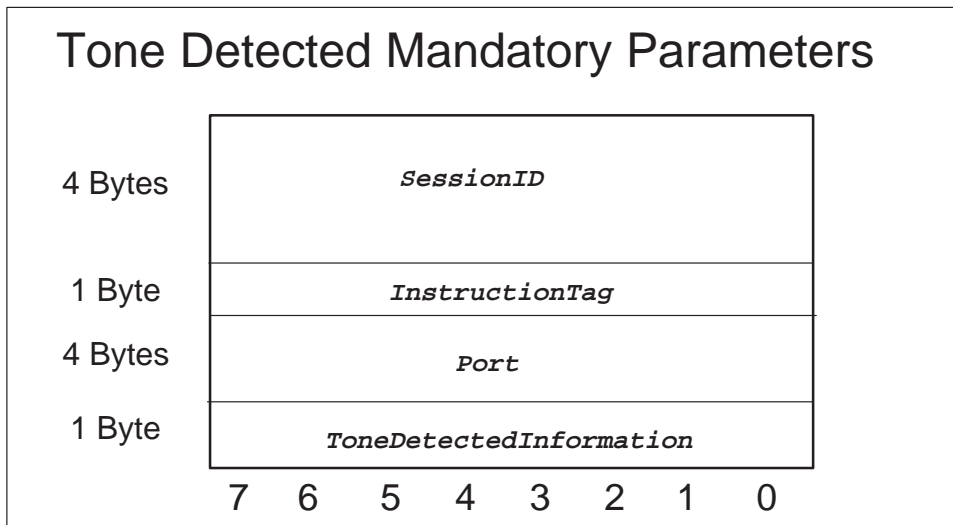
Parameter length: 1 byte

Parameter contents:

- The Spare as follows field consists of one byte and is the location which includes the To make message word aligned.

Tone detected mandatory parameters

The mandatory parameters for a `Tone_Detected` event are as follows:



Parameter length: 10 bytes

Parameter contents:

- The SESSION ID field consists of four bytes and is the location which includes the TYPE: *SessionID* parameter.

- The INSTRUCTION TAG field consists of one byte and is the location which includes the TYPE: *InstructionTagID* parameter.
- The PORT field consists of four bytes and is the location which includes the TYPE: *PortInfo* parameter.
- The TONE DETECTED INFORMATION field consists of one byte and is the location which includes the TYPE: *ToneDetected* parameter.

UCS PSN parameters version 4

This section of the document contains a detailed description of the UCS PSN Parameters Version 4. These PSN parameters may be a part of a PSN primitive and/or a PSN Event Notification. The use of the parameter depends upon the primitive/event notification that it is in.

PSN Peer To Peer Application Protocol

A new Peer to Peer Application protocol has been created for the PSN platform. The protocol defines the primitives that are provided to allow the SCU to control the calls on the PSN and the event notifications that are reported to the SCU from the PSN. The protocol also defines the parameters that go along with the primitives and event notifications.

The Peer to Peer protocol definition utilizes generic functional references to data rather than specific PSN data requirements. The protocol including the primitive/event notification definitions, parameter definitions and message flow (with service implementation examples) is described later in this document.

PSN Parameter Definitions

The SCU sends instructions via primitives to the PSN to control the service call, and the PSN notifies the SCU of the events that occur at the switch. The instructions and event notification messages contains one or more parameters with the appropriate information.

Each parameter has one or more bytes containing the parameter information. The division of parameter contents into fields and the encoding of these fields is given in detail in this section.

The primitives and/or the event notification messages which the parameter may be a part of is described earlier.

Tables 20-1 through 20-3 show the Primitives/Event Notifications and the associated parameters. Each parameter is marked with an “M” (for Mandatory), “O” (for Optional) or a “–” if the parameter is not associated with the primitive/event notification.

Table 20-1
Parameters for call control primitives

Parameters	B r i d g e	C o l l e c t D i g i t s & R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w C a l l A c c e p t e d	N e w C a l l R e j e c t e d	P l a y M e s s a g e	P P C D	R e c o n n e c t	S e t B i l l i n g R e c o r d	S t o p M e s s a g e	T r a n s m i t S i g I n f o
Access Type	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Agent Type	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Bearer Capability	-	-	O	-	-	-	-	-	-	-	-	-	-	-	-
Billing Info	O	O	O	O	O	O	O	O	O	O	O	O	M	O	O
Call Type	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
COT Required	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Control Info	O	O	O	-	O	O	O	O	-	O	O	O	O	O	O
Destination Trunk Group	-	-	M	-	-	-	-	-	-	-	-	-	-	-	-
<p>Note 1: When present, there is up to 3 of this parameter present in the primitive.</p> <p>Note 2: This parameter is Mandatory in a Connect primitive if the Signalling Info parameter is present and the signalling type of the agent is PTS.</p> <p>Note 3: This parameter, if present, takes the place of the Optional SigInfo Parameter.</p> <p>Note 4: This parameter is mandatory if the Signalling type of the port is PRI.</p>															
—continued—															

Table 20-1
Parameters for call control primitives (continued)

Parameters	B r i d g e	C o l l e c t D i g i t s & R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w C a l l A c c e p t e d	N e w C a l l R e j e c t e d	P l a y M e s s a g e	P P C D	R e c o n n e c t	S e t B i l l i n g R e c o r d	S t o p M e s s a g e	T r a n s m i t S i g I n f o
Digit Collection	-	M	-	-	-	-	-	-	-	-	M	-	-	-	-
Digits Collected	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Digits to Outputpulse Note 1	-	-	M Note 2	-	-	-	-	-	-	-	-	-	-	-	O Note 3
Digits Outputpulsed	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error Cause	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Flow Control Info	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<p>Note 1: When present, there is up to 3 of this parameter present in the primitive.</p> <p>Note 2: This parameter is Mandatory in a Connect primitive if the Signalling Info parameter is present and the signalling type of the agent is PTS.</p> <p>Note 3: This parameter, if present, takes the place of the Optional SigInfo Parameter.</p> <p>Note 4: This parameter is mandatory if the Signalling type of the port is PRI.</p>															
—continued—															

Table 20-1
Parameters for call control primitives (continued)

Parameters	B r i d g e	C o l l e c t D i g i t s & R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w C a l l A c c e p t e d	N e w C a l l R e j e c t e d	P l a y M e s s a g e	P P C D	R e c o n n e c t	S e t B i l l i n g R e c o r d	S t o p M e s s a g e	T r a n s m i t S i g I n f o
Flow Control Encountered	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Instruction ID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Instruction Tag	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M
ISUP Index	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Message Info	O	-	-	-	-	-	-	-	-	M	M	-	-	M	-
Monitor Mask	-	-	-	-	-	M	-	-	-	-	-	-	-	-	-
Parameter ID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Point In Call	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Port Info	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
<p>Note 1: When present, there is up to 3 of this parameter present in the primitive.</p> <p>Note 2: This parameter is Mandatory in a Connect primitive if the Signalling Info parameter is present and the signalling type of the agent is PTS.</p> <p>Note 3: This parameter, if present, takes the place of the Optional SigInfo Parameter.</p> <p>Note 4: This parameter is mandatory if the Signalling type of the port is PRI.</p>															
—continued—															

Table 20-1
Parameters for call control primitives (continued)

Parameters	B r i d g e	C o l l e c t D i g i t s & R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w C a l l A c c e p t e d	N e w C a l l R e j e c t e d	P l a y M e s s a g e	P P C D	R e c o n n e c t	S e t B i l l i n g R e c o r d	S t o p M e s s a g e	T r a n s m i t S i g I n f o
Port Service Info	O	O	O	-	O	O	O	M	-	O	O	O	O	O	O
Port Status	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset Reason	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Serving Translation Scheme	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Session ID	M	M	M	M	M	M	M	M	-	M	M	M	M	M	M
Signalling Info	-	-	M/O Note 4	O	-	-	-	-	-	-	-	-	-	-	M
Signalling Type	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<p>Note 1: When present, there is up to 3 of this parameter present in the primitive.</p> <p>Note 2: This parameter is Mandatory in a Connect primitive if the Signalling Info parameter is present and the signalling type of the agent is PTS.</p> <p>Note 3: This parameter, if present, takes the place of the Optional SigInfo Parameter.</p> <p>Note 4: This parameter is mandatory if the Signalling type of the port is PRI.</p>															
—continued—															

Table 20-1
Parameters for call control primitives (continued)

Parameters	B r i d g e	C o l l e c t D i g i t s & R e p o r t	C o n n e c t	D i s c o n n e c t	H o l d	M o n i t o r	M u t e	N e w C a l l A c c e p t e d	N e w C a l l R e j e c t e d	P l a y M e s s a g e	P P C D	R e c o n n e c t	S e t B i l l i n g R e c o r d	S t o p M e s s a g e	T r a n s m i t S i g I n f o
Switch ID	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Time of the Day	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Tone Detected	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<p>Note 1: When present, there is up to 3 of this parameter present in the primitive.</p> <p>Note 2: This parameter is Mandatory in a Connect primitive if the Signalling Info parameter is present and the signalling type of the agent is PTS.</p> <p>Note 3: This parameter, if present, takes the place of the Optional SigInfo Parameter.</p> <p>Note 4: This parameter is mandatory if the Signalling type of the port is PRI.</p>															
—end—															

Table 20-2
Parameters for event notifications

	D i g i t s C o l l e c t e d	E r r o r D e t e c t e d	I n s t r u c t i o n C o m p l e t e d	M e s s a g e P l a y e d	N e w C a l l	O f f - H o o k	O n - H o o k	R o u t e N o t A v a i l a b l e	R o u t e S e l e c t e d	S i g n a l l i n g E v e n t	T o n e D e t e c t e d
Parameters											
Access Type	-	-	-	-	M	-	-	-	-	-	-
Agent Type	-	-	-	-	M	-	-	-	-	-	-
Billing Info	-	-	-	-	-	-	-	-	-	-	-
Call Type	-	-	-	-	M	-	-	-	-	-	-
COT Required	-	-	-	-	O	-	-	-	-	-	-
Control Info	-	-	-	-	-	-	-	-	-	-	-
Digit Collection	-	-	-	-	-	-	-	-	-	-	-
Digits Collected	M	-	-	-	O	-	-	-	-	-	-
Digits to Outpulse	-	-	-	-	-	-	-	-	-	-	-
Digits Outpulsed	-	-	-	-	-	-	-	-	-	M	-
Error Cause	-	M	-	-	-	-	-	-	-	-	-
Flow Control Info	-	-	-	-	-	-	-	-	-	-	-
—continued—											

Table 20-2
Parameters for event notifications (continued)

Parameters	D i g i t s C o l l e c t e d	E r r o r D e t e c t e d	I n s t r u c t i o n C o m p l e t e d	M e s s a g e P l a y e d	N e w C a l l	O f f - H o o k	O n - H o o k	R o u t e N o t A v a i l a b l e	R o u t e S e l e c t e d	S i g n a l l i n g E v e n t	T o n e D e t e c t e d
Flow Control Encountered	-	-	-	-	O	-	-	-	-	-	-
Instruction ID	-	O	M	-	-	-	-	-	-	-	-
Instruction Tag	M	O	M	M	M	M	M	M	M	M	M
ISUP Index	-	-	-	-	O	-	-	-	-	-	-
Message Info	-	-	-	-	-	-	-	-	-	-	-
Monitor Mask	-	-	-	-	-	-	-	-	-	-	-
Parameter ID	-	O	-	-	-	-	-	-	-	-	-
Point In Call	-	-	-	-	-	-	-	-	-	-	-
Port Info	M	O	M	M	M	M	M	M	M	M	M
Port Service Info	-	-	-	-	-	-	-	-	-	-	-
Port Status	-	O	-	-	-	-	-	-	-	-	-
Reset Reason	-	-	-	-	-	-	-	-	-	-	-
Session ID	M	O	M	M	-	M	M	M	M	M	M
—continued—											

Table 20-2
Parameters for event notifications (continued)

	D i g i t s C o l l e c t e d	E r r o r D e t e c t e d	I n s t r u c t i o n C o m p l e t e d	M e s s a g e P l a y e d	N e w C a l l	O f f - H o o k	O n - H o o k	R o u t e N o t A v a i l a b l e	R o u t e S e l e c t e d	S i g n a l l i n g E v e n t	T o n e D e t e c t e d
Parameters											
Signalling Info	-	-	-	-	O	O	O	-	-	M	-
Signalling Type	-	-	-	-	M	-	-	-	-	-	-
Switch ID	-	-	-	-	M	-	-	-	-	-	-
Time of the Day	-	-	-	-	-	-	-	-	-	-	-
Tone Detected	-	-	-	-	-	-	-	-	-	-	M
—end—											

Table 20-3
Parameters for non-call related events and primitives

Parameters	A g e n t D a t a	C u r r e n t T i m e o f D a y	F l o w C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t S t a t u s	Q u e r y P o r t	Q u e r y T i m e o f D a y	R e s e t S w i t c h	S e t I P A d d r e s s	S t o p H e a r t b e a t
Access Type	-	-	-	-	-	-	-	-	-	-	-
Agent Data Info	M	-	-	-	-	-	-	-	-	-	-
Agent Type	-	-	-	-	-	-	-	-	-	-	-
Billing Info	-	-	-	-	-	-	-	-	-	-	-
Call Type	-	-	-	-	-	-	-	-	-	-	-
Info Change Reason	M	-	-	-	-	-	-	-	-	-	-
COT Required	-	-	-	-	-	-	-	-	-	-	-
Control Info	-	-	-	-	-	-	-	-	-	-	-
Destination Trunk Group	-	-	-	-	-	-	-	-	-	-	-
Digit Collection	-	-	-	-	-	-	-	-	-	-	-
Digits Collected	-	-	-	-	-	-	-	-	-	-	-
<p>Note: There is 1 Mandatory PSI Parameter called the Return IP Address parameter which is used to send the Instruction Completed. There are 0 to 20 Optional PSI Parameters called the IP Address to Reset parameter(s) which are used to do the actual resetting of calls.</p>											
<p>—continued—</p>											

Table 20-3
Parameters for non-call related events and primitives (continued)

Parameters	A g e n t D a t a	C u r r e n t T i m e o f D a y	F l o w C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t S t a t u s	Q u e r y P o r t	Q u e r y T i m e o f D a y	R e s e t S w i t c h	S e t I P A d d r e s s	S t o p H e a r t b e a t
Digits to Output	-	-	-	-	-	-	-	-	-	-	-
Digits Outputted	-	-	-	-	-	-	-	-	-	-	-
Error Cause	-	-	-	-	-	-	-	-	-	-	-
Flow Control Info	-	-	O	-	-	-	-	-	-	-	-
Flow Control Encountered	-	-	-	-	-	-	-	-	-	-	-
Instruction ID	-	-	-	-	-	-	-	-	-	-	-
Instruction Tag	-	M	M	-	-	M	M	M	M	M	-
ISUP Index	-	-	-	-	-	-	-	-	-	-	-
Message Info	-	-	-	-	-	-	-	-	-	-	-
Monitor Mask	-	-	-	-	-	-	-	-	-	-	-
Parameter ID	-	-	-	-	-	-	-	-	-	-	-
Point In Call	-	-	-	-	-	-	-	-	-	-	-
<p>Note: There is 1 Mandatory PSI Parameter called the Return IP Address parameter which is used to send the Instruction Completed. There are 0 to 20 Optional PSI Parameters called the IP Address to Reset parameter(s) which are used to do the actual resetting of calls.</p>											
—continued—											

Table 20-3
Parameters for non-call related events and primitives (continued)

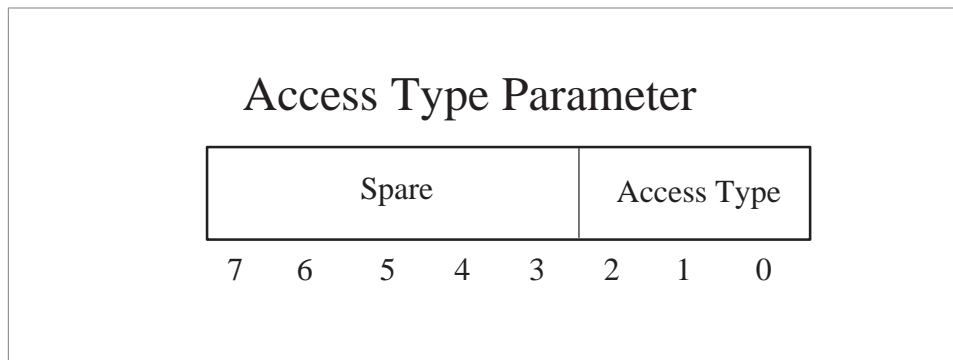
Parameters	A g e n t D a t a	C u r r e n t T i m e o f D a y	F l o w C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t S t a t u s	Q u e r y P o r t	Q u e r y T i m e o f D a y	R e s e t S w i t c h	S e t I P A d d r e s s	S t o p H e a r t b e a t
Port Info	-	-	-	-	-	M	M	-	-	M	-
Port Service Info	-	-	-	-	-	-	-	-	M and O Note	M	-
Port Status	-	-	-	-	-	M	-	-	-	-	-
Reset Reason	-	-	-	M	-	-	-	-	-	-	-
Session ID	-	-	-	-	-	O	O	-	-	-	-
Signalling Info	-	-	-	-	-	-	-	-	-	-	-
Signalling Type	-	-	-	-	-	-	-	-	-	-	-
Signinfo Mask	-	-	-	-	-	-	-	-	-	-	-
Switch ID	M	-	-	-	-	-	-	-	-	-	-
Time of the Day	-	M	-	-	-	-	-	-	-	-	-
<p>Note: There is 1 Mandatory PSI Parameter called the Return IP Address parameter which is used to send the Instruction Completed. There are 0 to 20 Optional PSI Parameters called the IP Address to Reset parameter(s) which are used to do the actual resetting of calls.</p>											
<p>—continued—</p>											

Table 20-3
Parameters for non-call related events and primitives (continued)

Parameters	A g e n t D a t a	C u r r e n t T i m e o f D a y	F l o w C o n t r o l	I n S e r v i c e	H e a r t b e a t	P o r t S t a t u s	Q u e r y P o r t	Q u e r y T i m e o f D a y	R e s e t S w i t c h	S e t I P A d d r e s s	S t o p H e a r t b e a t
Tone Detected	-	-	-	-	-	-	-	-	-	-	-
<p>Note: There is 1 Mandatory PSI Parameter called the Return IP Address parameter which is used to send the Instruction Completed. There are 0 to 20 Optional PSI Parameters called the IP Address to Reset parameter(s) which are used to do the actual resetting of calls.</p>											
—end—											

Access Type

This information is sent in the NEW CALL event notification and contains the originating trunk information. The only values supported for this parameter are FGD, DAL, and PRI.



Mandatory Parameter for a New Call event.

Parameter Length: 1 Byte

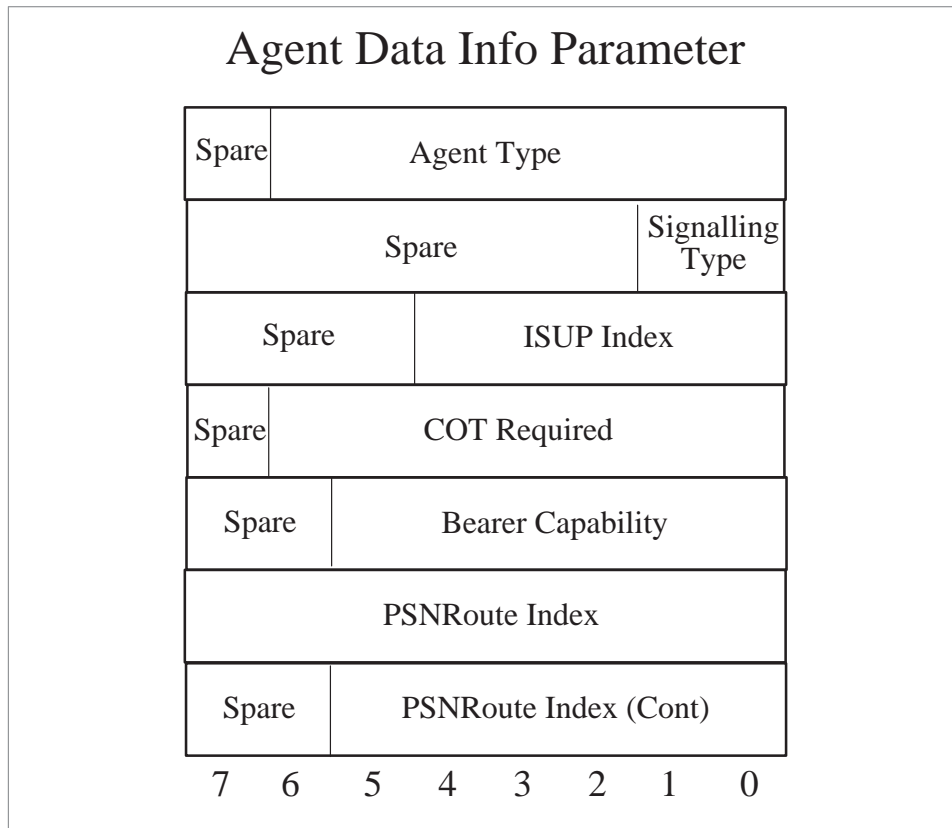
Parameter Contents:

ACCESS TYPE : 3 Bits – The access type of the port, which is encoded as follows:

- 0 0 0 : Unused (Spare)
- 1 0 0 1 : DAL 2–Wire
- 2 0 1 0 : DAL 4–Wire
- 3 0 1 1 : PTS FGD
- 4 1 0 0 : SS7 FGD
- 5 1 0 1 : PTS IMT
- 6 1 1 0 : SS7 IMT
- 7 1 1 1 : PRI

Agent Data Info

This parameter is only used in the Agent Data Event and the New Call Event.



Length: 7 Bytes

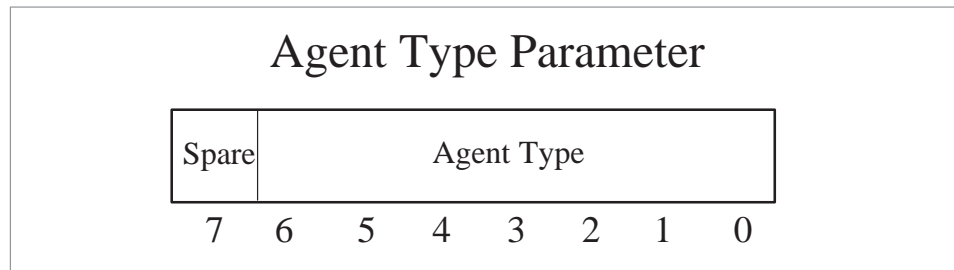
Parameter Contents:

- *AGENT TYPE*: 7 Bits – *TYPE: Agent Type Parameter*
- *SIGNALLING TYPE* : 2 Bits – *TYPE: Signalling Type Parameter*
- *ISUP INDEX*: 5 Bits – *TYPE: ISUP Index Parameter*
- *COT REQUIRED PERCENT*: 7 Bits – *TYPE: COT Required Parameter*
- *BEARER CAPABILITY*: 6 Bits – *TYPE: Bearer Capability Parameter*
- *PSNROUTE INDEX*: 14 Bits – This is the actual index into table PSNROUTE.

Note: This index is 0 when an Agent Data event occurs informing the SCU of a change in only the Switch ID. If the New Switch ID and Current Switch ID differ, all other fields of the Agent Data Info Parameter should be ignored. At all other times (when no Switch ID change occurs), this field has a value from 1 to 9999. When this occurs, all of the Agent Data Info Parameter fields may be considered valid.

Agent Type Parameter

This parameter is used in the Agent Data Event and in the New Call Event.



Optional Parameter ID: 29

Length: 1 Byte

Parameter Contents:

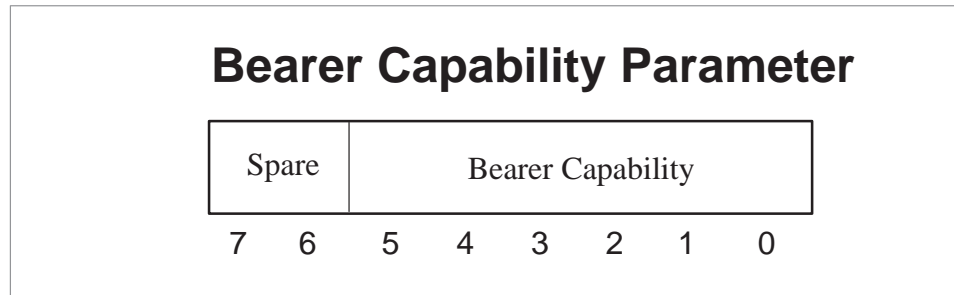
- *AGENT TYPE*: 7 Bits – This field allows the SCU to associate a generic trunk naming convention for each agent in table PSNROUTE. Values include:

0	0 0 0 0 0 0 0 : Unused
1	0 0 0 0 0 0 1 : DAL Trunk
2	0 0 0 0 0 1 0 : Unused
3	0 0 0 0 0 1 1 : ONAT Trunk
4	0 0 0 0 1 0 0 : EANT Trunk
	0 0 0 0 1 0 1 to 0 0 0 0 1 1 0 : Unused
7	0 0 0 0 1 1 1 : IMT Trunk

- 8 0001000 : Unused
- 9 0001001 : OP250 Trunk
0001010 to 0001100 : Unused
- 13 0001101 : PRA250 Trunk
0001110 to 1111111 : Unused

Bearer Capability parameter

This parameter contains the Bearer Capability information.



Optional Parameter ID: 1

Parameter Length: 6 Bits

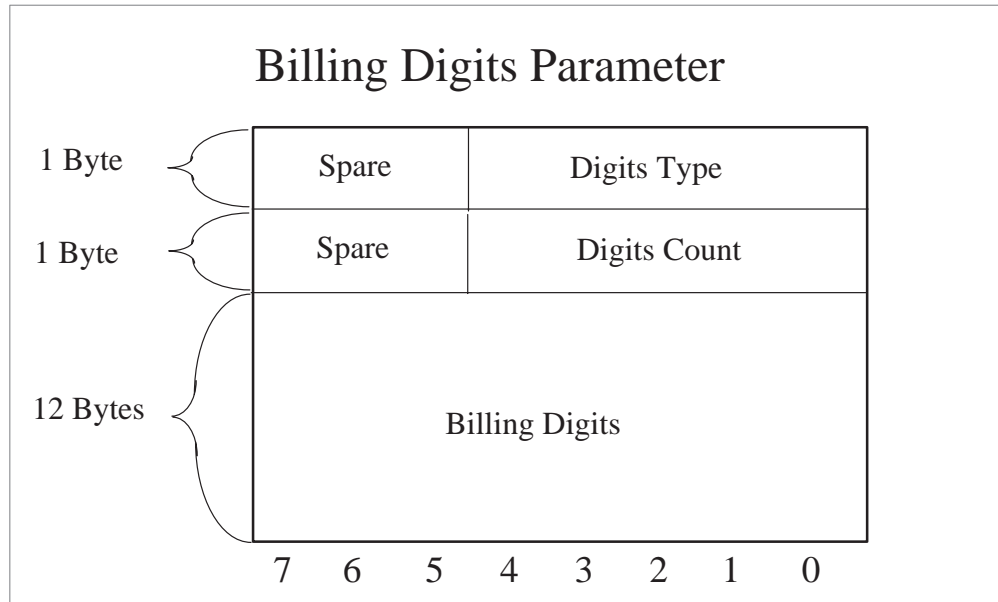
Parameter Contents:

- **BEARER CAPABILITY** : 6 Bits – The bearer capability field indicates the speed and type of information that is to be carried by the call. Values include:

- 0 000000 : Not Used
- 1 000001 : Speech
- 2 000010 : 64K Data
- 3 000011 : 64K X25
- 4 000100 : 56K Data
- 5 000101 : Data Unit
- 6 000110 : 64K Restricted
- 7 000111 : 3.1 kHz
- 8 001000 : 7 kHz
- 9 001001 : Voice Data
- 10 001010 : 64K Rate Data
- 11 001011 : 32K Speech
- 12 001100 : WideBand
- 001101 to 111111 : Spare Values

Billing Digits Parameter

This parameter contains the supported billing digits types, count of digits and the actual digits. Billing Info Parameter can have a max of 17 billing digits parameters.



Optional Parameter ID: 33

Parameter Length: 14 Bytes

Parameter Contents:

- **DIGITS TYPE** : 5 Bits – The digits type indicates how to interpret the billing info contents. Billing Info Parameter supports the following 17 digits types:
 - 2 0 0 0 1 0 : Calling Party Address Digits(ANI)
 - 5 0 0 1 0 1 : Billing Number Digits
 - 7 0 0 1 1 1 : Call Duration Digits
 - 8 0 1 0 0 0 : Carrier Identification Digits
 - 9 0 1 0 0 1 : Infodig Digits Number
 - 10 0 1 0 1 0 : ANISP Digits Number
 - 11 0 1 0 1 1 : Origgrp Digits
 - 12 0 1 1 0 0 : Origmem Digits
 - 14 0 1 1 1 0 : Personal Identification Number (PIN)
 - 16 1 0 0 0 0 : Call Reference ID (CRID) Digits Number
 - 18 1 0 0 1 0 : Termvsn Digits Number
 - 19 1 0 0 1 1 : Univacc Digits Number
 - 21 1 0 1 0 1 : DNIS Digits Number
 - 23 1 0 1 1 1 : Account Code Number

24 1 1 0 0 0 : Termgrp Digits
25 1 1 0 0 1 : Termmem Digits
26 1 1 0 1 0 : Dialed Digits Number

- DIGITS COUNT : 5 Bits – Every Digits Types has a maximum number of digits. Here are the max number of digits each supported Digits type can have:

Acctcode Digits	a max of 12 Digits
Anisp Digits	a max of 10 Digits
Billnum Digits	a max of 23 Digits
Call Duration Digits	a max of 5 Digits
Carrierid Digits	a max of 4 Digits
Calling Ptyaddr Digits	a max of 15 Digits
CRID Digits	a max of 9 Digits
Dialedno Digits	a max of 18 Digits
DNIS Digits	a max of 15 Digits
Infodig Digits	a max of 2 Digits
Origgrp Digits	a max of 4 Digits
Origmem Digits	a max of 4 Digits
PIN Digits	a max of 4 Digits
Termgrp Digits	a max of 4 Digits
Termmem Digits	a max of 4 Digits
Termpvn Digits	a max of 15 Digits
Univacc Digits	a max of 10 Digits

Note: The max value the SCU can send to populate the Call Duration CDR field is 65499 Seconds which equals 18.19 Hours, which gets displayed in the CDR fields in 10 ms increments. For example the value 123 becomes 1. It gets rounded up if last 2 digits are 50 or above and to round down if 49 or less. For example, 9550 gets displayed as 00000096 and 9549 gets displayed as 00000095.

- BILLING DIGITS : 12 Bytes – A table to hold the actual billing digits.

- **GENERATE CDR** : 1 Bit – This is a BOOL Bit to allow the SCU to decide whether to generate an RU (Recording Unit) and populate some CDR fields, or to drop the RU and stop writing to any CDR fields. The value of this bit must equal to one if billing info parameter is mandatory (Set Billing Info Primitive).

0 0 : FALSE -> Drop RU
 1 1 : TRUE -> Generate RU

- **NUMBER OF BILLING INFO** : 1 Byte – The number of billing info sub-parameters to follow. Each billing info sub-parameter consists of the billing info type and the billing info contents. The maximum number of Billing Info in Billing Info Parameter is one.

0 0 0 0 0 0 0 0 : (Number of Billing Infos = 0)
 0 0 0 0 0 0 0 1

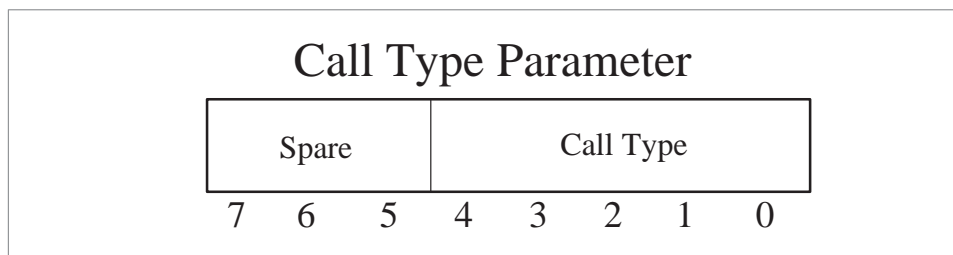
 0 0 0 1 0 0 0 1 : (Number of Billing Infos = 17)
 0 0 0 1 0 0 1 0 to 1 1 1 1 1 1 1 1 : Unused (Spare)

- **BILLING DIGITS PARAMETER** : 14 Bytes – Billing Info Parameter allows up to 17 Billing Digits parameters to be sent. Billing Digits parameter has three fields:

- Digits Type
- Digits Count
- Billing Digits

Call Type parameter

This parameter contains the call type for the call when it is determined by the PSN that the call is to be controlled by the SCU.



Mandatory Parameter for a New Call event.

Parameter Length: 1 Byte

Parameter Contents:

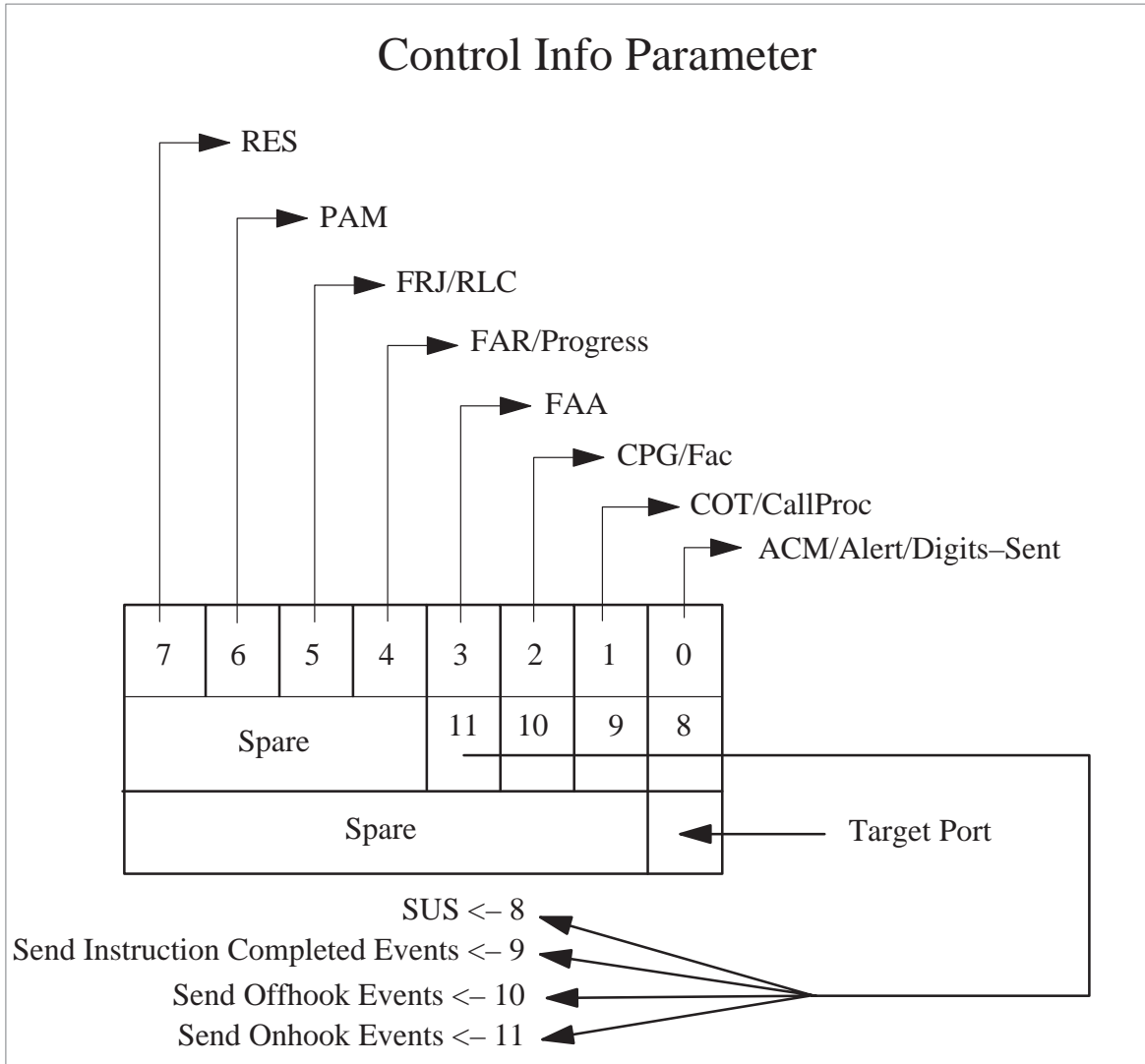
- CALL TYPE : 5 Bits – The call type gives information on the type of call being made. The values defined currently include:

0	0 0 0 0 0	: Undetermined *
1	0 0 0 0 1	: Onnet *
2	0 0 0 1 0	: Offnet *
3	0 0 0 1 1	: Public Speed
4	0 0 1 0 0	: Private Speed
5	0 0 1 0 1	: Hotline Speed
6	0 0 1 1 0	: N00*
7	0 0 1 1 1	: Zero Plus – Onnet
8	0 1 0 0 0	: Zero Plus – Offnet
9	0 1 0 0 1	: INTOA *
	0 1 0 1 0 to 1 1 1 1 1	: Spare Values

Note: Only the values marked with an “*” are currently supported.

Control Info Parameter

This parameter contains the action to be taken if the primitive is successfully processed.



Optional Parameter ID: 4

Parameter Length: 3Bytes

Parameter Contents:

The first nine bits are the contents of what used to be Siginfo Mask Parameter:

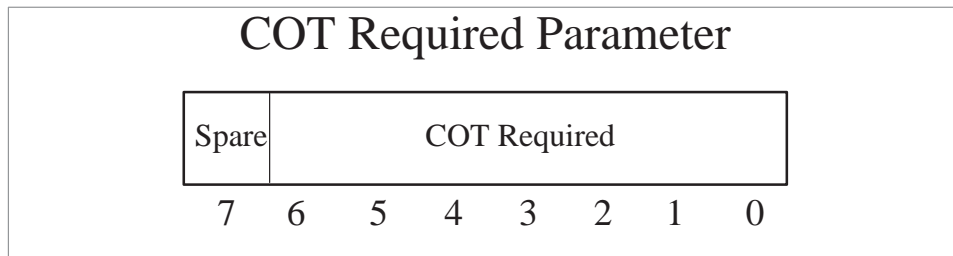
- ACM/Alert : 1 Bit

- Values : 0 = do not send ACM (for SS7 port) or Alert (for PRI port) message to the SCU.
1 = send the ACM (for SS7 port) or Alert (for PRI port) to the SCU.
- COT/CallProc : 1 Bit
Values : 0 = do not send COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU.
1 = send the COT (for SS7 port) or Call Proceeding (for PRI port) message to the SCU.
 - CPG/FAC : 1 Bit
Values : 0 = do not send CPG (for SS7 port) or FAC (for PRI port) message to the SCU.
1 = send the CPG (for SS7 port) or FAC (for PRI port) to the SCU.
 - FAA : 1 Bit
Values : 0 = do not send FAA (for SS7 port) message to the SCU.
1 = send the FAA (for SS7 port) to the SCU.
 - FAR/Progress : 1 Bit
Values : 0 = do not send FAR (for SS7 port) or Progress (for PRI port) message to the SCU.
1 = send the FAR (for SS7 port) or Progress (for PRI port) to the SCU.
 - FRJ/RLC : 1 Bit
Values : 0 = do not send FRJ (for SS7 port) or RLC (for PRI port) message to the SCU.
1 = send the FRJ (for SS7 port) or RLC (for PRI port) to the SCU.
 - PAM
Values : 0 = do not send PAM (for the SS7 port) to the SCU.
1 = send the PAM (recvd. on the SS7 port) to the SCU.
 - RES : 1 Bit
Values : 0 = do not send RES (for the SS7 port) to the SCU.
1 = send the RES (recvd. on the SS7 port) to the SCU.
 - SUS : 1 Bit
Values : 0 = do not send SUS (for the SS7 port) to the SCU.
1 = send the SUS (recvd. on the SS7 port) to the SCU.
 - SEND ONHOOK EVENTS : 1 Bit – This field is a boolean which indicates that the agent should or should not send any Onhook events that occur on the agent. Values include:
 - 0 : OFF – Do not send any Onhook events.
 - 1 : ON – Send all Onhook events.

- **SEND OFFHOOK EVENTS** : 1 Bit – This field is a boolean which indicates that the agent should or should not send any Offhook events that occur on the agent. Values include:
 - 0 : OFF – Do not send any Offhook events.
 - 1 : ON – Send all Offhook events.
- **SEND INSTRUCTION COMPLETED EVENTS** : 1 Bit – This field is a boolean which indicates that the agent should or should not send any Instruction Completed events that occur on the agent. Values include:
 - 0 : OFF – Do not send any Instruction Completed events.
 - 1 : ON – Send all Instruction Completed events.
- **TARGET PORT** : 1 Byte – Since some primitives (i.e. Connect) involves more than one port in a call, the Target Port bit indicates to what port a primitive is being sent to.
 - 0 0 : Port_A This is the default value.
 - 1 1 : Port_B Valid only when used with the Connect and/or Reconnect.
 - 0 0 0 0 0 1 0 to 1 1 1 1 1 1 1 : Spare Values

COT Required Parameter

This parameter is used in the Agent Data Event and in the New Call Event.



Optional Parameter ID: 30

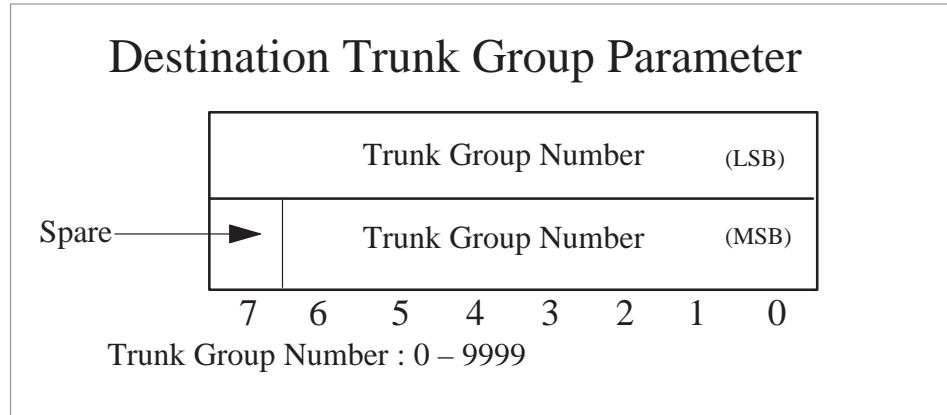
Length: 1 Byte

Parameter Contents:

- **COT REQUIRED PERCENT**: 7 Bits – This field represents the percentage of this SS7 PSN agent’s trunk members are datafilled to have COT testing performed on them. The range of this parameter is 0 (for no COT testing) to 100 (for COT testing on each member).

Destination Trunk Group parameter

This parameter contains the external trunk group number of the trunk to route to. The call terminates to an idle member of this trunk group.



Optional Parameter ID: 5

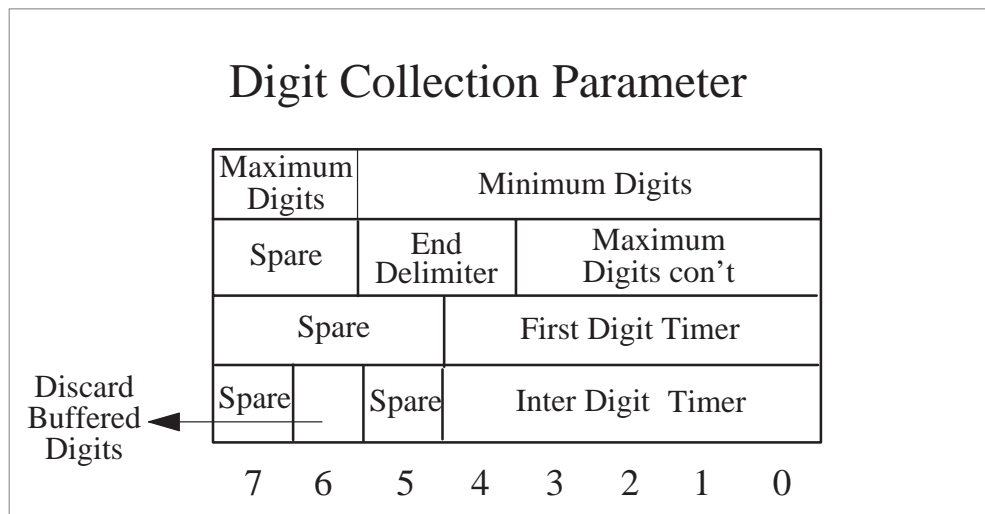
Parameter Length: 2 Bytes

Parameter Contents:

- **TRUNK GROUP NUMBER** : 15 Bits – This is the external trunk group number. The valid range is 0 to 9999. The `NIL_TRUNK_GROUP` number is defined as 32767.

Digit Collection Parameter

This parameter contains information required to collect digits on a specified port.



Optional Parameter ID: 6

Parameter Length: 4 bytes

Parameter Contents:

- **MINIMUM_DIGITS** : 6 Bits – This field contains the minimum number of digits to collect. This value must be less than or equal to the **MAXIMUM_DIGITS** field. Values : 0 – 45.
0 0 0 0 0 0 : (Minimum number of digits to collect = 0)
0 0 0 0 0 1
.....
1 0 1 1 0 1 : (Minimum number of digits to collect = 45)
1 0 1 1 1 0 to 1 1 1 1 1 1 : Unused (Spare)
- **MAXIMUM_DIGITS** : 6 Bits – This field contains the maximum number of digits to collect. Values : 0 – 45.
0 0 0 0 0 0 : (Maximum number of digits to collect = 0)
0 0 0 0 0 1
.....
1 0 1 1 0 1 : (Maximum number of digits to collect = 45)
1 0 1 1 1 0 to 1 1 1 1 1 1 : Unused (Spare)
- **END_DELIMITER** : 2 Bits – This field identifies the end of digits delimiter; upon detection of the end of digits delimiter digit collection stops and the collected digits are reported. If the delimiter is dialed as the very first digit then the digit collection stops immediately. The delimiter digit values include:
0 0 : no delimiter specified
0 1 : *
1 0 : #
1 1 : * and #
- **FIRST_DIGIT_TIMER**: 5 Bits – This field specifies the time to wait until the first digit is dialed. Valid Values : 2 – 30 seconds.
0 0 0 0 0 : Unused (Spare)
0 0 0 0 1 : Unused (Spare)
0 0 0 1 0 : Timer Value = 2 second
.....
1 1 1 1 0 : Timer Value = 30 seconds
1 1 1 1 1 : Unused (Spare)
- **INTER_DIGIT_TIMER**: 5 Bits – This field specifies the inter digit timer value, i.e., the time to wait when collecting the second and subsequent digits. Values : 2 – 20 seconds.

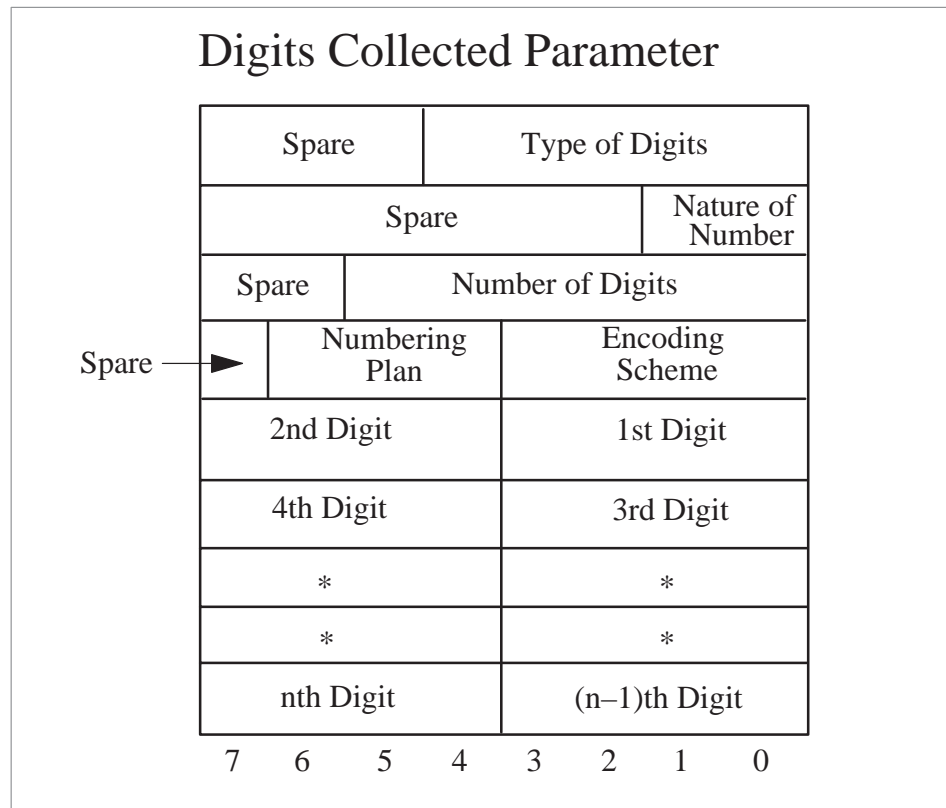
0 0 0 0 0 : Unused (Spare)
 0 0 0 0 1 : Unused (Spare)
 0 0 0 1 0 : Timer Value = 2 second

 1 0 1 0 0 : Timer Value = 20 seconds
 1 0 1 0 1 to 1 1 1 1 1 : Unused (Spare)

- **DISCARD BUFFERED DIGITS** : 1 Bit – This field is a bool which if true indicates that if the PSN had been buffering any digits prior to receiving this parameter, then it is to discard it and start digit collection all over again. Please refer to section AD8716:PSNFMSM for details on buffering of digits.

Digits Collected parameter

This parameter contains the digits collected or received on the port/agent. The digits contained in this parameter are always in the TBCD format. Also, included is COUNT which specifies the number of digits included in this parameter.



Optional Parameter ID: 7

Parameter Length: Variable in size

Parameter Contents:

Note: The contents of this parameter (including the Nature of Number, Numbering Plan and Encoding Scheme fields have been encoded as per the following documents: TR–NWT–000317 Switching Systems Requirements for Call Control Using ISDNUP, TR–NWT–000394 Switching Systems Requirements for IEC Interconnection using ISDNUP and TR–NWT–000444 Switching System Requirements Supporting ISDN Access Using the ISDNUP. However, there are some values marked “Private” which are application specific.

- **TYPE OF DIGITS : 5 Bits**– This field contains the type of digits encoded as given below. The type of digits is known when the collected digits are sent in the *New Call event notification*. When this parameter is sent in the *Digits Collected event notification*, a value of “Unknown” is used.

0	0 0 0 0 0	: Unknown
1	0 0 0 0 1	: Called Party Address
2	0 0 0 1 0	: Calling Party Address (ANI)
3	0 0 0 1 1	: Caller Interaction
4	0 0 1 0 0	: Routing Number
5	0 0 1 0 1	: Billing Number
6	0 0 1 1 0	: Destination Number
7	0 0 1 1 1	: Local Access and Transport Area (LATA)
8	0 1 0 0 0	: Carrier Identification
9	0 1 0 0 1	: Referral Number (Private)
10	0 1 0 1 0	: True Billing Number (Private)
11	0 1 0 1 1	: Alternate Preferred Carrier (Private)
12	0 1 1 0 0	: Preferred INC (Private)
13	0 1 1 0 1	: Primary Preferred Carrier (Private)
14	0 1 1 1 0	: Personal Identification Number (PIN)
15	0 1 1 1 1	: Authorization Code (Private)
16	1 0 0 0 0	: TCM (Private)
17	1 0 0 0 1	: Second Alternate Preferred Carrier (Private)
18	1 0 0 1 0	: Business Customer ID (Private)
19	1 0 0 1 1	: Hop–off Office (Private)
20	1 0 1 0 0	: Outpulse Number (Private)
21	1 0 1 0 1	: Originating Station (DN)
22	1 0 1 1 0	: MCCS Card Number
23	1 0 1 1 1	: Account Code Number
24	1 1 0 0 0	: COSOVE Number
25	1 1 0 0 1	: Generic Digits Number
26	1 1 0 1 0	: Dialed Digits Number
27	1 1 0 1 1	: Facility Code
28	1 1 1 0 0	: Country Code
29	1 1 1 0 1	: STS Digits
30	1 1 1 1 0	: OPart Digits

31 1 1 1 1 1 : TPart Digits

- NATURE OF NUMBER : 2 Bits – This field is encoded as follows:
 - 0 0 0 : Not Applicable
 - 1 0 1 : International
 - 2 1 0 : National
 - 3 1 1 : Network Specific

- NUMBER OF DIGITS : 6 Bits – This field contains the number of digits that are sent in this parameter. This number may be as low as 0 and as high as 45.

- ENCODING SCHEME : 4 Bits – contains the scheme used to encode the digits that are sent in this parameter and it is encoded as follows:
 - 0 0 0 0 : Unknown
 - 1 0 0 1 : Binary Coded Decimal (BCD)
 - 0 0 1 0 to 1 1 0 1 : Spare Values
 - 14 1 1 1 0 : Telephony Binary Coded Decimal (TBCD)
 - 1 1 1 1 : Spare

- NUMBERING PLAN : 3 Bits – is encoded as follows:
 - 0 0 0 0 : Unknown or Not Applicable
 - 1 0 0 1 : ISDN Numbering Plan (E.164)
 - 2 0 1 0 : Telephony Numbering Plan (E.163)
 - 3 0 1 1 : Data Numbering Plan (X.121)
 - 4 1 0 0 : Telex Numbering Plan (F.69)
 - 5 1 0 1 : Maritime Mobile Numbering Plan (E.120,211)
 - 6 1 1 0 : Land Mobile Numbering Plan (E.212, 213)
 - 1 1 1 : Spare Value

- 1ST DIGIT to NTH DIGIT : Each digit is 4 Bits – This field contains n digits. These DTMF digits may be encoded in the BCD format as follows:
 - 0 0 0 0 0 : Digit 0
 - 1 0 0 0 1 : Digit 1
 - 2 0 0 1 0 : Digit 2
 - 3 0 0 1 1 : Digit 3
 - 4 0 1 0 0 : Digit 4
 - 5 0 1 0 1 : Digit 5
 - 6 0 1 1 0 : Digit 6
 - 7 0 1 1 1 : Digit 7
 - 8 1 0 0 0 : Digit 8
 - 9 1 0 0 1 : Digit 9
 - 10 1 0 1 0 : Filler
 - 11 1 0 1 1 : *

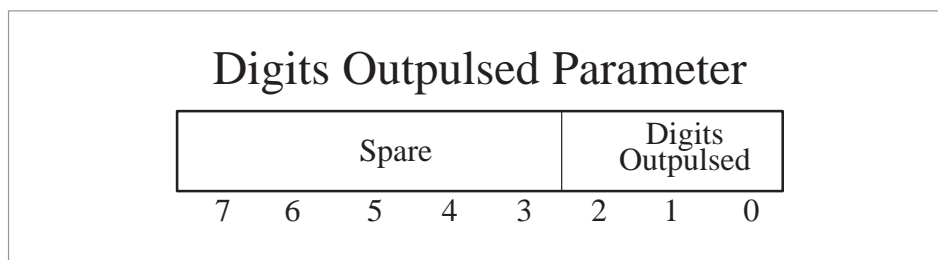
12 1 1 0 0 : #
13 1 1 0 1 : D
14 1 1 1 0 : E
15 1 1 1 1 : F

- Or the DTMF digits may be encoded in the TBCD format as follows:

0 0 0 0 0 : Filler
1 0 0 0 1 : Digit 1
2 0 0 1 0 : Digit 2
3 0 0 1 1 : Digit 3
4 0 1 0 0 : Digit 4
5 0 1 0 1 : Digit 5
6 0 1 1 0 : Digit 6
7 0 1 1 1 : Digit 7
8 1 0 0 0 : Digit 8
9 1 0 0 1 : Digit 9
10 1 0 1 0 : Digit 0
11 1 0 1 1 : *
12 1 1 0 0 : #
13 1 1 0 1 : D
14 1 1 1 0 : E
15 1 1 1 1 : F

Digits Outpulsed parameter

This parameter indicates information on the digits that were outpulsed on a PTS trunk agent. This is especially useful in multi-stage outpulsing. This parameter is included in the Signalling Event notification message to let the SCU know that all the digits were outpulsed on the trunk agency.



Optional Parameter ID: 8

Parameter Length: 1 Byte

Parameter Contents:

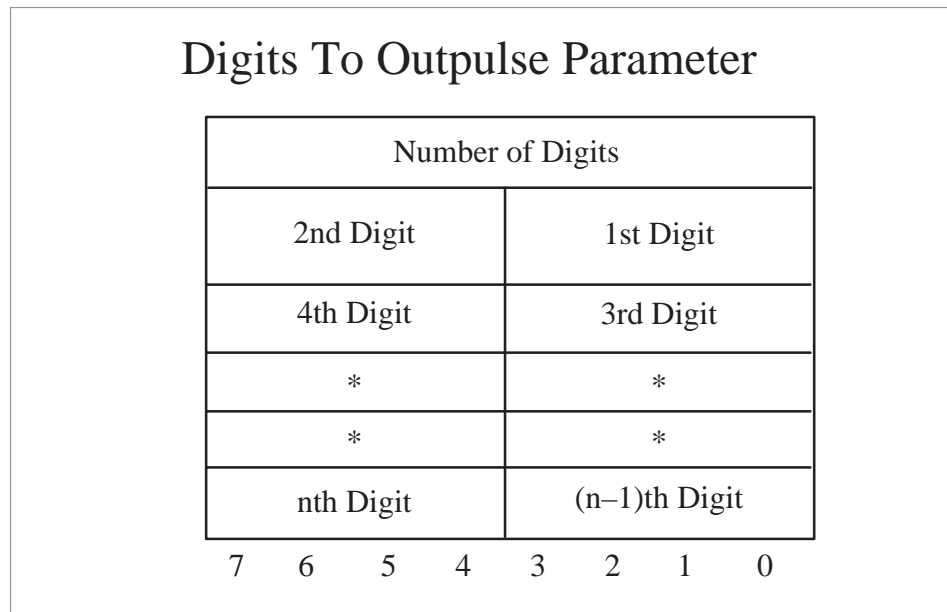
- DIGITS OUTPULSED : 3 Bits – This field is encoded as follows:

0 0 0 : Unknown

0 0 1 : all streams outpulsed
 0 1 0 to 1 1 1 : Spare

Digits To Outpulse parameter

This parameter contains the digits to be outpulsed on a PTS trunk agent.



Optional Parameter ID: 9

Parameter Length: Variable in size

Parameter Contents:

- **NUMBER OF DIGITS** : 1 Byte – This field contains the number of digits that are to be outpulsed on this port.
 The maximum number of digits that may be outpulsed is 23. The valid range for this field is 1 – 23.
- **1ST DIGIT to NTH DIGIT** : Each digit is 4 Bits – This field contains the n digits.

DTMF digits are encoded as follows:

0 0 0 0 0 : Filler
 1 0 0 0 1 : Digit 1
 2 0 0 1 0 : Digit 2
 3 0 0 1 1 : Digit 3
 4 0 1 0 0 : Digit 4
 5 0 1 0 1 : Digit 5
 6 0 1 1 0 : Digit 6

7 0 1 1 1 : Digit 7
8 1 0 0 0 : Digit 8
9 1 0 0 1 : Digit 9
10 1 0 1 0 : Digit 0
11 1 0 1 1 : *
12 1 1 0 0 : #
13 1 1 0 1 : D
14 1 1 1 0 : E
1 1 1 1 : F

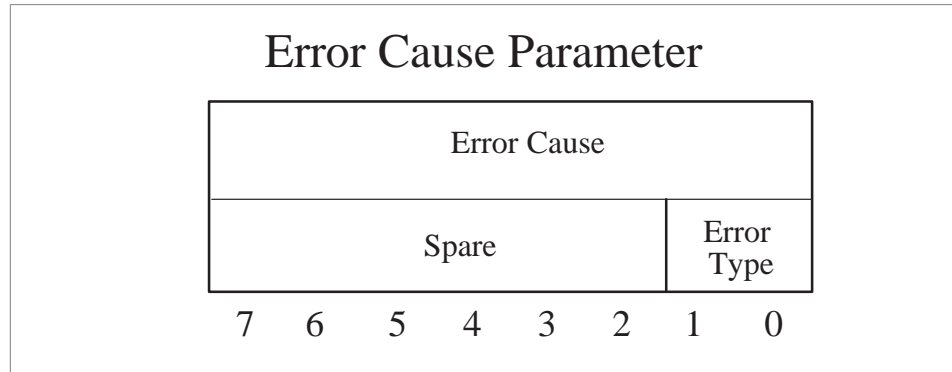
MF digits are encoded as follows:

0 0 0 0 0 : Filler
1 0 0 0 1 : Digit 1
2 0 0 1 0 : Digit 2
3 0 0 1 1 : Digit 3
4 0 1 0 0 : Digit 4
5 0 1 0 1 : Digit 5
6 0 1 1 0 : Digit 6
7 0 1 1 1 : Digit 7
8 1 0 0 0 : Digit 8
9 1 0 0 1 : Digit 9
10 1 0 1 0 : Digit 0
11 1 0 1 1 : KP3 and ST3P
12 1 1 0 0 : KPP and STP
13 1 1 0 1 : KP and STKP
14 1 1 1 0 : KP2 and ST2P
1 1 1 1 : ST

If multiple “Digits To Output” parameters are contained in a message, then multiple streams of digits are outputted on the agent/port – each stream contained in one parameter of type “Digits to Output.”

Error Cause parameter

This parameter contains the cause of an error that is detected on the PSN.



Optional Parameter ID: 10

Parameter Length: 2 Bytes

Parameter Contents:

- **ERROR CAUSE** : 1 Byte – This field is used to represent the cause of the error. Values include:
 - 0 00000000 : Nil Error Cause
 - 1 00000001 : Header decode failure
 - 2 00000010 : Bad macro tag
 - 3 00000011 : Unrecognized primitive
 - 4 00000100 : Missing mandatory parameter
 - 5 00000101 : Mandatory parameter decode failure
 - 6 00000110 : Optional parameter decode failure
 - 7 00000111 : Parameter contents out of range
 - 8 00001000 : Primitive userclass mismatch
 - 9 00001001 : Maximum primitive exceeded
 - 10 00001010 : Missing mandatory Signfo parameter
 - 11 00001011 : One or more agents in the primitive are not PSN agents
 - 12 00001100 : Port not in table PSNROUTE
 - 13 00001101 : Agent not supported
 - 14 00001110 : Port down due to WARM restart
 - 15 00001111 : Primitive invalid for current port state
 - 16 00010000 : Unexpected message
 - 17 00010001 : STR not available (affects the Monitor primitive)
 - 18 00010010 : UTR not available
 - 19 00010011 : Conference circuit not available
 - 20 00010100 : No IDLE message

21	0 0 0 1 0 1 0 1	: CCB not available
22	0 0 0 1 0 1 1 0	: Primitive extension block not available
23	0 0 0 1 0 1 1 1	: Scratchpad extension block not available
24	0 0 0 1 1 0 0 0	: Software Resources unavailable
25	0 0 0 1 1 0 0 1	: Message failure
26	0 0 0 1 1 0 1 0	: Software error
27	0 0 0 1 1 0 1 1	: Not minimum number ports to Bridge
28	0 0 0 1 1 1 0 0	: Maximum ports to Bridge exceeded
29	0 0 0 1 1 1 0 1	: Bearer capability incompatible
30	0 0 0 1 1 1 1 0	: Message index not in table PSNMSGIX
31	0 0 0 1 1 1 1 1	: Unsupported signalling type
32	0 0 0 1 0 0 0 0	: Duplicate message
33	0 0 1 0 0 0 0 1	: Bad agent state
34	0 0 1 0 0 0 1 0	: Termination failure
35	0 0 1 0 0 0 1 1	: Abnormal Exit
36	0 0 1 0 0 1 0 0	: Message not playing
37	0 0 1 0 0 1 0 1	: Tone duration unsupported
38	0 0 1 0 0 1 1 0	: Prompt failure
39	0 0 1 0 0 1 1 1	: Digit collection failure
40	0 0 1 0 1 0 0 0	: Q764 protocol problem
41	0 0 1 0 1 0 0 1	: Invalid duration gap
42	0 0 1 0 1 0 1 0	: Unexpected FC message
43	0 0 1 0 1 0 1 1	: Agent not in Table TRKGRP
44	0 0 1 0 1 1 0 0	: BBF Not Possible

Note: The value “BBF Not Possible” is used to indicate that the BBF is not possible for this port based on its current agent/connection state. A port must be connected to an agent and must be monitored for BBF. If it is held or bridged, BBF monitoring for the agent is not done.

- **ERROR TYPE : 2 Bits** – This field contains the type of the error that is detected, and is encoded as follows:
 - 0 0 : Non Fatal Error
 - 0 1 : Fatal Error

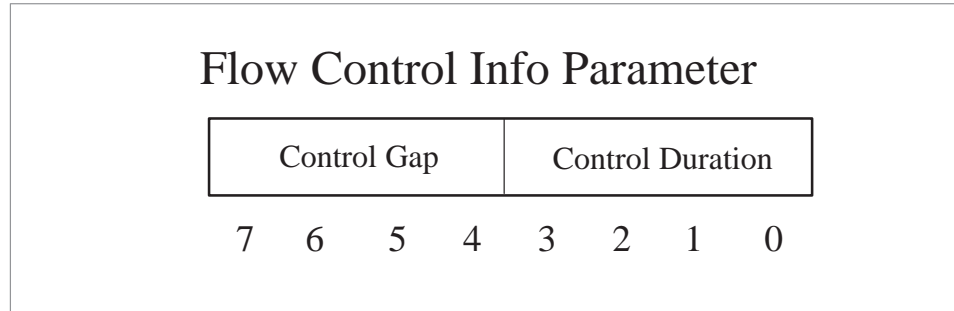
Fatal errors are defined as errors that are so severe that they do not allow normal call processing to proceed on this port.

If the error that is detected is non-fatal, then the PSN does not take any action other than report this error to the SCU. If the error that is detected is fatal, then the PSN reports the error to the SCU, and in addition to this takes down the associated port.

Flow Control Info Parameter

This parameter consists of control Duration which specifies the maximum amount of time the flow control is effective at the PSN and control Gap

which specifies the maximum rate at which the New Call event notifications may be sent to the SCU while the control is effective.



Optional Parameter ID: 11

Parameter Length: 1 Byte

Parameter Contents:

- **CONTROL DURATION:** 4 Bits – Maximum amount of time that the flow control is effective at the PSN. The control duration is encoded as follows:

0	0 0 0 0	: Not Used
1	0 0 0 1	: 1 Second
2	0 0 1 0	: 2 Seconds
3	0 0 1 1	: 4 Seconds
4	0 1 0 0	: 8 Seconds
5	0 1 0 1	: 16 Seconds
6	0 1 1 0	: 32 Seconds
7	0 1 1 1	: 64 Seconds
8	1 0 0 0	: 128 Seconds
9	1 0 0 1	: 256 Seconds
10	1 0 1 0	: 512 Seconds
11	1 0 1 1	: 1024 Seconds
12	1 1 0 0	: 2048 Seconds
	1 1 0 1 to 1 1 1 1	: Spare Values

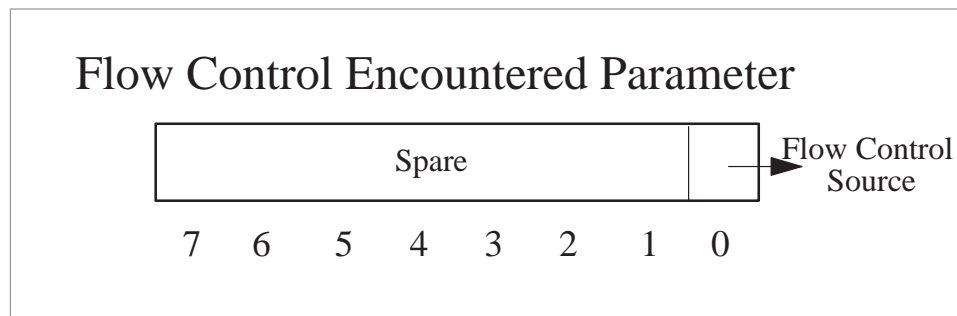
- **CONTROL GAP:** 4 Bits – Maximum rate at which the New Call Event Notifications may be sent to the SCU while the flow control is effective. The control gap is encoded as follows:

0	0 0 0 0	: Remove Gap Control
1	0 0 0 1	: 1 1/10th of a Second
2	0 0 1 0	: 3 1/10ths of a Second
3	0 0 1 1	: 5 1/10ths of a Second
4	0 1 0 0	: 1 Second
5	0 1 0 1	: 2 Seconds

6	0 1 1 0	: 5 Seconds
7	0 1 1 1	: 10 Seconds
8	1 0 0 0	: 15 Seconds
9	1 0 0 1	: 30 Seconds
10	1 0 1 0	: 50 Seconds
11	1 0 1 1	: 80 Seconds
12	1 1 0 0	: 120 Seconds
13	1 1 0 1	: 300 Seconds
14	1 1 1 0	: 600 Seconds
15	1 1 1 1	: Stop All Calls

Flow Control Encountered Parameter

This parameter is sent in New Call events sent to the SCU while the flow control is active. This parameter informs the SCU that the flow control is active and it identifies the source that initiated the flow control.



Optional Parameter ID: 12

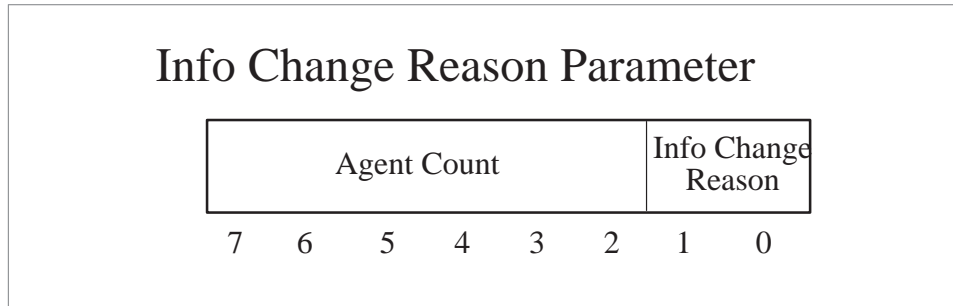
Parameter Length: 1 Byte

Parameter Contents:

- **FLOW CONTROL SOURCE:** 1 Bit – The source that initiated the flow control. It is encoded as follows:
 - 0 : Flow control initiated by the SCU
 - 1 : Flow control initiated by the PSN

Info Change Reason Parameter

This parm contains the Info Change Reason and Agent Count information used in Agent Data event notifications.



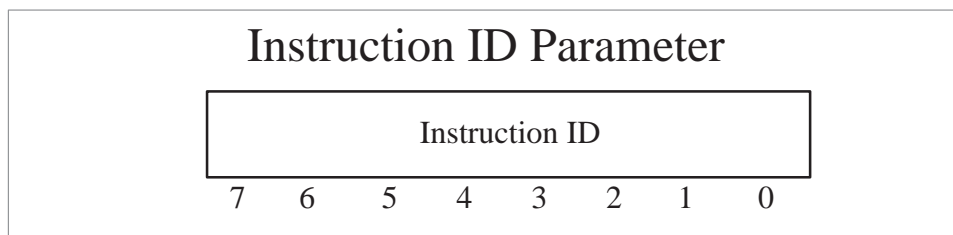
Parameter Length: 1 Byte

Parameter Contents:

- **INFO CHANGE REASON** : 2 Bits – The info change reason field indicates what type of Agent Data Event is occurring for the given message. Values include:
 - 0 0 : Agent Deleted in table PSNROUTE
 - 0 1 : Agent Added in table PSNROUTE
 - 1 0 : Agent information Modified in table TRKGRP or TRKSGRP
- **AGENT COUNT** : 6 Bits – The agent count field indicates the number of Agent Data Info parameters that exist in the given Agent Data event message. Valid values are:
 - 0 0 0 0 0 1 to 1 1 1 1 1 1

Instruction ID parameter

This parameter contains the Identifier of the Primitive that is received from the SCU.



Optional Parameter ID: 13

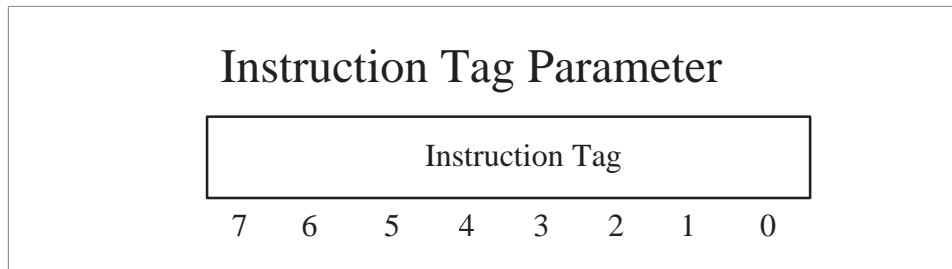
Parameter Length: 1 Byte

Parameter Contents:

- **INSTRUCTION ID** : 1 Byte – This field is a byte that is used to represent the primitive instruction . Values include:
 - 0 00000000 : Unknown
 - 1 00000001 : Bridge
 - 2 00000010 : Collect Digits & Report
 - 3 00000011 : Connect
 - 4 00000100 : Disconnect
 - 5 00000101 : Hold
 - 6 00000110 : Monitor
 - 7 00000111 : Mute
 - 8 00001000 : New Call Accepted
 - 9 00001001 : New Call Rejected
 - 10 00001010 : Play Message
 - 11 00001011 : Play Prompt, Collect Digits & Report
 - 12 00001100 : Query Port
 - 13 00001101 : Reconnect
 - 14 00001110 : Reset Switch
 - 15 00001111 : Set Billing Record
 - 16 00010000 : Set IP Address
 - 17 00010001 : Stop Message
 - 18 00010010 : Transmit Signinfo
 - 19 00010011 : Heartbeat
 - 20 00010100 : Query Time of Day
 - 21 00010101 : Error Detected
 - 22 00010110 : Port Status
 - 23 00010111 : Flow Control
 - 00011000 to 11111111 : Spare Values

Instruction Tag parameter

This parameter contains the tag associated with the instruction.



- For primitives that are sent from the SCU, the SCU generates an Instruction tag and sends it to the PSN in this parameter. When a response for this primitive is generated by the PSN, the PSN includes this tag in the response so that the SCU may correlate the response with the primitive request that it had sent earlier.
- For asynchronous event notifications generated by the PSN (e.g., On-Hook, Off-Hook and Signalling Event), the Instruction tag is hard-coded to #01.
- For instructions that are generated by the PSN that require a reply from the SCU (that is, New Call, Query Port from the Audit application), the PSN sends a nil Instruction tag.

Optional Parameter ID: 14

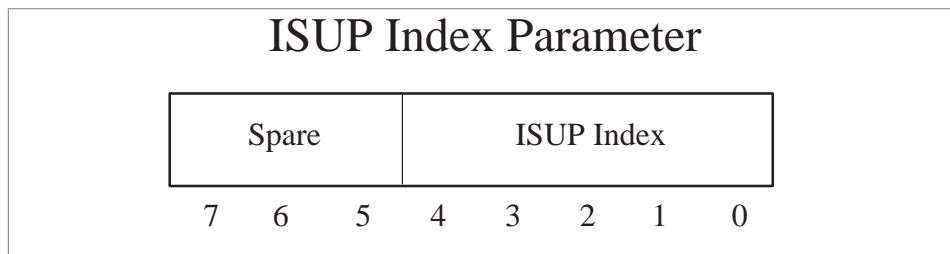
Parameter Length: 1 Byte

Parameter Contents:

- INSTRUCTION TAG : 1 Byte – This field is a byte that is used to represent the tag of the instruction that is received from or sent to the SCU. Values include:
 - 0 0 0 0 0 0 0 0 : Nil Instruction Tag
 - 0 0 0 0 0 0 0 1 : Asynchronous Event Notification from the PSN.
 - 0 0 0 0 0 0 0 1 to 1 1 1 1 1 1 1 1 : Valid tags for instructions from the SCU.

ISUP Index parameter

This parameter is used in the Agent Data Event and in the New Call Event.



Optional Parameter ID: 31

Length: 1 Byte

Parameter Contents:

- ISUP INDEX: 5 Bits – This field represents the type of facility this SS7 agent is connect to. This field is only valid for SS7 signalling types. Values include:

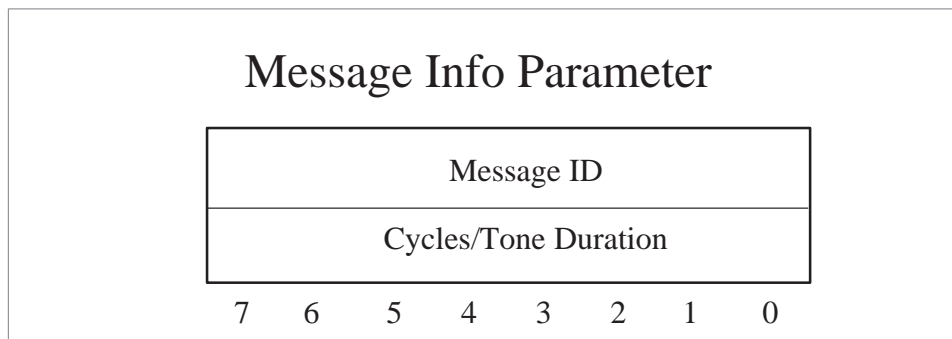
```

0  0 0 0 0 : NILIDX
1  0 0 0 1 : UCS2UCS
2  0 0 0 1 0 : UCS2DEX8
3  0 0 0 1 1 : UCS2EAE0
4  0 0 1 0 0 : UCS2USP
5  0 0 1 0 1 : UCS2MCI
6  0 0 1 1 0 : UCSGITU
7  0 0 1 1 1 : USP2USP
8  0 1 0 0 0 : USP2UCS
9  0 1 0 0 1 : UCSGWAY
   0 1 0 1 0 to 1 1 1 1 1 : Unused

```

Message Info parameter

This parameter contains the Message ID and the Cycles/Tone Duration information.



The Message ID is used to index into table PSNMSGIX, to get either an index into table ANNS (if the message ID corresponds to an announcement), or table TONES (if the message ID corresponds to a tone).

Note: Please refer to Chapter 10, “PSN office parameters” for details on table PSNMSGIX.

Optional Parameter ID: 15

Parameter Length: 2 Bytes

Parameter Contents:

- MESSAGE ID : 1 Byte – Valid Values include 1 to 255. The value is an index into table PSNMSGIX. A value of 0 is invalid.
- CYCLES : 1 Byte – The 2nd byte of the parameter contents is treated as the number of cycles to play the announcement for, if the message ID corresponds to an announcement. Values include:

0 0 0 0 0 0 0 0 : Play the Announcement indefinitely
0 0 0 0 0 0 0 1 : Play the Announcement for 1 cycle
.....
0 0 0 1 1 1 1 0 : Play the Announcement for 30 cycles
0 0 0 1 1 1 1 1 to 1 1 1 1 1 1 1 1 : Play the Announcement
indefinitely

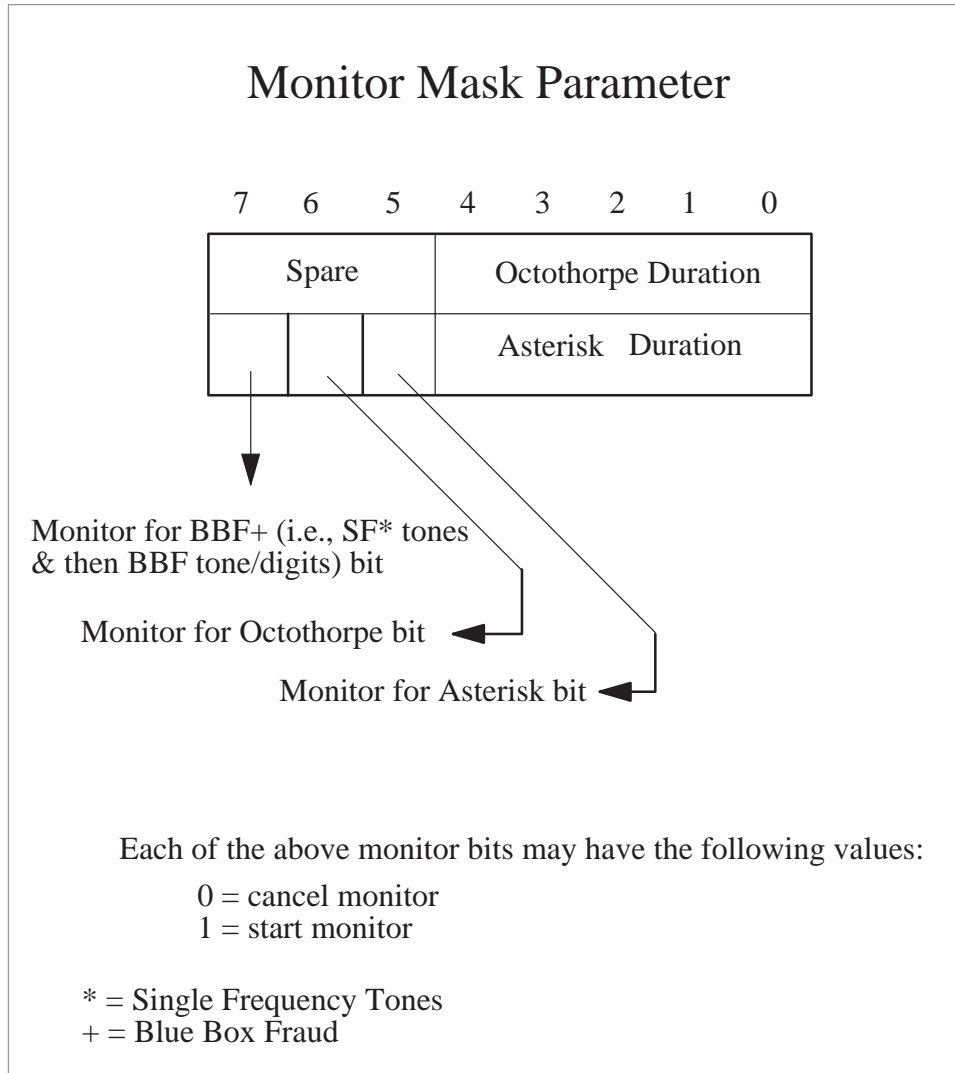
TONE DURATION : 1 Byte – The 2nd byte of the parameter contents is treated as the tone duration value if the message ID corresponds to a tone. Valid Values include:

0 0 0 0 0 0 0 0 : Play the Tone forever
0 0 0 0 0 0 0 1 : Invalid
0 0 0 0 0 0 1 0 : Invalid
0 0 0 0 0 0 1 1 : Play the Tone for 3 seconds
0 0 0 0 0 1 0 0 : Play the Tone for 4 seconds
.....
1 1 1 1 1 1 1 1 : Play the Tone for 255 seconds

Note: The values for the cycles or tone duration may be set to any value when the Message Info parameter is sent in the Stop Message primitive. The Stop Message primitive only cares about the message ID (that is, to stop the appropriate message on the PSN) in the Message Info parameter.

Monitor Mask parameter

This parameter is a bitmap of monitor values. Each bit in this bitmap indicates what digit/tone to monitor or not monitor a port for. Also, each bit has two values – 0 (indicates cancel the monitor) or 1 (start monitor).



Optional Parameter ID: 16

Parameter Size: 2 Bytes

Parameter Contents:

- **ASTERISK/OCTOTHORPE DURATION** : 5 Bits each – The duration in 100 milliseconds for which the asterisk/octothorpe has to be detected before it is reported as a valid tone/digit to the SCU. Valid values include 5 to 30.

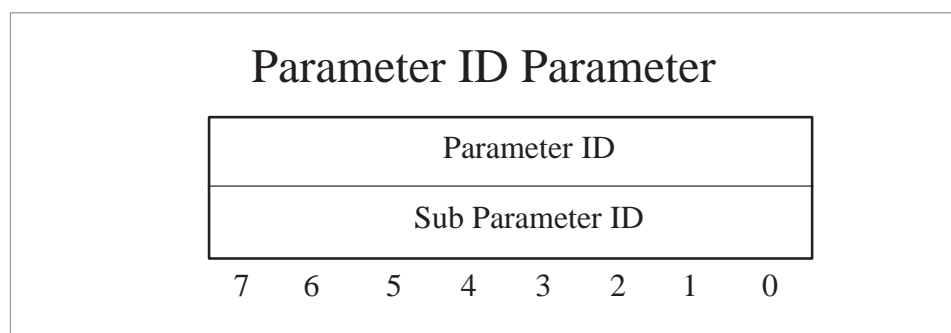
For example, a duration of 5 for an Asterisk implies that the user has to hold down the asterisk for 500 msec before it is detected as a valid asterisk by the PSN and reported to the SCU.

Note: For BBF: the SF Tone Duration, the signalling type (MF/DTMF), the minimum digits to collect before a blue box fraud is declared and the partial dial timer values are all determined from the BBTKSGRP. Therefore, it is necessary to datafill the “terminating” port (where “terminating” indicates Party B of a Connect or a Reconnect primitive) in table BBTKSGRP.

- **TONE BITMAP** : 3 Bits– This field is a boolean each for BBF, Octothorpe and Asterisk. If the bool is true then the port is monitored for the appropriate tone/digit. If the bool is false then the appropriate tone/digit monitoring on that port is canceled.

Parameter ID parameter

This parameter contains the Identifier of the Parameter (and the identifier of the sub-parameter if the parameter is composed of sub-parameters). It is returned to the SCU in case the decoding of the parameter resulted in an error.



Optional Parameter ID: 17

Parameter Length: 2 Bytes

Parameter Contents:

- **PARAMETER ID** : 1 Byte – This field is a byte that is used to represent the parameter in error. Values include:
 0 0 0 0 0 0 0 0 : Nil Error Cause

1	00000001	: Bearer Capability
2	00000010	: Billing Info
3	00000011	: Call Reference Identifier
4	00000100	: Control Info
5	00000101	: Destination Trunk Group
6	00000110	: Digit Collection
7	00000111	: Digits Collected
8	00001000	: Digits Outpulsed
9	00001001	: Digits To Outpulse
10	00001010	: Error Cause
11	00001011	: Flow Control Information
12	00001100	: Flow Control Encountered
13	00001101	: Instruction ID
14	00001110	: Instruction Tag
15	00001111	: Message Info
16	00010000	: Monitor Mask
17	00010001	: Parameter ID
18	00010010	: Port Count
19	00010011	: Port Info
20	00010100	: Port Service Information
21	00010101	: Port Status
22	00010110	: Reset Reason
23	00010111	: Session ID
24	00011000	: SigInfo Mask
25	00011001	: Signalling Info
26	00011010	: Switch ID
27	00011011	: Time of Day
28	00011100	: Tone Detected
	00011101 to 10000001	: Spare Values
130	10000010	: UCS Point In Call
131	10000011	: UCS STS
	10000100 to 11111111	: Spare Values

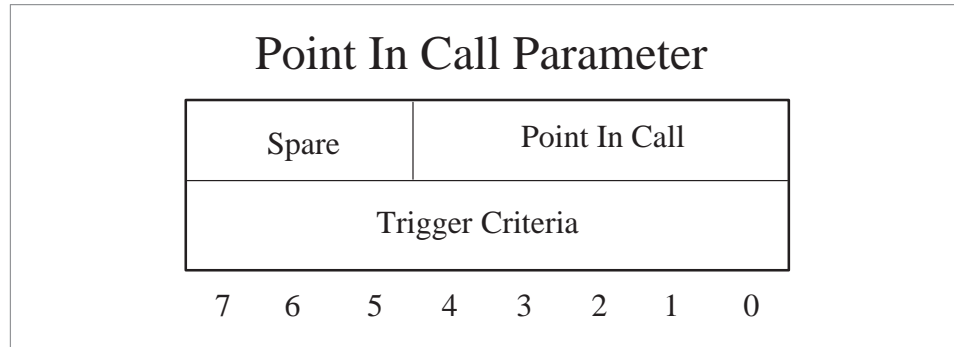
- SUB PARAMETER ID : 1 Byte – This field is used to represent the parameter in error.

Values for PTS agents include:

0	00000000	: Unknown
1	00000001	: Message Type
2	00000010	: Digits Outpulsed
3	00000011	: Digits to Outpulse
4	00000100	: PTS off-hook
5	00000101	: PTS on-hook
	00000110 to 11111111	: Spare

Point In Call parameter

This parameter contains the point in the call when all criteria were met and the PSN gave the SCU control over the call.



Optional Parameter Tag: 130

Parameter Length: 2 Byte

Parameter Contents:

- POINT IN CALL : 5 Bits – This field contains the point in the call, when the PSN determines that the call is a service call and needs to be controlled by the SCU. It is encoded as follows:
 - 0 00000 : Not Used
 - 1 00001 : Orig Null
 - 2 00010 : Authorize Orig Attempt
 - 3 00011 : Collect Information
 - 4 00100 : Analyze Information*
 - 5 00101 : Select Route
 - 6 00110 : Authorize Call Setup
 - 7 00111 : Send Call
 - 8 01000 : Orig Alerting
 - 9 01001 : Orig Active
 - 10 01010 : Orig Suspended
 - 11 01011 : Term Null
 - 12 01100 : Authorize Termination
 - 13 01101 : Select Facility
 - 14 01110 : Present Call
 - 15 01111 : Term Alerting
 - 16 10000 : Term Active
 - 17 10001 : Term Suspended
 - 10010 to 11111 : Spare Values

Note: Only the values marked with an “*” are currently supported.

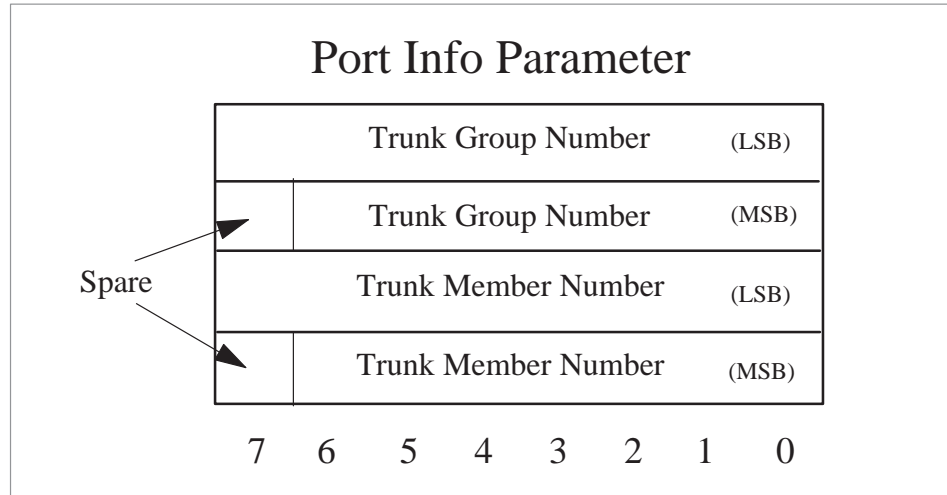
- TRIGGER CRITERIA: 8 Bits – This field is encoded as follows:

0	00000000	: Feature Activator
1	00000001	: Vertical Service Code
2	00000010	: Customized Access
3	00000011	: Customized Intercom*
4	00000100	: NPA
5	00000101	: NPA_NXX
6	00000110	: NXX
7	00000111	: NXX_XXXX
8	00001000	: NPA_NXXXXXX*
9	00001001	: Country_Code_NPA_NXXXXXX*
10	00010000	: Offhook Immed
11	00011000	: Net Busy
12	00011011	: Orig Called Party Busy
13	00011101	: Orig No Answer
14	00100000	: Orig Feature Activator
15	01100000	: Channel Setup PRI CLID
16	01100001	: Channel Setup PRI Addr
17	01100010	: Channel Setup PRI N00
18	01100011	: Channel Setup PRI Intl
19	01100100	: Specific Digit String Info*
20	01100101	: Specific Digit String ANI*
21	01100110	: Specific Digit String N00*
22	01100111	: Specific Digit String CIC
23	01101000	: Shared Interoffice CIC
24	01101001	: Shared Interoffice Info
25	01101010	: Shared Interoffice ANI
26	01101011	: Shared Interoffice Addr
27	01101100	: Shared Interoffice N00
28	01101101	: Shared Interoffice Intl
	01101110 to 11111111	: Unused (Spare)

Note: Only the values marked with an “*” are currently supported.

Port Info parameter

This parameter contains the port information for an agent. The port info consists of the external trunk group number and the trunk member number.



Optional Parameter ID: 19

Parameter Length: 4 Bytes

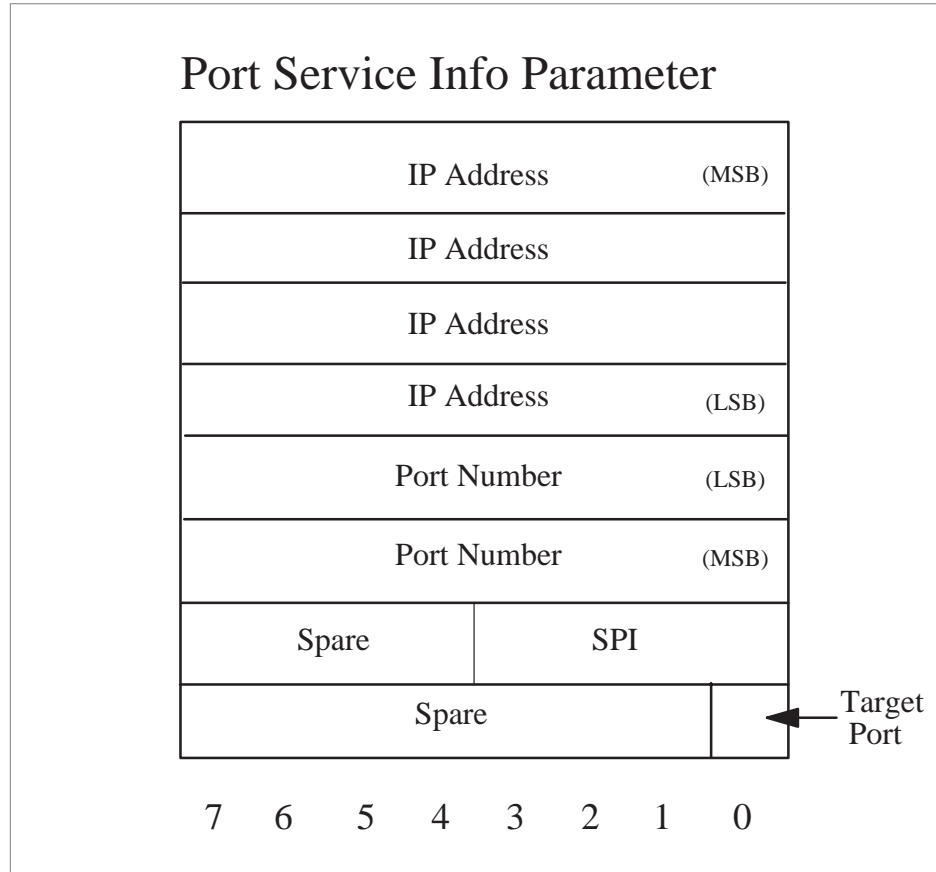
Parameter Contents:

- **TRUNK GROUP NUMBER** : 15 Bits – This field is the external trunk group number. The Range is 0 to 9999. A value of 32767 indicates NIL_TRUNK_GROUP.
- **TRUNK MEMBER NUMBER** : 15 Bits – This field contains the member number of the trunk. Range is 0 to 9999. A value of 32767 indicates a NIL_TRUNK_MEMBER.

Port Service Info parameter

This parameter contains the IP address and SPI information. The SCU returns the IP address and SPI to the PSN. The IP address is the return address used to send the event notification messages back to the SCU (to the

right address and port). The SPI number is the Service Programming Interface version for the specified agent.



Optional Parameter ID: 20

Parameter Length: 8Bytes

Parameter Contents:

- **IP ADDRESS** : 4 Bytes – This field contains the IP address sent by the SCU and which is eventually used to return the event notification messages.

This is a table of 4 bytes – for example, an IP address of 47.122.64.153 is stored as 4 bytes – byte 1 is 47, byte 2 is 122, byte 3 is 64 and byte 4 is 153. The *nil_ip_address* is defined as 0.0.0.0.

- **PORT NUMBER** : 2 Bytes – This field contains the transport layer port number associated with the IP address. The *nil_port_number* is defined as 0.

The following table illustrates how the combination of IP Address and Port number in the IP address Info parameter is interpreted for the different primitives.

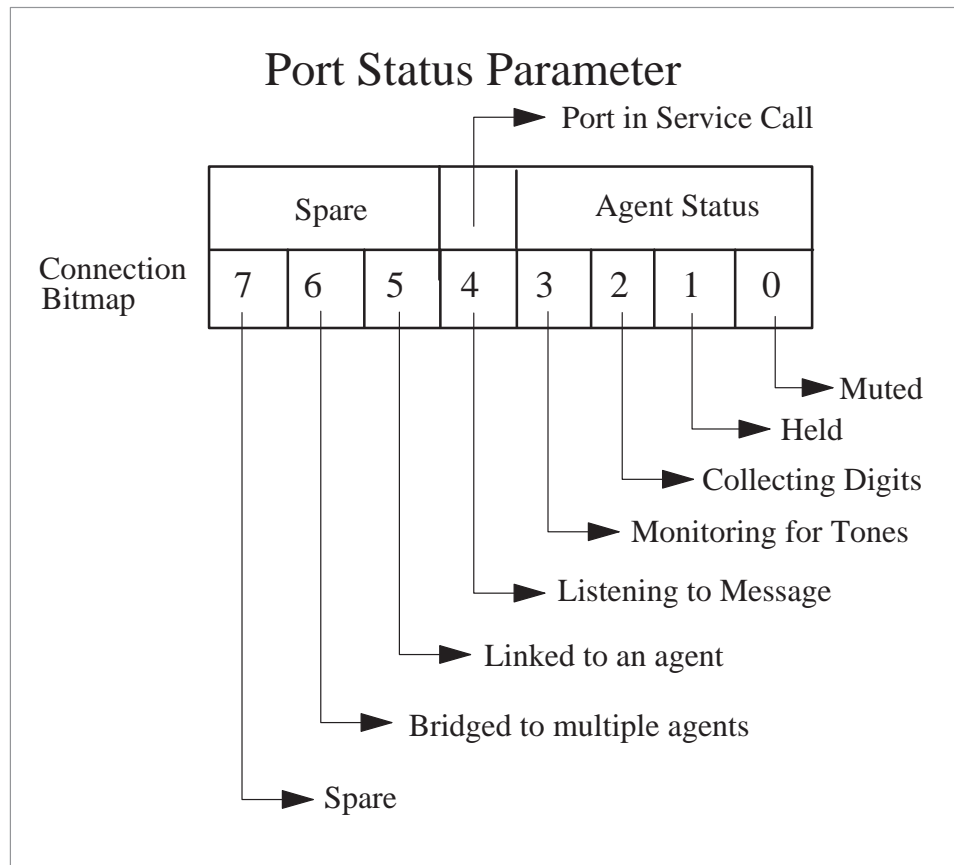
IP Address	Port Number	How is the IP Address Info used for in RESET SWITCH	How is the IP Address Info used in all the other Primitives
nil_ip_address	nil_port_number	Idle all ports that are currently serviced by the SCU (System Reset)	INVALID. If the parameter is mandatory for the primitive then the primitive is discarded. If the parameter is Optional for the primitive then the parameter is dropped and the processing of the primitive continues.
NON nil_ip_address	nil_port_number	Idle all ports with the given IP Address and any port number (Shelf Reset)	INVALID
nil_ip_address	NON nil_port_number	INVALID	INVALID
NON nil_ip_address	NON nil_port_number	Idle ports with the appropriate IP Address Info (Service Reset)	IP Address info parameter applies to a specific port – update the port's IP Address Info.

Note: Example Primitives include Set_IP_Address, and call control primitives such as Connect, or Disconnect.

- SPI : 4 Bits – This field contains the SPI version for the specified agent. Valid range is 1 to 15.
- TARGET PORT : 1 Byte – Since some primitives (i.e. Connect) involves more than one port in a call, the Target Port bit indicates to what port a primitive is being sent to.
 - 0 0 : Port_A This is the default value.
 - 1 1 : Port_B Valid only when used with the Connect and/or Reconnect.
 - 0 0 0 0 0 1 0 to 1 1 1 1 1 1 1 1 : Spare Values

Port Status parameter

This parameter contains the status of the port/agent. It also contains information which indicates if the port is currently being controlled by the SCU.



Optional Parameter ID: 21

Parameter Length: 2 Bytes

Parameter Contents:

- AGENT STATUS : 4 Bits – This field contains the agent status of the port, and is encoded as follows:

0	0 0 0 0 0	: Idle
1	0 0 0 0 1	: Seized
2	0 0 0 1 0	: Answered
3	0 0 0 1 1	: ManBusy
4	0 0 1 0 0	: Lockout
5	0 0 1 0 1	: System Busy
6	0 0 1 1 0	: PM Busy

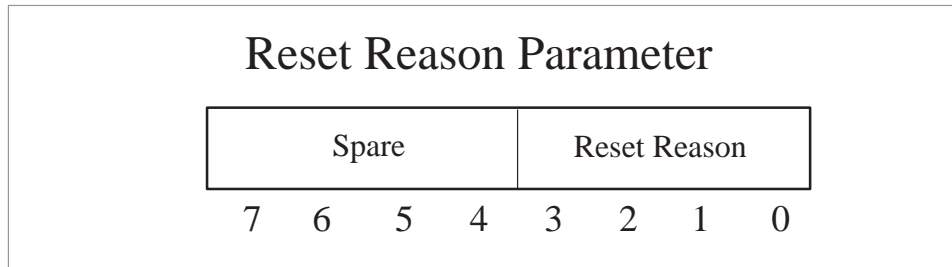
7 0 0 1 1 1 : Unknown
 1 0 0 0 to 1 1 1 1 : Spare Values

- **PORT IN SERVICE CALL** : 1 Bit – This field indicates if the port is currently a part of a service call or the port that is currently being serviced by the SCU.
 - 0 : Port not a part of a service call
 - 1 : Port is a part of a service call.
- **CONNECTION BITMAP** : 6 Bits – This is a bitmap where each bit indicates what the connection status of the agent is. The bits are used to represent the connection states as specified below:
 - Bit 0 : Muted
 - Values : 0 = port is un-muted,
 - 1 = port is muted.
 - Bit 1 : Held
 - Values : 0 = port is not held,
 - 1 = port is held.
 - Bit 2 : Collecting Digits
 - Values : 0 = port is not collecting digits,
 - 1 = port is in the process of collecting digits.
 - Bit 3 : Monitoring for Tones
 - Values : 0 = port is not monitoring for tones,
 - 1 = port is monitoring for tones.
 - Bit 4 : Listening to Message
 - Values : 0 = port is not listening to a message,
 - 1 = port is listening to message (announcement/tones).
 - Bit 5 : Linked to an Agent
 - Values : 0 = port is not linked to another agent,
 - 1 = port is linked to another agent.
 - Bit 6 : Bridged to multiple agents
 - Values : 0 = port is not bridged to two or more agents,
 - 1 = port is bridged to two or more agents.

Reset Reason parameter

This parameter contains the reason why a restart occurred on the PSN. It

indicates to the SCU the type of reset that is required at the SCU.



Optional Parameter ID: 22

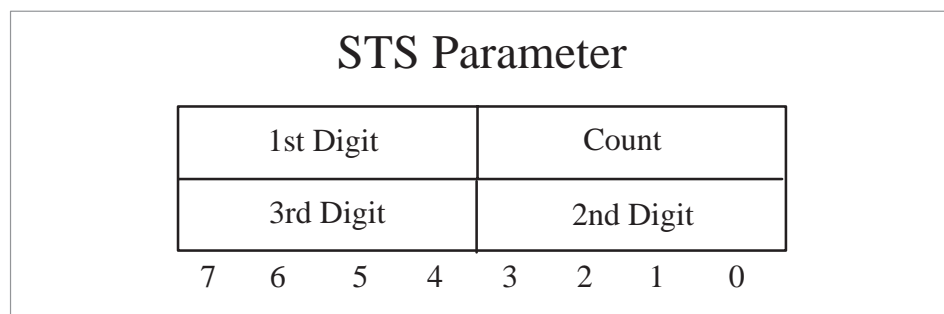
Parameter Length: 1 Byte

Parameter Contents:

- **RESET REASON** : 4 Bits – This field indicated the reason why a restart occurred on the PSN. Values include:
 - 0 0 0 0 : Unknown
 - 1 0 0 1 : Warm Restart performed on the PSN
 - 2 0 0 1 0 : Cold Restart performed on the PSN
 - 3 0 0 1 1 : Reload Restart performed on the PSN
 - 4 0 1 0 0 : PSN has just come into service
 - 5 0 1 0 1 : Arbitrator Heartbeat has failed
 - 0 1 1 0 to 1 1 1 1 : Spare Values

Serving Translation Scheme parameter

This parameter contains the serving translation scheme (STS) of the call. The switch sends this parameter to the SCU in the New Call event notification message.



Optional Parameter Tag: 131

Parameter Length: 2 Bytes

Parameter Contents:

- **COUNT** : 4 Bits – This contains the number of digits in the STS. Valid values include 1 to 3.
- **DIGITS 1, 2, and 3** : Each digit is 4 Bits – This field contains the 3 digits in the TBCD format, which are encoded as follows:

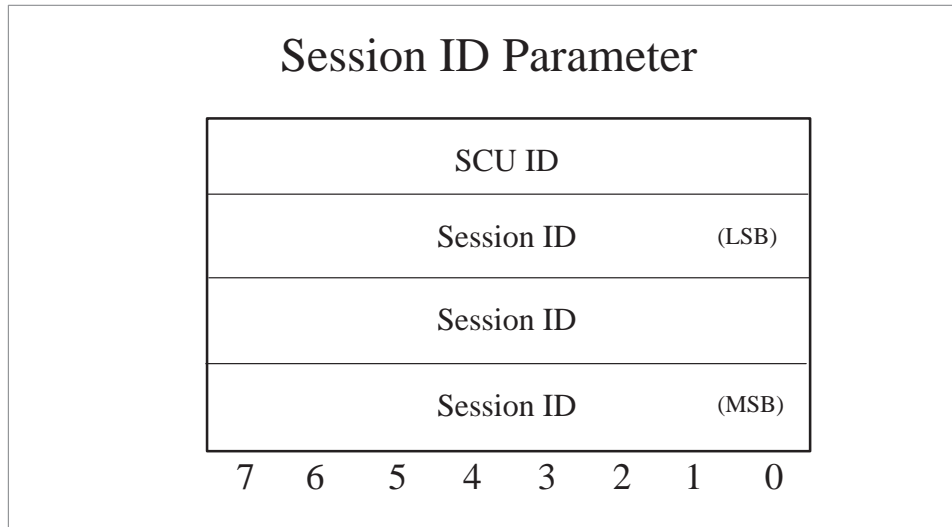
0	0 0 0 0	: Filler
1	0 0 0 1	: Digit 1
2	0 0 1 0	: Digit 2
3	0 0 1 1	: Digit 3
4	0 1 0 0	: Digit 4
5	0 1 0 1	: Digit 5
6	0 1 1 0	: Digit 6
7	0 1 1 1	: Digit 7
8	1 0 0 0	: Digit 8
9	1 0 0 1	: Digit 9
10	1 0 1 0	: Digit 0
11	1 0 1 1	: *
12	1 1 0 0	: #
13	1 1 0 1	: D
14	1 1 1 0	: E
15	1 1 1 1	: F

Session ID parameter

This parameter contains the Session ID. The Session ID is generated by the SCU to associate the different ports involved in a service call.

Upon receipt of the Session ID parameter, the PSN does not perform any error checking. It merely re-transmits this value in the event notification

message.



Optional Parameter ID: 23

Parameter Length: 4 Bytes

Parameter Contents:

- SCU ID : 1 Byte – This field is a unique 1 byte ID generated by the SCU.
- SESSION ID : 3 Bytes – This field is a unique 3 byte ID generated by the SCU. Values Include

0 : 0

to

1 : 16777215

Signalling Info parameter

This parameter contains the Signalling Information in the standard format that is applicable to the signalling type of the port. This parameter may be used to receive or send signalling information from / to the SCU.

Depending upon the primitive/event notification in which this parameter is sent/received and the signalling type of the associated port, its contents are different. In general, the contents contain parameters that are encoded in the standard format.

For PTS agents, there are no standards used. For SS7 agents, the parameters are encoded based on the TR444 (Issue 4, August 1992) and GR394 (Issue 1, February 1994), and for PRI agents. Refer to Chapter 15, “PRI messages” for more details.

Signalling Info is included in the following primitives and event notification messages:

- *Connect primitive from the SCU*
 - the following parameters are sent in Signalling Info on a 4-Wire PTS agent (outpulsing, and therefore the Signalling Info parameter, is not allowed on 2-Wire PTS agents). A Connect may contain from 1 to 3 Signalling Info Parameters where each Signalling Info contains either a Bearer Capability AND a Digits to Outpulse or just a Digits to Outpulse parameter. These parameters are encoded as described earlier in “Bearer Capability” on page 13 and “Digits To Outpulse” on page 31.
 - the following IAM message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Forward Call Indicator (Mandatory)
 - Calling Party Category (Mandatory)
 - User Service Information (Mandatory)
 - Called Party Number (Mandatory)
 - Nature of Connection Ind.(Mandatory)

Note: Ensure the COT datafill in Table TRKSGRP is correct when the ContinuityCheckIndicator in the Nature of Connection is set.
 - Calling Party Address (Optional)
 - Carrier Identification(Optional)
 - Carrier Selection Information (Optional)
 - Channel Assignment Map (Optional)
 - Charge Number (Optional)
 - Authcode (GD) (Optional)

Note: GD implies that this parameter is enclosed in the Generic Digits.
 - Call Reference Identifier (GD) (Optional)
 - Callid (GD) (Optional)

- CLI Admin (GD) (Optional)
- IMT Info (GD) (Optional)
- RLT Treatment Code (GD) (Optional)
- Term SWID & TRKGRP (GD) (Optional)
- XFR Operator Queue (GD) (Optional)
- Network Information (Optional)
- Network Specific Facilities (Optional)
- Network Specific IAM (Optional)
- Operator Information (Optional)
- Operator Services Indicator (Optional)
- Originating Line Information (Optional)
- Supplementary Line Info (Optional)
- Transit Network Selection (Optional)
- the following SETUP message parameters (on a PRI agent) encoded as detailed in Chapter 15, “PRI messages:”
 - Protocol Discriminator (Mandatory)
 - Message Type (Mandatory)
 - Bearer Capability (Mandatory)
 - Called Party Number (Mandatory)
 - Business Group (Optional)
 - Called Party Subaddress (Optional)
 - Progress Indicator (Optional)
 - Calling Party Number (Optional)
 - Calling Party Subaddress (Optional)
 - Display (Optional)
 - Higher Layer Compatibility (Optional)
 - Lower layer Compatibility (Optional)
 - Network Specific Facility (Optional)
 - Original Called Number (Optional)
 - Transit Network Selection (Optional)
 - User to User Information (Optional)
- *Disconnect primitive from the SCU*

-
- the following REL message parameters (on an SS7 agent) encoded as detailed in Chapter 15, “PRI messages:”
 - Message Type (Mandatory)
 - Cause Indicators (Mandatory)
 - the following DISC message parameters (on a PRI agent) encoded as detailed in Chapter 15, “PRI messages:”
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Cause (Mandatory)
 - the following REL message parameters (on a PRI agent) encoded as detailed in Chapter 15, “PRI messages:”
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Cause (Optional)
 - User to User Information (Optional)
 - *Transmit Siginfo primitive from the SCU*
 - the following parameters are sent in Signalling Info on a 4-Wire PTS agent (outpulsing, and therefore the Signalling Info parameter, is not allowed on 2-Wire PTS agents). A Transmit Siginfo may contain from 1 to 3 Signalling Info Parameters where each Signalling Info contains a Digits to Output parameter. This parameter is encoded as described earlier in “Digits To Output” on page 31.
 - Message Type (Mandatory)
 - Digits to Output (Mandatory)
 - Digits to Output (Optional)
 - Digits to Output (Optional)
 - the following information is sent (on a PTS agent) for a PTS Onhook. There is no Signaling Information present when the message type is PTS Onhook:
 - Message Type (Mandatory)
 - the following information is sent on a 4-Wire PTS agent for a PTS Offhook (Offhook Propagation is not allowed on 2-Wire PTS Agents). There is no Signaling Information present when the message type is PTS Offhook:
 - Message Type (Mandatory)

- the Transmit Siginfo is used to send the IAM, ANM, CPG, FAA, FAR, FRJ, PAM, SUS, REL & RES for an SS7 port. For the definition of these SS7 messages, please refer to “Signalling Event” bullet in this section.
- the Transmit Siginfo is used to send the CONNECT, CALLPROC, FACILITY, RLC, REL & DISC messages for a PRI port. For the definition of these PRI messages, please refer to “Signalling Event” bullet in this section.
- *Off-Hook event notification to the SCU*
 - the following ANM message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Backward Call Indicator (Optional)
 - Call Reference (Optional)
 - Carrier Selection (Optional)
 - Internetwork Specific ANM (Optional)
 - Intranetwork Specific ANM (Optional)
 - Network Specific ANM (Optional)
 - Operator Information (Optional)
 - US Network Parameter (Optional)
 - User to User Indicator (Optional)
 - User to User Information (Optional)
 - the following CONNECT message parameters (on a PRI agent) encoded as detailed in Chapter 15, “PRI messages:”
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - User to User Information (Optional)
- *On-Hook event notification to the SCU*
 - the following REL (A SUSpend message is reported to the PSN as a RELease message and the port is taken down) message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Cause Indicators (Mandatory)
 - User to User Indicator (Optional)
 - User to User Information (Optional)

- the following RSC message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
- the following DISC (When a PRI agent goes on-hook – a DISConnect is sent to the SCU if the release/on-hook is normal and a RELease is sent if the on-hook is abnormal) message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Cause (Mandatory)
 - User to User Information (Optional)
- the following REL message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Cause (Optional)
 - User to User Information (Optional)
- *New Call event notification to the SCU*
 - the following IAM message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Forward Call Indicator (Mandatory)
 - Calling Party Category (Mandatory)
 - User Service Information (Mandatory)
 - Called Party Number (Mandatory)
 - Nature of Connection Ind.(Mandatory)

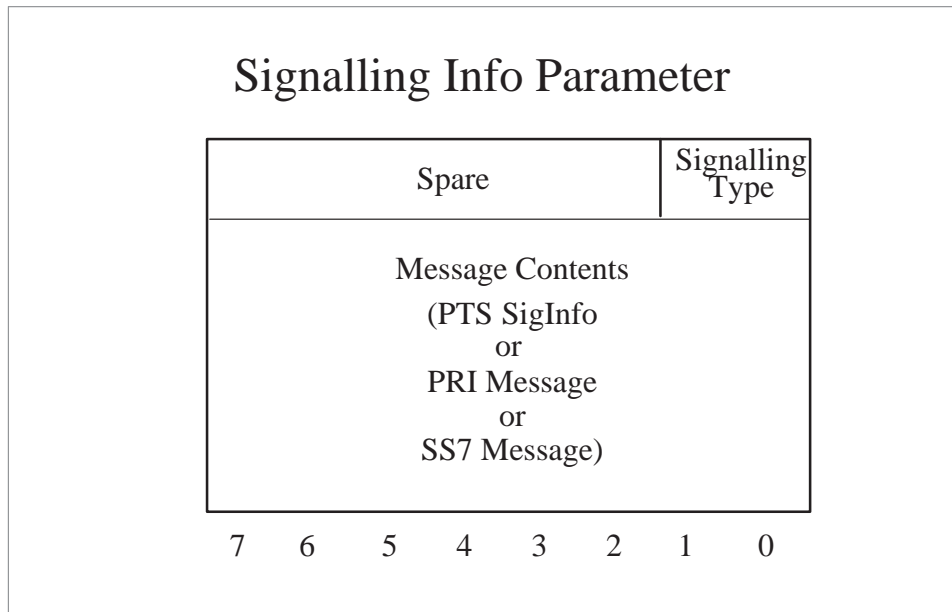
Note: SS7 COT (COnTinuity) checks are not supported. Hence the ContinuityCheckIndicator in the Nature of Connection must not be set.
 - Calling Party Address (Optional)
 - Carrier Identification (Optional)
 - Carrier Selection Information (Optional)
 - Channel Assignment Map (Optional)
 - Charge Number (Optional)

- Authcode (GD) (Optional)
Note: GD implies that this parameter is enclosed in the Generic Digits.
- Call Reference Identifier (GD) (Optional)
- Callid (GD) (Optional)
- CLLI Admin (GD) (Optional)
- IMT Info (GD) (Optional)
- RLT Treatment Code (GD) (Optional)
- Term SWID & TRKGRP (GD) (Optional)
- XFR Operator Queue (GD) (Optional)
- Network Information (Optional)
- Network Specific Facilities (Optional)
- Network Specific IAM (Optional)
- Operator Information (Optional)
- Operator Services Indicator (Optional)
- Originating Line Information (Optional)
- Supplementary Line Info (Optional)
- Transit Network Selection (Optional)
- the following SETUP message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Protocol Discriminator (Mandatory)
 - Message Type (Mandatory)
 - Bearer Capability (Mandatory)
 - Called Party Number (Mandatory)
 - Business Group (Optional)
 - Called Party Subaddress (Optional)
 - Progress Indicator (Optional)
 - Calling Party Number (Optional)
 - Calling Party Subaddress(Optional)
 - Display (Optional)
 - Higher Layer Compatibility (Optional)
 - Lower layer Compatibility (Optional)

-
- Network Specific Facility (Optional)
 - Original Called Number (Optional)
 - Transit Network Selection (Optional)
 - User to User Information (Optional)
 - *Signalling Event event notification to the SCU*
 - the following parameters are sent in Signalling Info (on a PTS agent). These parameters are encoded as described earlier in “Digits Outpulsed” on page 30.
 - Digits Outpulsed Indicator (Mandatory)
 - the following ACM message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Backward Call Indicator (Mandatory)
 - Call Reference (Optional)
 - US Network Parameter (Optional)
 - User to User Indicator (Optional)
 - User to User Information (Optional)
 - the following COT message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Continuity Indicators (Mandatory)
 - the following CPG message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Event Information (Mandatory)
 - Backward Call Indicator (Mandatory)
 - Party Information (Optional)
 - Redirection Number (Optional)
 - Supply end-to-end Inf. Req. (Optional)
 - Supply end-to-end Inf. Resp. (Optional)
 - User to User Indicator (Optional)
 - User to User Information (Optional)
 - the following FAA message parameters (on an SS7 agent) encoded as per the TR444 and GR394:

- Message Type (Mandatory)
- Facility Indicator (Mandatory)
- Call Reference (Optional)
- the following FAR message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Facility Indicator (Mandatory)
 - Call Reference (Optional)
 - Called Party Address (Optional)
 - Calling Party Number (Optional)
 - Charge Adjustment (Optional)
 - Charge Number (Optional)
 - Callid (GD) (Optional)
 - Operator Information (Optional)
 - Originating Line Info (Optional)
- the following FRJ message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Facility Indicator (Mandatory)
 - Cause Indicator (Mandatory)
 - Call Reference (Optional)
- the following PAM message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - RPM Message (Mandatory)
- the following RES message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Suspend/Resume Indicators (Mandatory)
- the following SUS message parameters (on an SS7 agent) encoded as per the TR444 and GR394:
 - Message Type (Mandatory)
 - Suspend/Resume Indicators (Mandatory)

- the following PROGRESS message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Progress Indicator (Mandatory)
 - Cause (Optional)
 - User to User Information (Optional)
- the following ALERT message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - User–User Information (Optional)
 - Progress Indicator (Optional)
- the following CALL PROCEEDING message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
- the following FACILITY message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - Protocol Discriminator (Mandatory)
 - Called Party Number (Optional)
 - Called Party Subaddress (Optional)
- the following REL Complete (RLC) message parameters (on a PRI agent) encoded as per AD9100:UCSPRI1:
 - Message Type (Mandatory)
 - User to User Information (Optional)



Optional Parameter ID: 25

Parameter Length: MAX: 412 Bytes

Parameter Contents:

- **SIGNALLING TYPE** : 2 Bits – TYPE: Signalling Type Parameter
- **MESSAGE CONTENTS** : MAX: 411 Bytes – This field is the message contents containing the parameters in the standard format listed earlier. The contents are encoded depending upon the type of signalling used.

Table 20-4
PTS, SS7, and PRI messages included in Primitives/Event notifications

Primitive / Event Notification	PTS Messages	SS7 Messages	PRI Messages
Connect	Digits To Outpulse, Digits To Outpulse with Bearer Capability	IAM	SETUP
—continued—			

Table 20-4
PTS, SS7, and PRI messages included in Primitives/Event notifications
 (continued)

Primitive / Event Notification	PTS Messages	SS7 Messages	PRI Messages
Disconnect	—	REL	DISC, REL
Transmit Signinfo	Digits To Outputpulse, PTS On-Hook, PTS Off-Hook	ACM, IAM, ANM, REL, CPG, FAA, FAR, FRJ, PAM, RES, SUS	CONNECT, REL, DISC, ALERT, FACility
New Call	—	IAM	SETUP
Off-Hook	—	ANM	CONNECT
On-Hook	—	REL, RSC	DISC, REL
Signalling Event	Digits Outputpulsed Indicator	ACM, CPG, COT, FAA, FAR, FRJ, PAM, RES, SUS	PROGRESS, ALERT, CALL PROCeeding, FACility, RELease COMPlete
—end—			

PTS Message Contents:

— PTS MESSAGE TYPE : 1 Byte – This field is the message type for a PTS message. Values include:

0 00000000 : Unknown
 1 00000001 : Digits to Outputpulse
 2 00000010 : Digits to Outputpulse with Bearer Capability
 3 00000011 : PTS Off-Hook
 4 00000100 : PTS On-Hook
 5 00000101 : Digits Outputpulsed
 00000110 to 11111111 : Spare

— PTS SIGINFO MESSAGE AREA : MAX 410 Bytes – This area contains the signinfo message but depends on the Message Type.

– PTS Message Contents:

Digits to Outputpulse then the PTS Signinfo Message Area contains from 1 up to 3 Digits to Outputpulse Parameter(s). Please refer to “1.1.15 Digits To Outputpulse” on page 31.

Digits to Outputpulse with Bearer Capability then the PTS Siginfo Message Area contains the Bearer Capability Parameter first in the area, “1.1.5 Bearer Capability” on page 13, and the following bytes contains from 1 up to 3 of Digits to Outputpulse Parameter(s), “1.1.15 Digits To Outputpulse” on page 31.

PTS Off-Hook then there is no information in the PTS Siginfo Message Area. There is no signalling information associated with a PTS Off-Hook.

PTS On-Hook then there is no information in the PTS Siginfo Message Area. There is no signalling information associated with a PTS On-Hook.

Digits Outputpulsed then PTS Siginfo Message Area contains the Digits Outputpulsed Parameter. Please refer to “1.1.14 Digits Outputpulsed” on page 30.

– SS7 Message Contents:

The TR444 and GR394 SS7 with ISDN support are used to define the contents of the SS7 Message Contents.

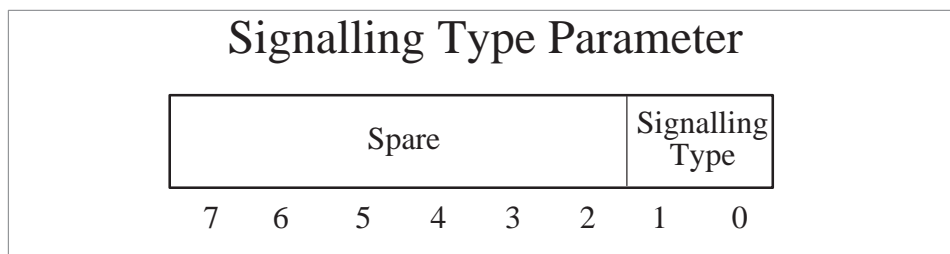
– PRI Message Contents:

The AD9100:UCSPRI1 document is used to define the contents for the PRI Message Contents.

Note: The SS7/PRI parameter contents are not defined in this document. Please refer to the above documents for details.

Signalling Type parameter

This parameter is used in the Agent Data Event and in the New Call Event.



Optional Parameter ID: 32

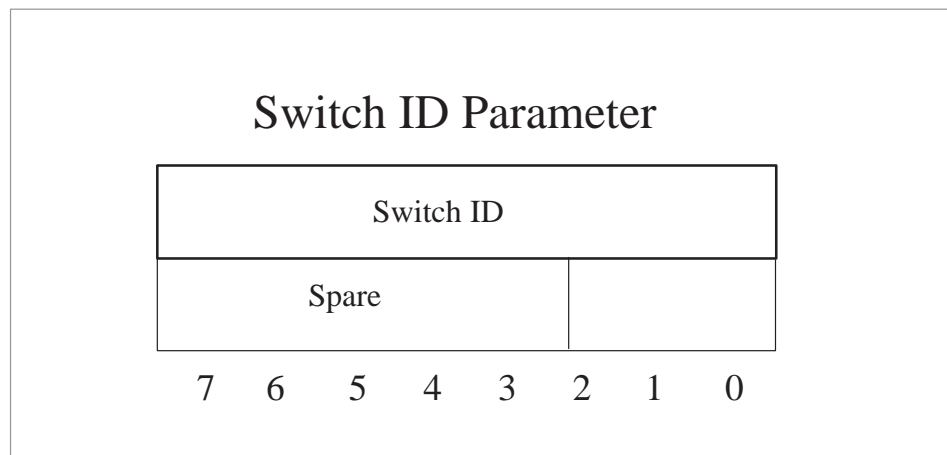
Length: 1 Byte

Parameter Contents:

- **SIGNALLING TYPE** : 2 Bits – This field is the type of signalling that the port is using. Values include:
 - 0 0 : PTS (4 Wire)
 - 0 1 : PTS (2 Wire)
 - 1 0 : SS7
 - 1 1 : PRI

Switch ID parameter

This parameter contains the switch ID of the PSN.



Optional Parameter ID: 26

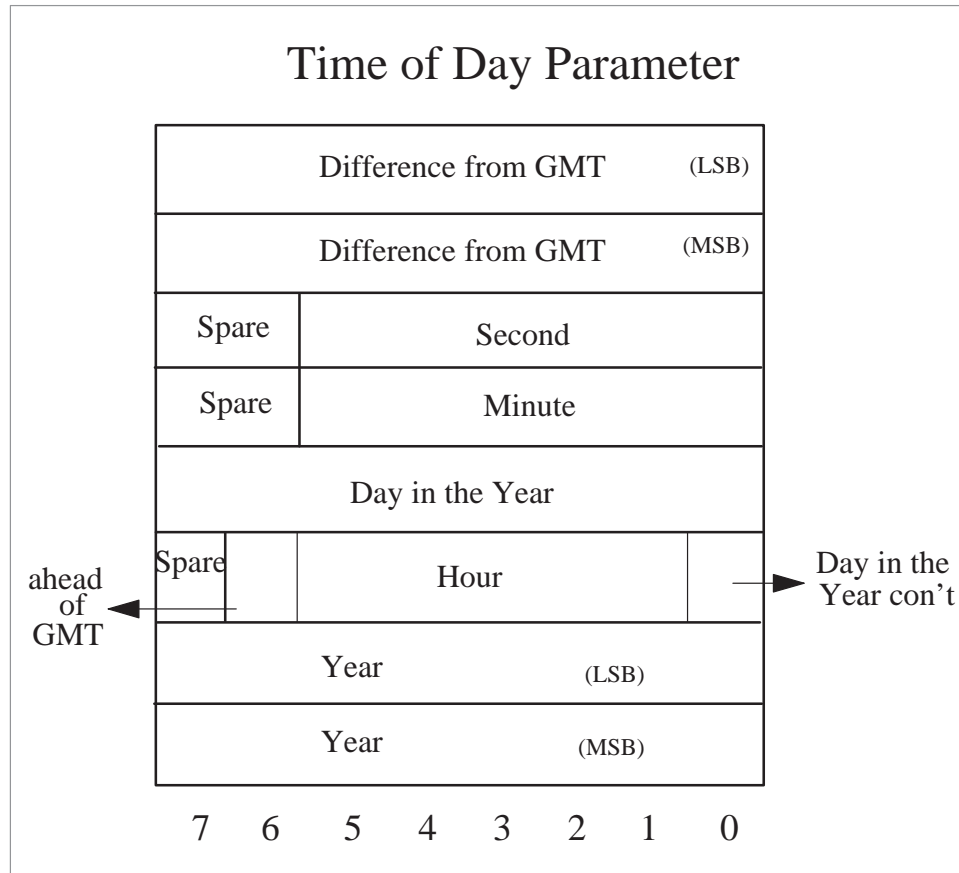
Parameter Length: 2 Bytes

Parameter Contents:

- **SWITCH ID** : 11 bits – This is the switch ID with a range of 0 to 999.

Time of Day parameter

This parameter contains the time of the day and is returned in the Current Time of Day event notification to the SCU.



Optional Parameter ID: 27

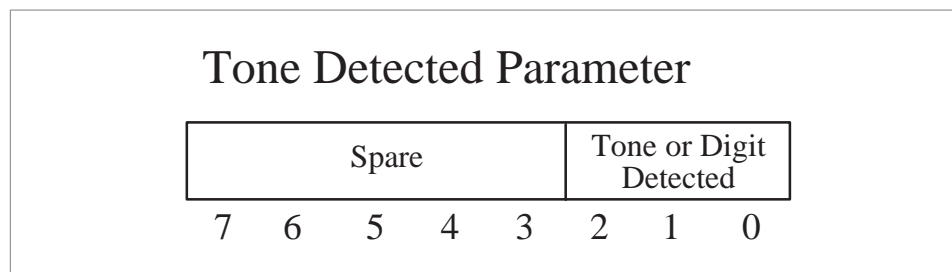
Parameter Length: 8 Bytes

- Parameter Contents:
- YEAR: 2 Bytes – This field contains the year.
- DIFFERENCE FROM GMT: 2 Bytes – This field contains the time difference between the time specified by the “hour, minute and second” fields above and the Greenwich Mean Time.
- DAY IN THE YEAR : 9 Bits – This field contains the day of the year. Valid values include 1–365.
- MINUTE : 6 Bits – This field contains the minute. Valid values include: 0 to 59. The rest are spare values.

- **AHEAD OF GMT:** 1 Bits – If true, this field indicates that the switch time is **AHEAD** of the Greenwich Mean Time (GMT). The switch time is **BEHIND** the GMT if this field is set to false.
- **HOUR :** 5 Bits – This field contains the hour. Valid values include: 0 to 23. The rest are spare values.
- **SECOND :** 6 Bits – This field contains the minute. Valid values include: 0 to 59. The rest are spare values.

Tone Detected parameter

This parameter contains the tone that is detected on a given port/agent by the PSN.



Optional Parameter ID: 28

Parameter Length: 1 Byte

Parameter Contents:

- **TONE OR DIGIT DETECTED :** 3 Bits – This field contains the tone or the digit that is detected, and is encoded as follows:
 - 0 0 0 : tone / digit monitoring aborted
 - 1 0 0 1 : asterisk detected
 - 2 0 1 0 : octothorpe detected
 - 3 0 1 1 : SF tone detected
 - 4 1 0 0 : BBF tone detected
 - 5 1 0 1 : BBF digits detected
 - 1 1 0 – 1 1 1 : Unused (Spare)

Note: The BBF digits are not sent to the SCU by the PSN.

UCS PSN LOPER messages and parameters version 4

This chapter contains a detailed description of the UCS PSN messages and parameters Version 4 described in the Low Overhead Protocol Encoding Rule (LOPER) message format.

LOPER Format

Each of the messages in the Low Overhead Protocol Encoding Rule (LOPER) follow the same basic format and rules.

The following rules apply to LOPER:

- A message is byte aligned meaning that each parameter starts on a new byte.
- The first byte in a message is considered to be the 0th index of the message, the second byte in the message is considered to be the 1st index of the message, and so on,
- Two bytes are used in describing the length of a message/parameter. Two bytes are also used to contain at which byte location in the message that the optional parameters begin in the message.

Note: Two bytes are used because of the large structures found PSN Call Processing such as the New Call event which is 310 bytes. The maximum value that a byte can represent is 255 and this is not large enough to indicate the length of a New Call event message.

- A LOPER message can either contain one primitive, one event, or one macro which can contain up to five primitives.
- The SS7 signalling message of the Siginfo Contents in a Optional Siginfo Parameter to be transmitted on a SS7 agent is not validated. Since the SS7 signalling message is to be built by the SCU in the TR444 BellCore standard, it is assumed that the SS7 signalling message has been correctly built and is sent onto the SS7 network with out validation. Any error in the SS7 signalling message is caught by the network and handled accordingly by the SS7 network.

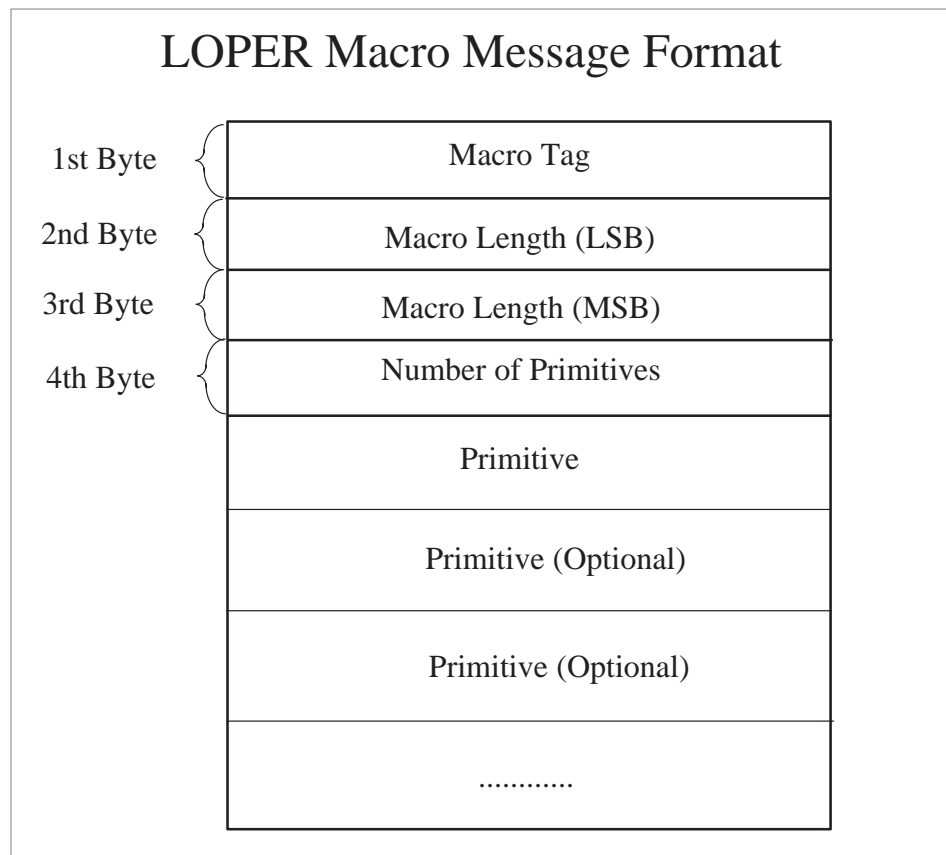
Messages sent from the SCU to the PSN

Macros

The first byte of a Macro designates that the message is a Macro. The second and third bytes indicate the length of the Macro in the number of bytes that comprise the length of the message from the Macro Tag to the end of the last primitive in the Macro. The fourth byte identifies how many primitives there are in the Primitive Area. The Primitive Area contains the individual primitives. Please refer to “Figure 1 LOPER Macro Message Format” on page 3.

If there is an error in the decoding of one of the primitives found in the Macro, that primitive is dropped and the decoding process is ceased since the rest of the Macro is considered to be corrupted. Any primitive decoded before the error is still be processed.

Figure 1 LOPER Macro Message Format

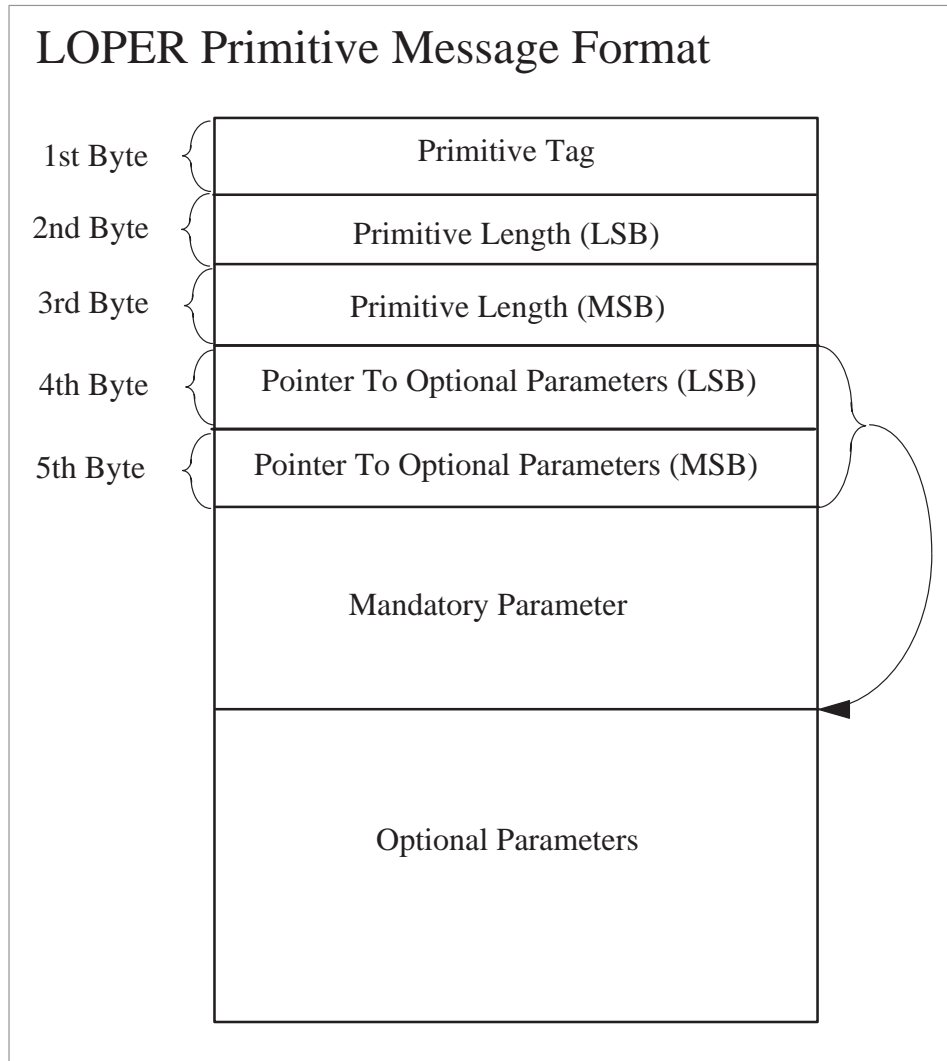


- **MACRO TAG:** 1 Byte – This field is a byte that is used to identify the primitive in the message. Values include:
 0 0 0 0 0 0 0 : Macro Tag

Primitives

The first byte is the Primitive Tag and identifies the primitive. The second and third bytes indicate the length of the primitive beginning with the primitive tag to the end of the mandatory parameters or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the Mandatory Parameters. The contents of the mandatory parameter depend on the primitive/event and the customer using LOPER. Please refer to “Figure 2 LOPER Primitive Message Format” on page 4.

Figure 2 LOPER Primitive Message Format



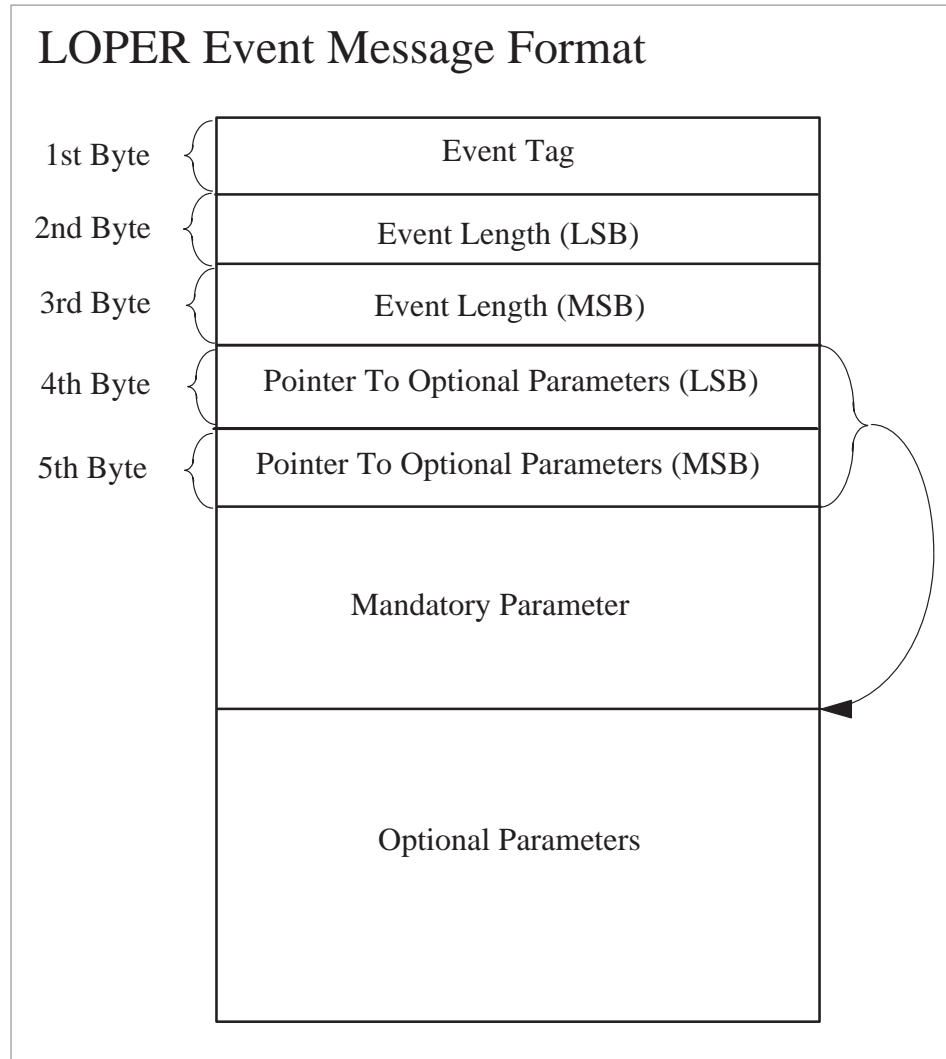
- 1 **PRIMITIVE TAG:** 1 Byte – This field is a byte that is used to identify the primitive in the message. Values include:

0	00000000	: Nil Primitive Tag
1	00000001	: Bridge
2	00000010	: Collect Digits
3	00000011	: Connect
4	00000100	: Disconnect
5	00000101	: Hold
6	00000110	: Monitor
7	00000111	: Mute
8	00001000	: RESERVED FOR INTERNAL PSN USE
9	00001001	: New Call Accepted
10	00001010	: New Call Rejected
11	00001011	: Play Message
12	00001100	: PPCD
13	00001101	: Query Port
14	00001110	: Reconnect
15	00001111	: Set Billing Record
16	00010000	: Stop Message
17	00010001	: Transmit Signalling Info
18	00010010	: Error Detected
19	00010011	: Heart Beat
20	00010100	: Port Status
21	00010101	: Query TOD
22	00010110	: Reset Switch
23	00010111	: Set IP Address
24	00011000	: Flow Control
	00011001 to 11111111	: Spare

Messages sent from the PSN to the SCU

Events

The first byte is the Event Tag and identifies the event. The second and third bytes indicate the length of the event beginning with the event tag to the end of the mandatory parameters or optional parameters if present. The fourth and fifth bytes contain the Nth byte location at which the optional parameters begin in the message. The sixth byte is the start of the Mandatory Parameters. The contents of the mandatory parameter depend on the primitive/event and the customer using LOPER. Please refer to “Figure 3 LOPER Event Message Format” on page 6.

Figure 3 LOPER Event Message Format

- **EVENT TAG:** 1 Byte – This field is a byte that is used to identify the event in the message. Values include:
 - 0 00000000 : Current TOD
 - 1 00000001 : Digits Collected
 - 2 00000010 : Error Detected
 - 3 00000011 : In Service
 - 4 00000100 : Instruction Completed
 - 5 00000101 : Message Played
 - 6 00000110 : New Call
 - 7 00000111 : Off Hook
 - 8 00001000 : On Hook
 - 9 00001001 : Port Status

10 00001010 : Query Port
11 00001011 : Route Not Available
12 00001100 : Route Selected
13 00001101 : Signalling Event
14 00001110 : Tone Detected
15 00001111 : Agent Data
16 00010000 : Stop Heartbeat
00010001 to 11111111 : Spare

Parameters

LOPER divides parameters into two classes, mandatory and optional. Mandatory parameters contain the information required by the primitive/event in order to be processed. Optional parameters contain the information not required by the primitive/event but add extra information for processing. Please refer to the tables in AD9100:UCSPARM1 for a mapping of which parameters are mandatory and/or optional for primitives and events.

Mandatory Parameters

Mandatory parameters in LOPER are customized for each primitive and event. The sixth byte in the primitive/event is always the beginning of the mandatory parameters for primitives and events. Please refer to “Figure 2 LOPER Primitive Message Format” on page 4. There is no need for a mandatory parameter tag since mandatory parameters are identified by the primitive/event id and can only be found in their corresponding primitive/event such as a Connect Mandatory Parameter can only be found in a Connect Primitive. There is also no need for a length indicator for the mandatory parameters since the pointer to the optional parameters designates the end of the mandatory parameters and if there are no optional parameters then the length of the primitive/event is the end of the mandatory parameters. The mandatory parameters are defined later in this document and the contents mandatory parameters are defined in AD9100:UCSPARM1.

If there is an error in the decoding of the mandatory parameters then entire message is dropped, the OM PSN_ERPS:DECODEFL is incremented, the logs PSN202 and PSN212 are produced, and an Error Detected event is produced with the Error Cause: HEADER_DECODE_FAILURE_EC

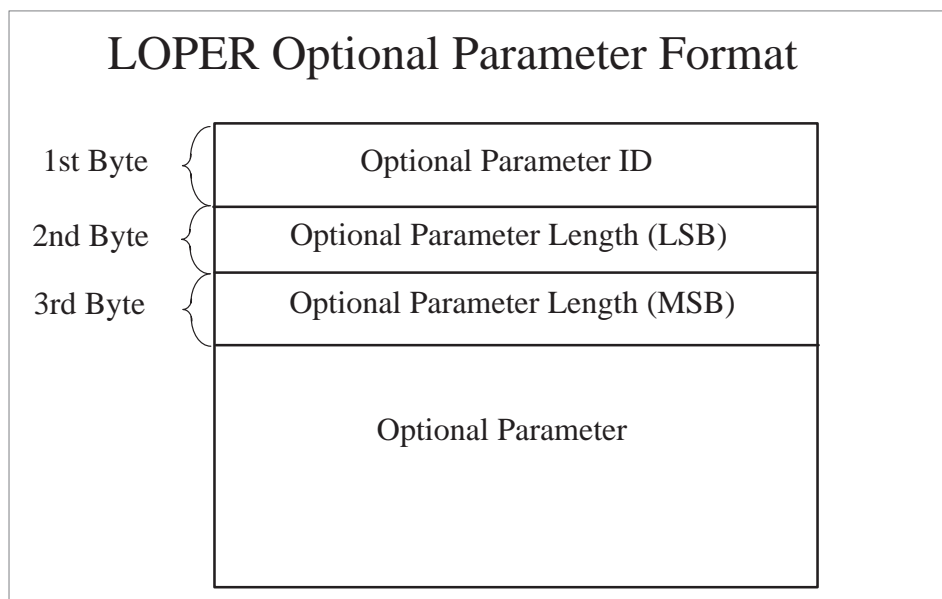
If there is an error in the validating of the mandatory parameters values then entire message is dropped, the OM PSN_ERPS:MANDPDEF is incremented, the logs PSN202 and PSN212 are produced, and an Error Detected event is produced with the Error Cause : PARAM_CONTENTS_OUT_OF_ – RANGE_EC.

Optional Parameters

The first byte of an optional parameter is the Optional Parameter Tag and identifies the optional parameter. The second and third bytes indicate the length of the optional parameter in bytes starting with the beginning of the optional parameter ID to the end of the optional parameter. The contents of the optional parameter depends on the specific parameter and the customer using LOPER. The contents of the optional parameters are defined in PSN PARMS. Please refer to “Figure 4 LOPER Optional Parameter Format” on page 9.

If a LOPER message is received with an unknown Optional Parameter ID, the unknown Optional Parameter is skipped and the decoding of the rest of the optional parameters in the message continued. If there is an error in the decoding a known Optional Parameter, the Optional Parameter is dropped, decoding process is ceased since the reset of the message is considered to be corrupted, the OM PSN_ERPS:OPPRMDEF is incremented, the logs PSN202 and PSN212 are produced, and the primitive along with the optional parameters decoded before the error is processed.

Figure 4 LOPER Optional Parameter Format



- **OPTIONAL PARAMETER ID** : 1 Byte – This field is a byte that is used to represent the parameter. Values include:
 - 0 00000000 : Unknown
 - 1 00000001 : Bearer Capability
 - 2 00000010 : Billing Info

3	00000011	: Call Reference Identifier
4	00000100	: Control Info
5	00000101	: Destination Trunk Group
6	00000110	: Digit Collection
7	00000111	: Digits Collected
8	00001000	: Digits Outpulsed
9	00001001	: Digits To Outpulse
10	00001010	: Error Cause
11	00001011	: Flow Control Information
12	00001100	: Flow Control Encountered
13	00001101	: Instruction ID
14	00001110	: Instruction Tag
15	00001111	: Message Info
16	00010000	: Monitor Mask
17	00010001	: Parameter ID
18	00010010	: Port Count
19	00010011	: Port Info
20	00010100	: Port Service Information
21	00010101	: Port Status
22	00010110	: Reset Reason
23	00010111	: Session ID
24	00011000	: SigInfo Mask
25	00011001	: Signalling Info
26	00011010	: Switch ID
27	00011011	: Time of Day
28	00011100	: Tone Detected
29	00011101	: Agent Type
30	00011110	: COT Required
31	00011111	: ISUP Index
32	00100000	: Signalling Type
33	00100001 to 10000001	: Spare Values
130	10000010	: UCS Point In Call
131	10000011	: UCS STS
	10000100 to 11111111	: Spare Values

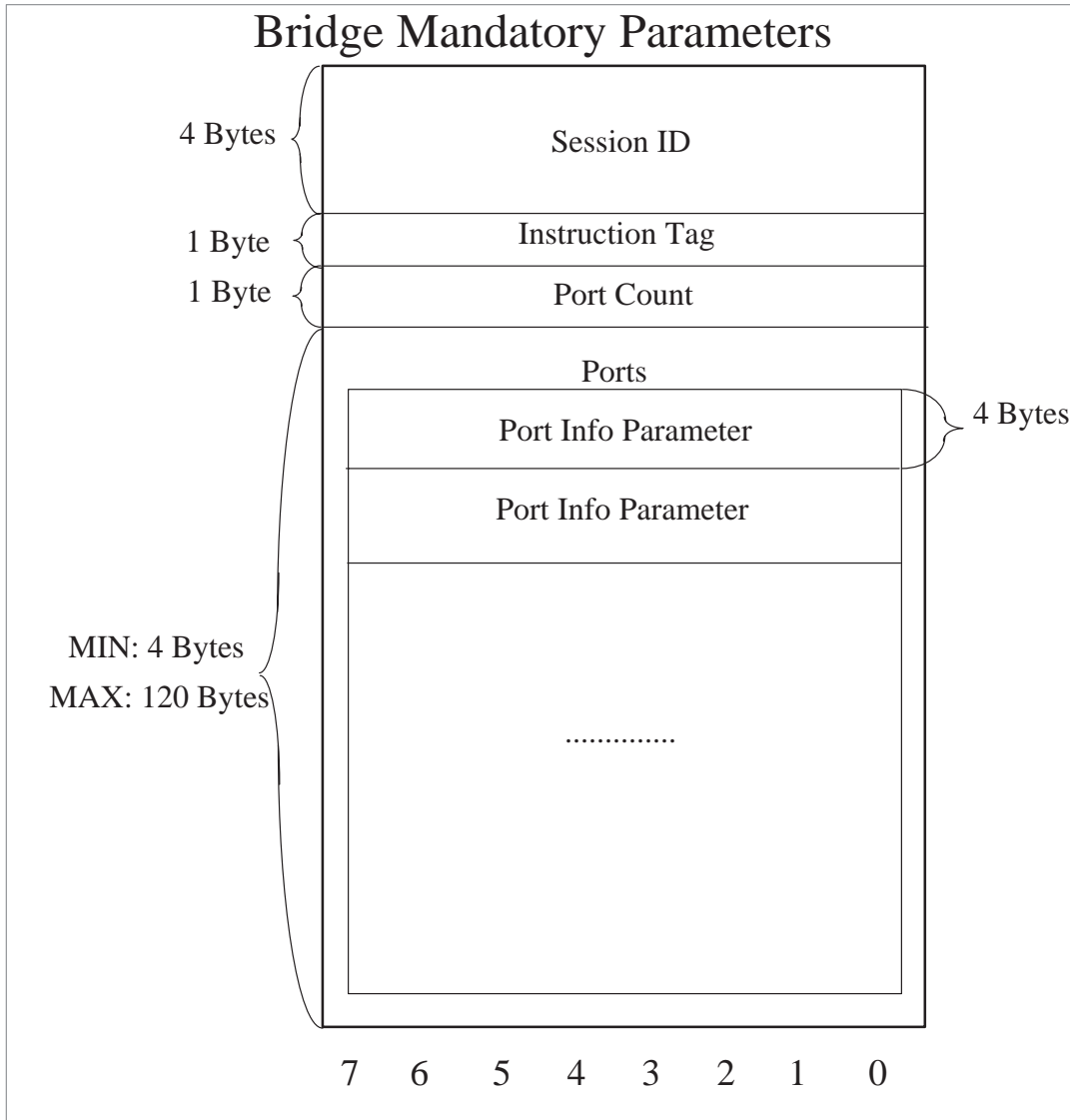
PSN LOPER Mandatory Parameters for Primitives

Note: The parameters are in Little Endian format with the least significant bit being bit 0 and the most significant bit being bit 7.

The following mandatory parameters are used to build Primitives. Their description includes the length of the parameter as well as the TYPE of each field in the parameter.

Bridge Mandatory Parameters

The mandatory parameters for a Bridge Primitive.



Length: MIN: 10 Bytes MAX: 126 Bytes

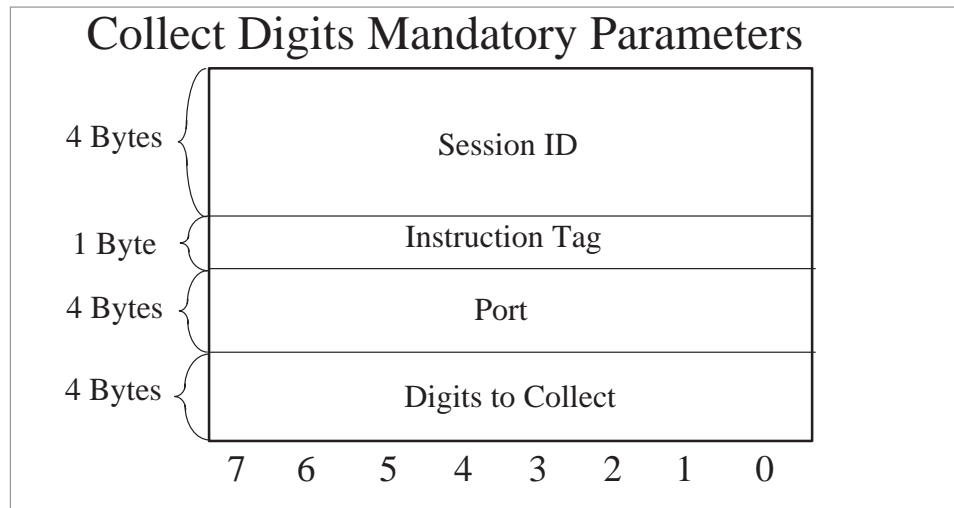
Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1Byte – *TYPE: Instruction Tag Parameter*
- **PORT COUNT:** 1 Byte – *TYPE: Count of 1 to 30*
- **PORTS:** MIN: 4 Bytes MAX: 120 Bytes – Ports consists of a minimum of 1 port to a maximum of 30 ports.

Port TYPE: Port Info Parameter

Collect Digits Mandatory Parameters

The mandatory parameters for a Collect Digits Primitive.

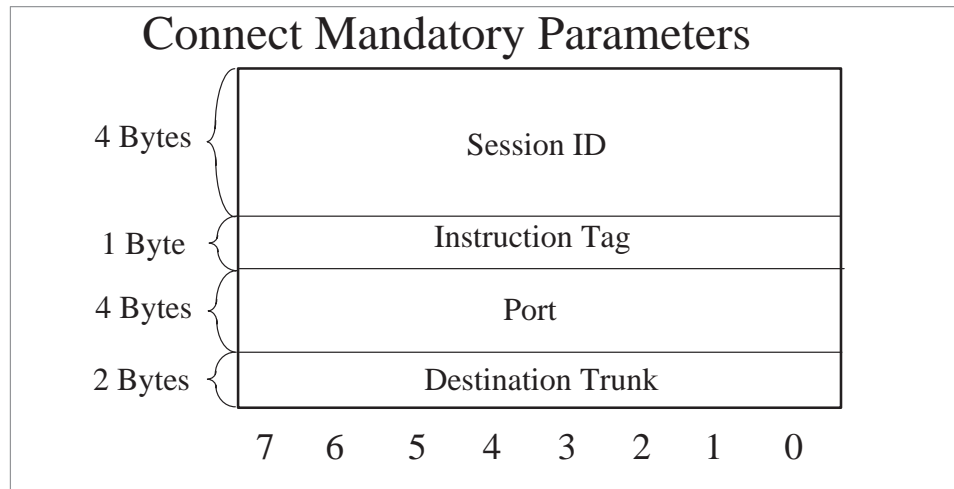


Length: 13 Bytes

- Contents:
- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **DIGITS TO COLLECT:** 4 Bytes – *TYPE: Digit Collection Parameter*

Connect Mandatory Parameters

The mandatory parameters for a Connect Primitive.



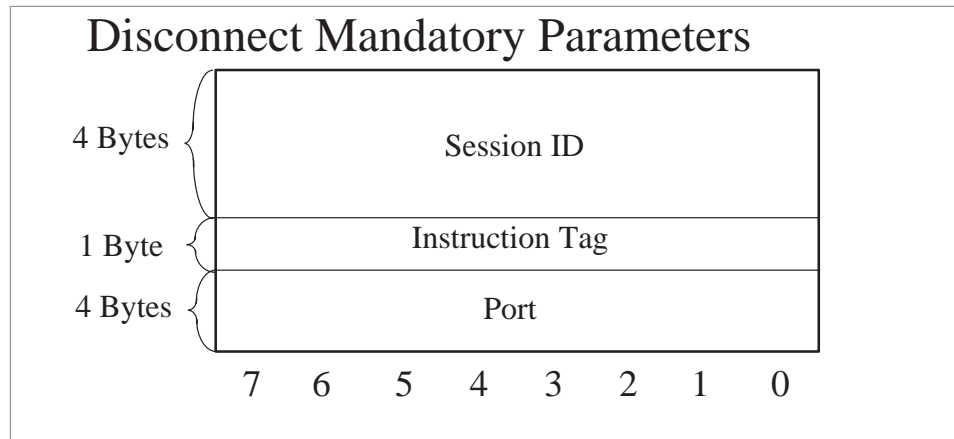
Length: 11 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **DESTINATION TRUNK:** 2 Bytes – *TYPE: Destination Trunk Group Parameter*

Disconnect Mandatory Parameters

The mandatory parameters for a Disconnect Primitive.



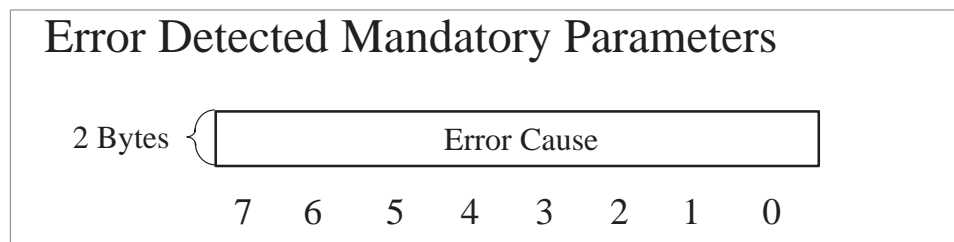
Length: 9 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

Error Detected Mandatory Parameters

The mandatory parameters for a Error Detected Primitive.



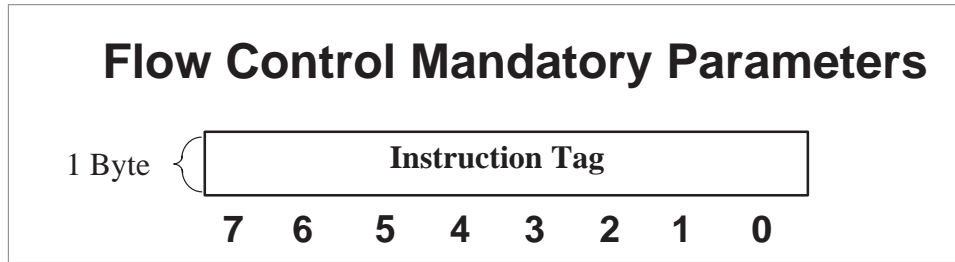
Length: 2 Bytes

Contents:

- **ERROR CAUSE:** 2 Bytes – *TYPE: Error Cause Parameter*

Flow Control Mandatory Parameters

The mandatory parameters for a Flow Control Primitive.



Length: 1 Byte

Contents:

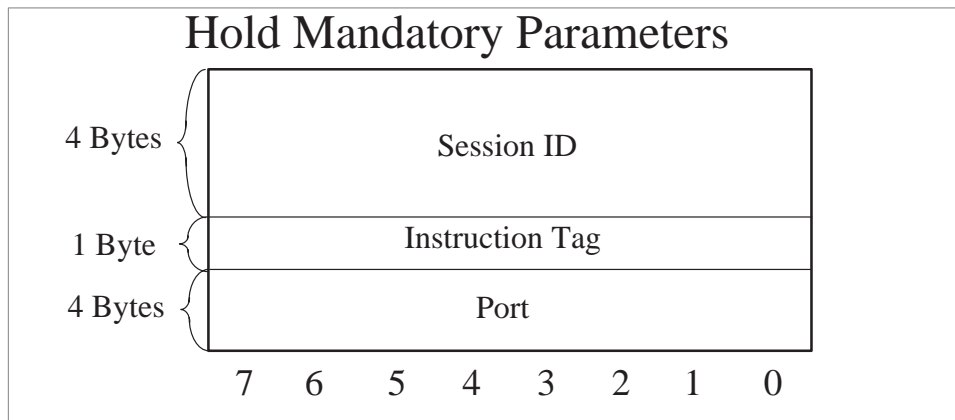
- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*

Heartbeat Mandatory Parameters

The Heartbeat Primitive does not contain any information in the primitive. A Heartbeat primitive message contains only what is mandatory for a LOPER primitive which is the Primitive Tag, the Primitive Length, and the Optional Pointer making a total of 5 bytes.

Hold Mandatory Parameters

The mandatory parameters for a Hold Primitive.



Length: 9 Bytes

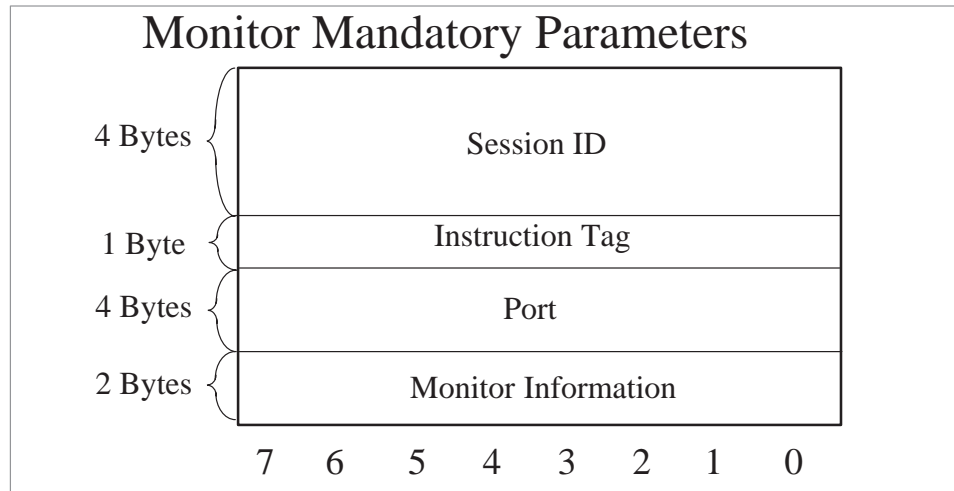
Contents:

- *SESSION ID*: 4 Bytes – *TYPE*: *Session ID Parameter*

- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE*: *Port Info Parameter*

Monitor Mandatory Parameters

The mandatory parameters for a Monitor Primitive.



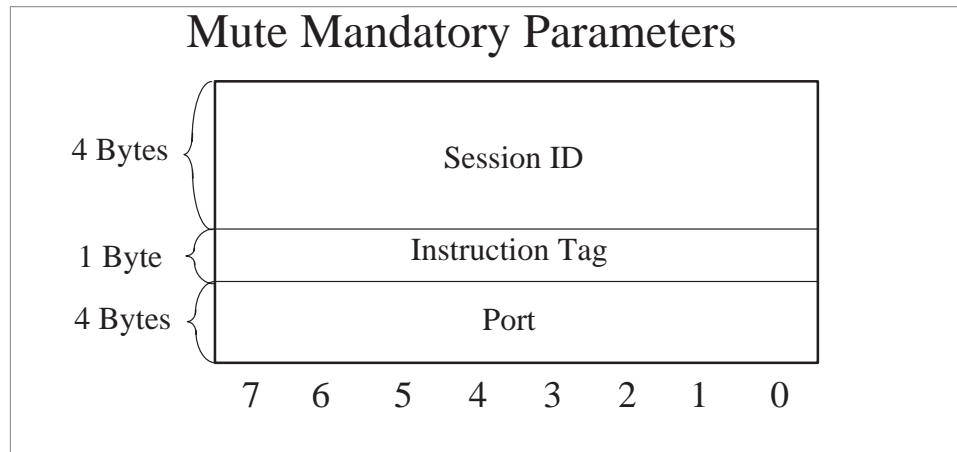
Length: 11 Bytes

Contents:

- *SESSION ID*: 4 Bytes – *TYPE*: *Session ID Parameter*
- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE*: *Port Info Parameter*
- *MONITOR INFORMATION*: 2 Bytes – *TYPE*: *Monitor Mask Parameter*

Mute Mandatory Parameters

The mandatory parameters for a Mute Primitive.



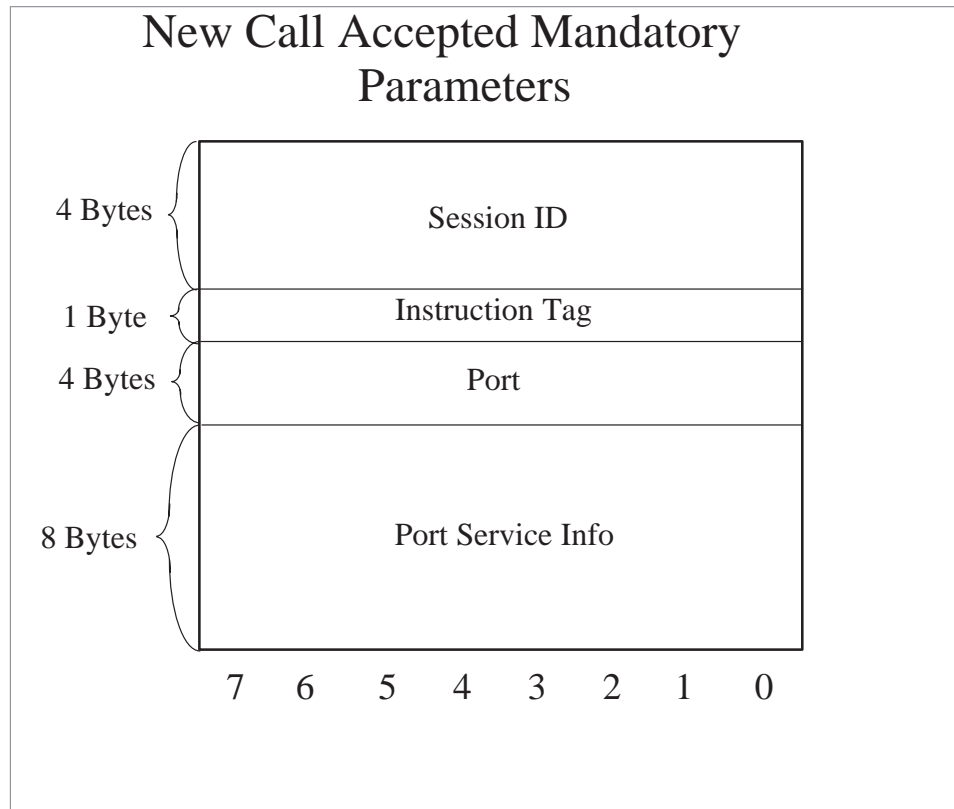
Length: 9 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

New Call Accepted Mandatory Parameters

The mandatory parameters for a New Call Accepted Primitive.



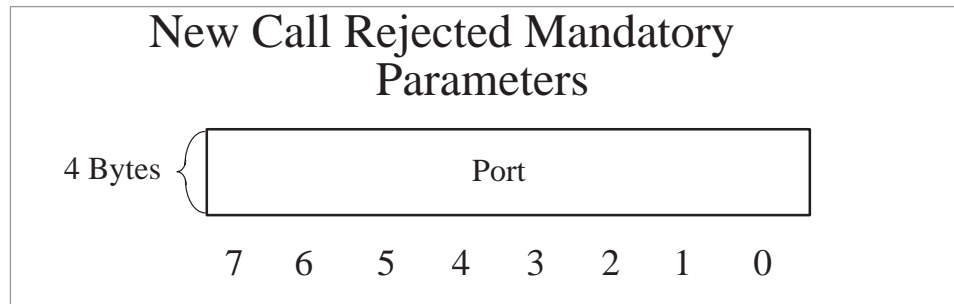
Length: 17 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **PORT SERVICE INFO:** 8 Bytes – *TYPE: Port Service Info Parameter*

New Call Rejected Mandatory Parameters

The mandatory parameters for a New Call Rejected Primitive.



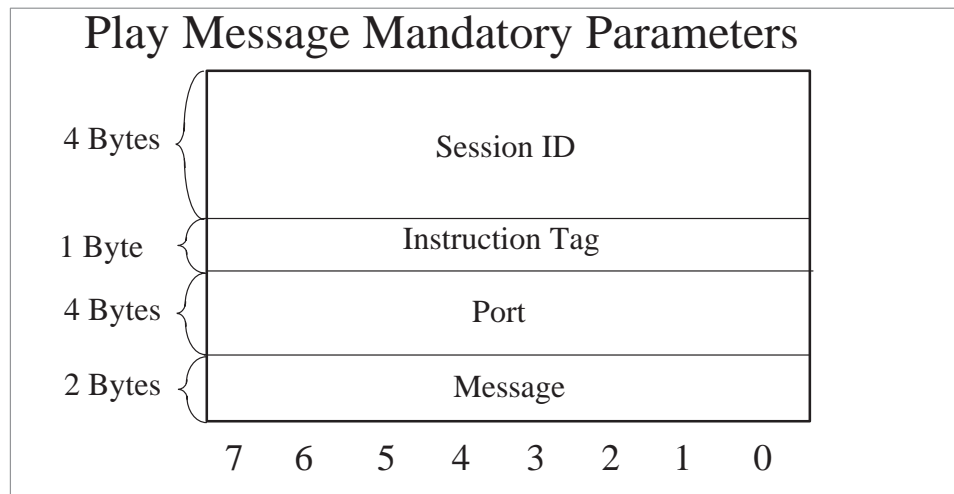
Length: 4 Bytes

Contents:

- *PORT*: 4 Bytes – *TYPE*: Port Info Parameter

Play Message Mandatory Parameters

The mandatory parameters for a Play Message Primitive.



Length: 11 Bytes

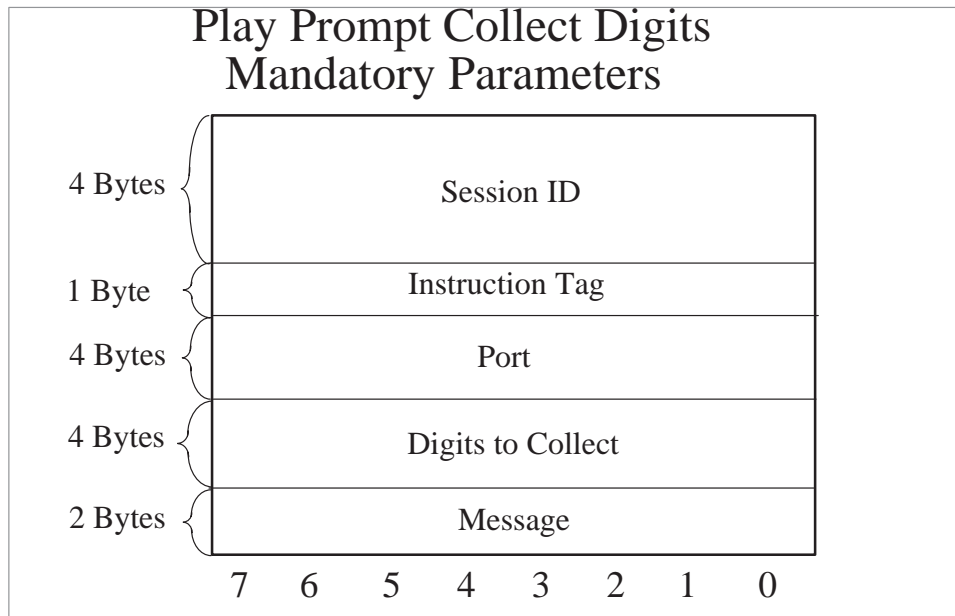
Contents:

- *SESSION ID*: 4 Bytes – *TYPE*: Session ID Parameter
- *INSTRUCTION TAG*: 1 Byte – *TYPE*: Instruction Tag ID Parameter
- *PORT*: 4 Bytes – *TYPE*: Port Info Parameter

- *MESSAGE*: 2 Bytes – *TYPE*: *Message Info Parameter*

Play Prompt Collect Digits Mandatory Parameters

The mandatory parameters for a Play Prompt Collect Digits Primitive.



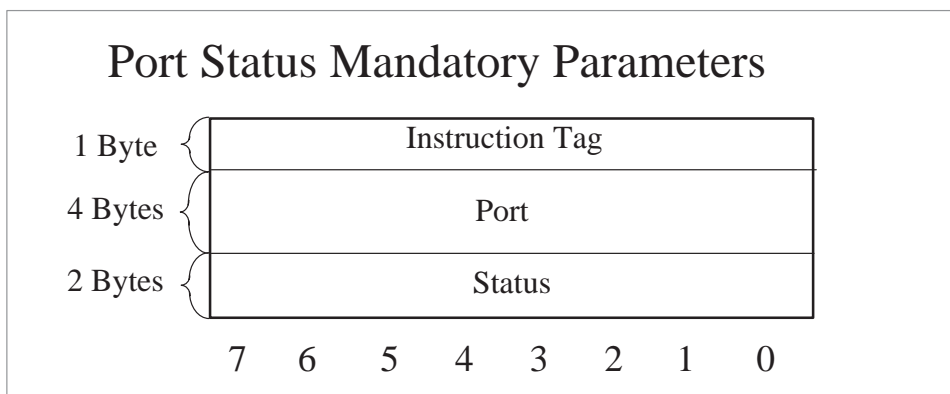
Length: 15 Bytes

Contents:

- *SESSION ID*: 4 Bytes – *TYPE*: *Session ID Parameter*
- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE*: *Port Info Parameter*
- *DIGITS TO COLLECT*: 4 Bytes – *TYPE*: *Digit Collection Parameter*
- *MESSAGE*: 2 Bytes – *TYPE*: *Message Info Parameter*

Port Status Mandatory Parameters

The mandatory parameters for a Port Status Primitive.



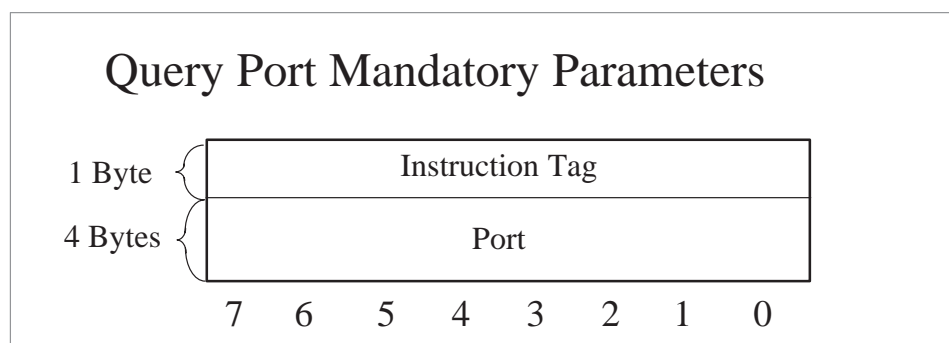
Length: 7 Bytes

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE*: *Port Info Parameter*
- *STATUS*: 2 Bytes – *TYPE*: *Port Status Parameter*

Query Port Mandatory Parameters

The mandatory parameters for a Query Port Primitive.



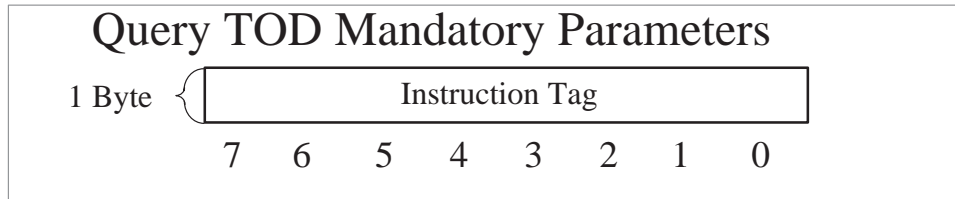
Length: 5 Bytes

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE*: *Port Info Parameter*

Query Time of Day (TOD) Mandatory Parameters

The mandatory parameters for a Query TOD Primitive.



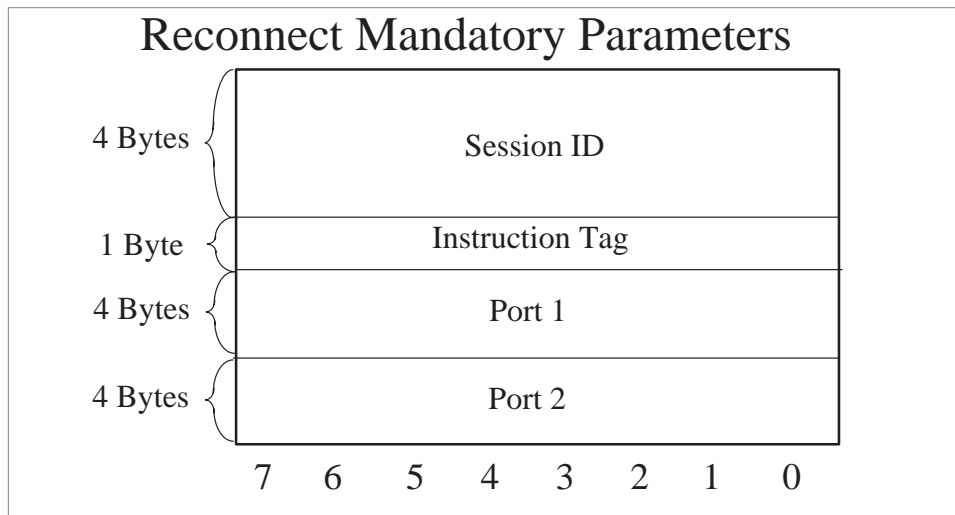
Length: 1 Byte

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*

Reconnect Mandatory Parameters

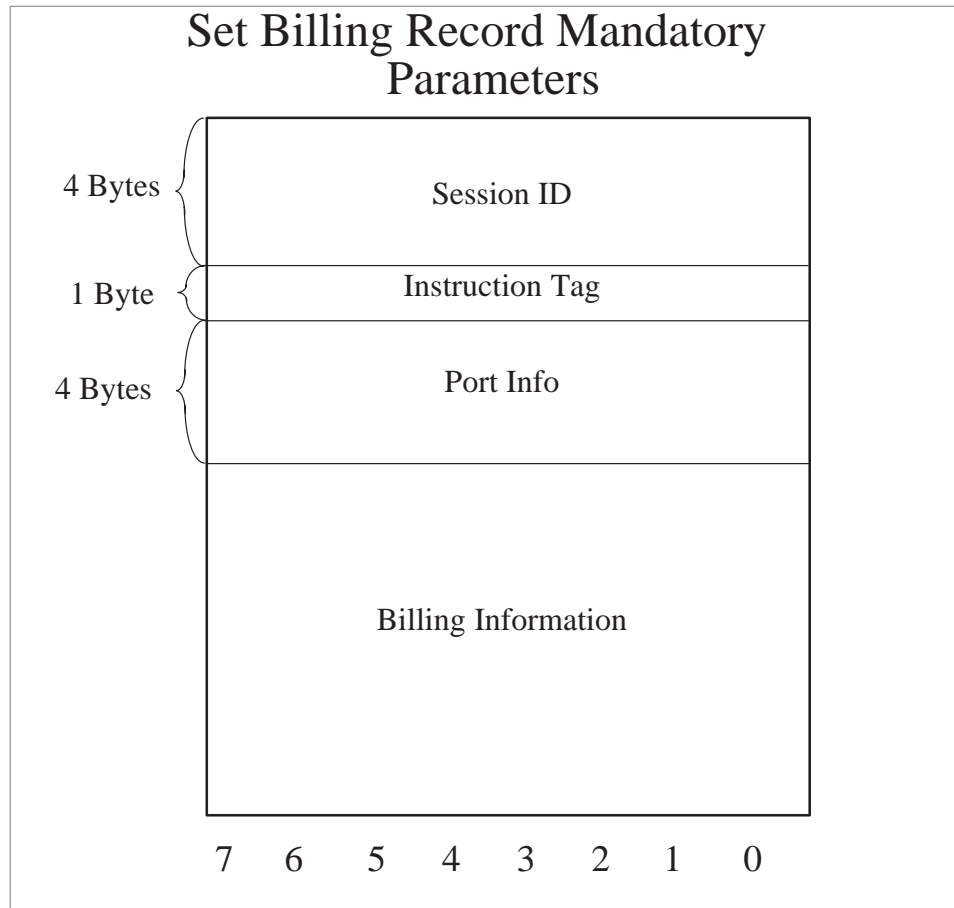
The mandatory parameters for a Reconnect Primitive.



Length: 13 Bytes

Contents:

- *SESSION ID*: 4 Bytes – *TYPE*: *Session ID Parameter*
- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *PORT 1*: 4 Bytes – *TYPE*: *Port Info Parameter*



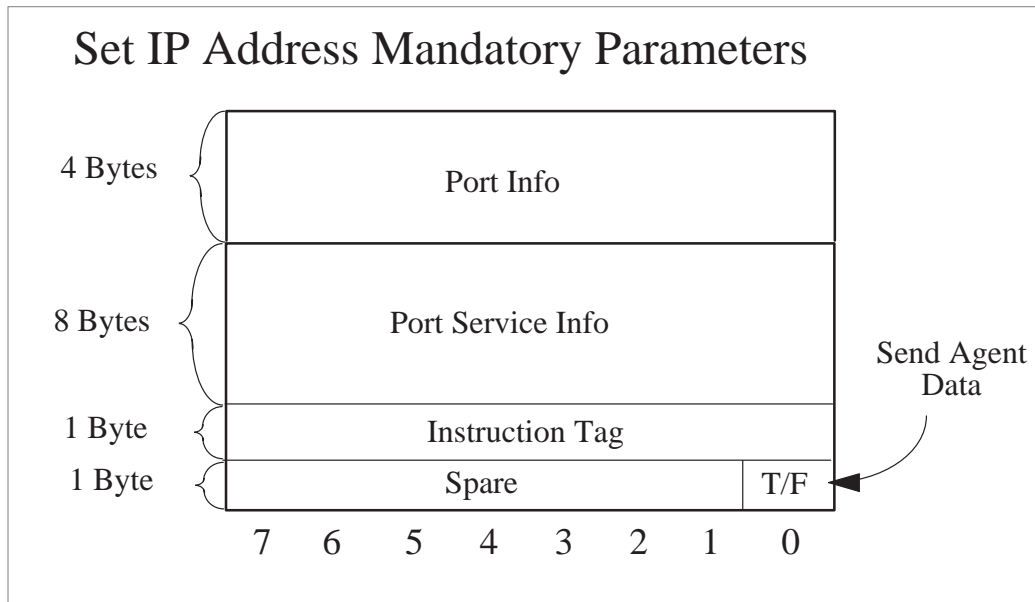
Length: Variable in Size

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **BILLING INFORMATION:** Variable in Size – *TYPE: Billing Info Parameter*

Set IP Address Mandatory Parameters

The mandatory parameters for a Set IP Address Primitive.



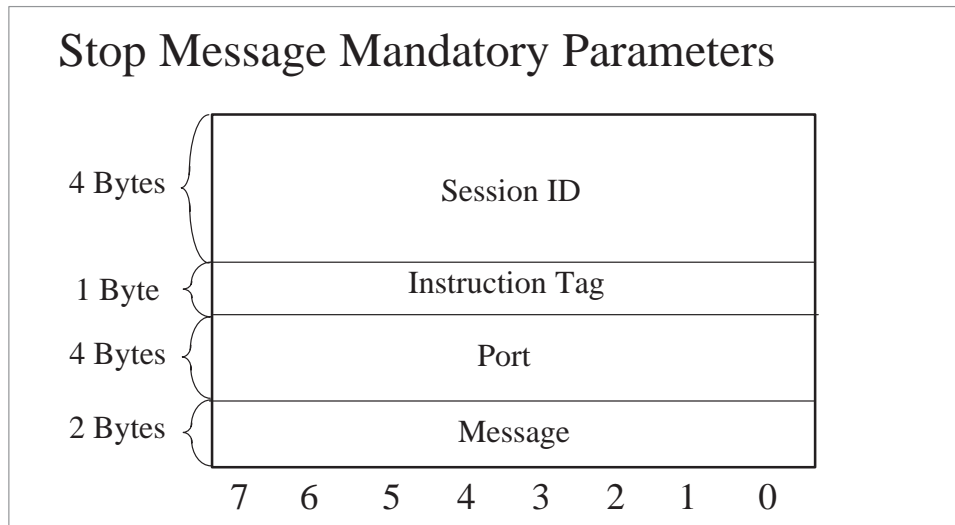
Length: 14 Bytes

Contents:

- *PORT:* 4 Bytes – *TYPE:* *Port Info Parameter*
- *PORT SERVICE INFO:* 8 Bytes – *TYPE:* *Port Service Info Parameter*
- *SEND AGENT DATA:* 1 Bit – *TYPE:* Boolean (TRUE = 1, FALSE = 0)
- *INSTRUCTION TAG:* 1 Byte – *TYPE:* *Instruction Tag ID Parameter*

Stop Message Mandatory Parameters

The mandatory parameters for a Stop Message Primitive.



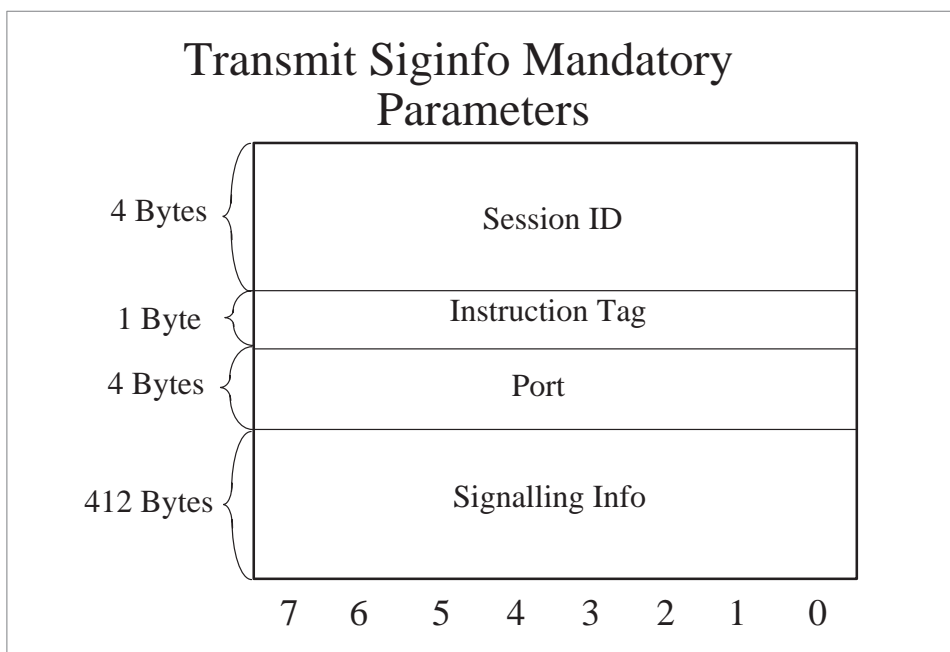
Length: 11 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **MESSAGE:** 2 Bytes – *TYPE: Message Info Parameter*

Transmit Siginfo Mandatory Parameters

The mandatory parameters for a Transmit Siginfo Primitive.



Length: 421 Bytes

Contents:

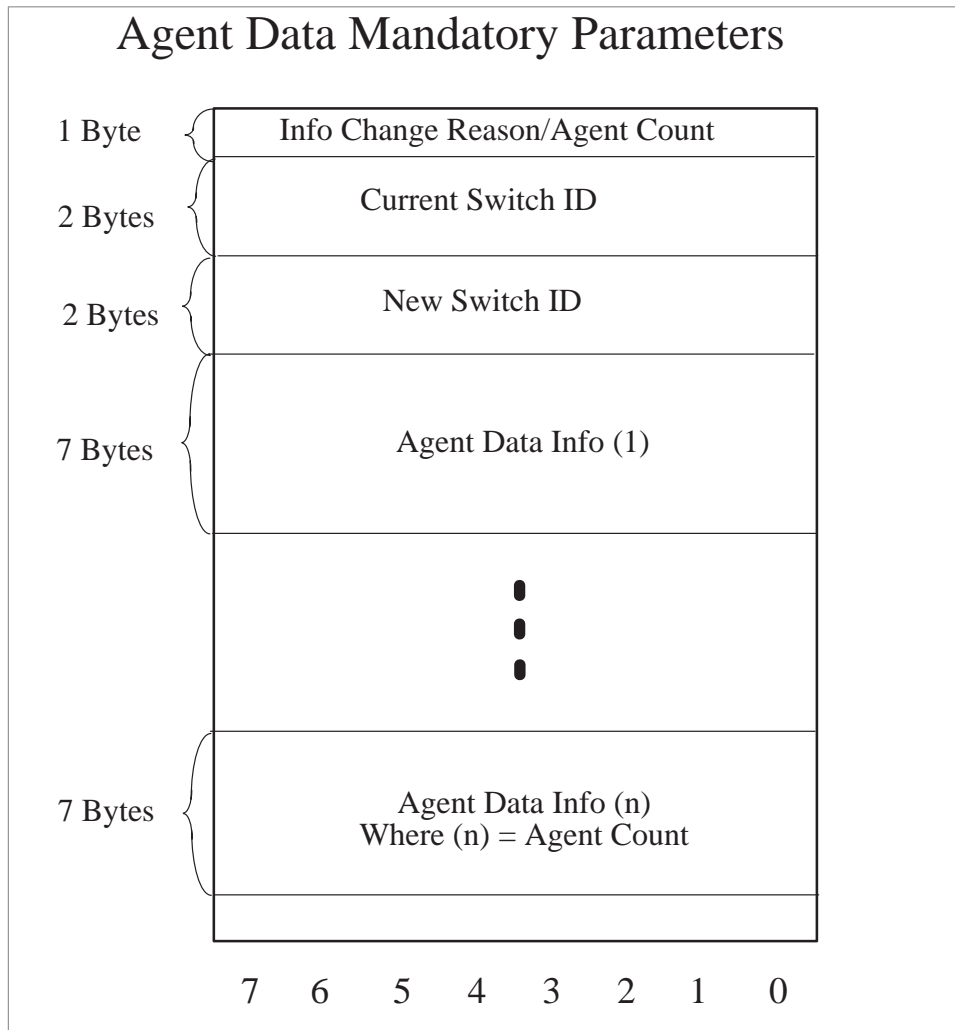
- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **SIGNALLING INFO:** 412 Bytes – *TYPE: Signalling Info Parameter*

PSN LOPER Mandatory Parameters for Events

The following mandatory parameters are used to build Event notifications. Their description includes the length of the parameter as well as the **TYPE** of each field in the parameter.

Agent Data Mandatory Parameters

The mandatory parameters for an Agent Data Event.



Length: 12 Bytes (Min) – 446 Bytes (Max)

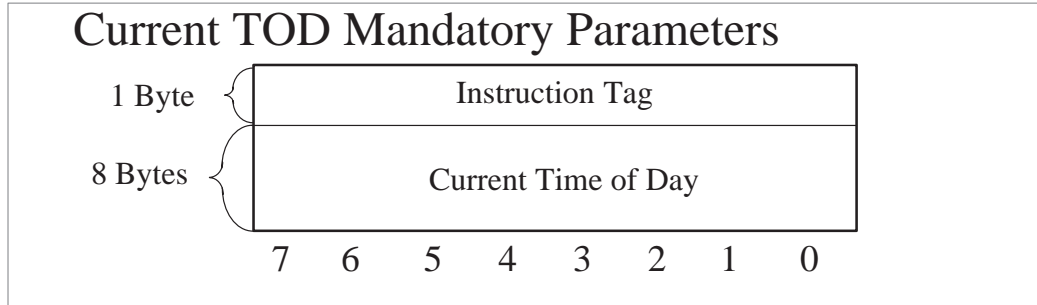
Contents:

- *Info Change Reason/Agent Count*: 1 Byte – *TYPE: Info Change Reason Parameter*
- *Current Switch ID*: 2Bytes – *TYPE: Switch ID Parameter*
- *New Switch ID*: 2Bytes – *TYPE: Switch ID Parameter*
- *Agent Data Info (1)*: 7 Bytes – *TYPE: Agent Data Info Parameter*
- *Agent Data Info (n)*: 7 Bytes – *TYPE: Agent Data Info Parameter*

Note: The number of Agent Data Info parms must equal the Agent Count Value in the first byte.

Current Time of Day (TOD) Mandatory Parameters

The mandatory parameters for a Current TOD Event.



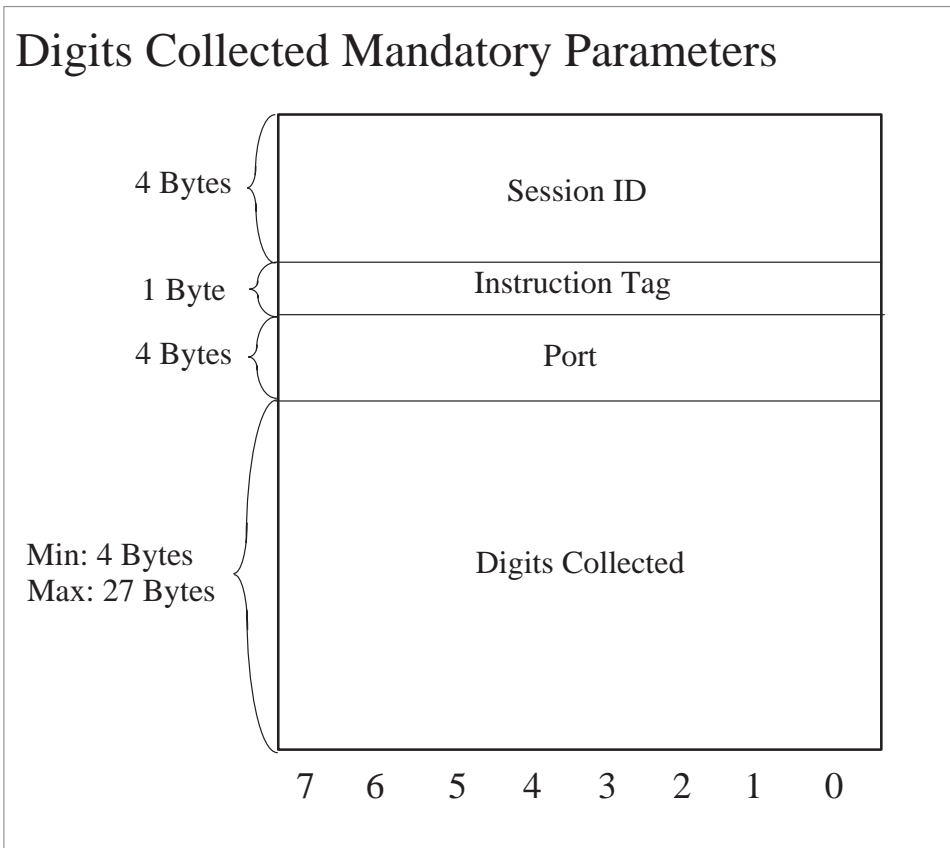
Length: 9 Bytes

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE*: *Instruction Tag ID Parameter*
- *CURRENT TIME OF DAY*: 8 Byte – *TYPE*: *Time of Day Parameter*

Digits Collected Mandatory Parameters

The mandatory parameters for a Digits Collected Event.



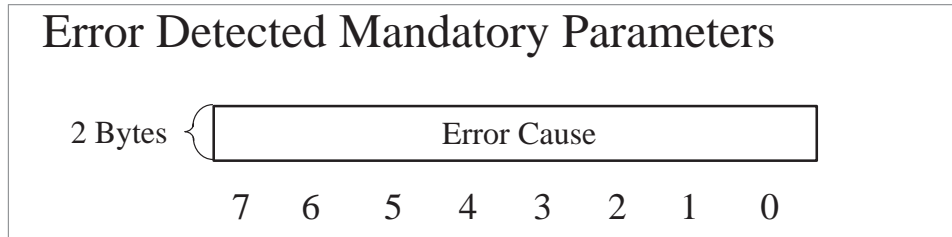
Length: Min: 13 Bytes Max: 36 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **DIGITS COLLECTED:** MIN: 4 Bytes MAX: 27 Bytes – *TYPE: Digits Collected Parameter*

Error Detected Mandatory Parameters

The mandatory parameters for a Error Detected Event.



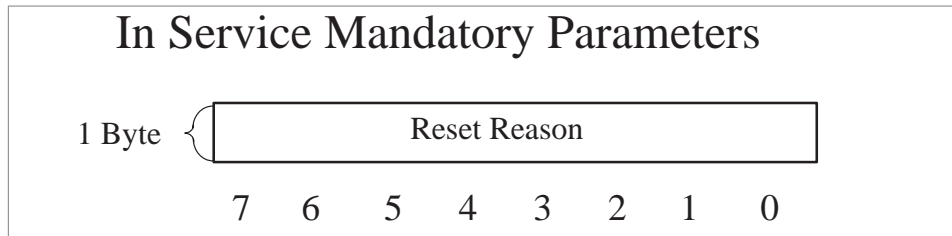
Length: 2 Bytes

Contents:

- ERROR CAUSE: 2 Bytes – *TYPE: Error Cause Parameter*

In Service Mandatory Parameters

The mandatory parameters for a In Service Event.



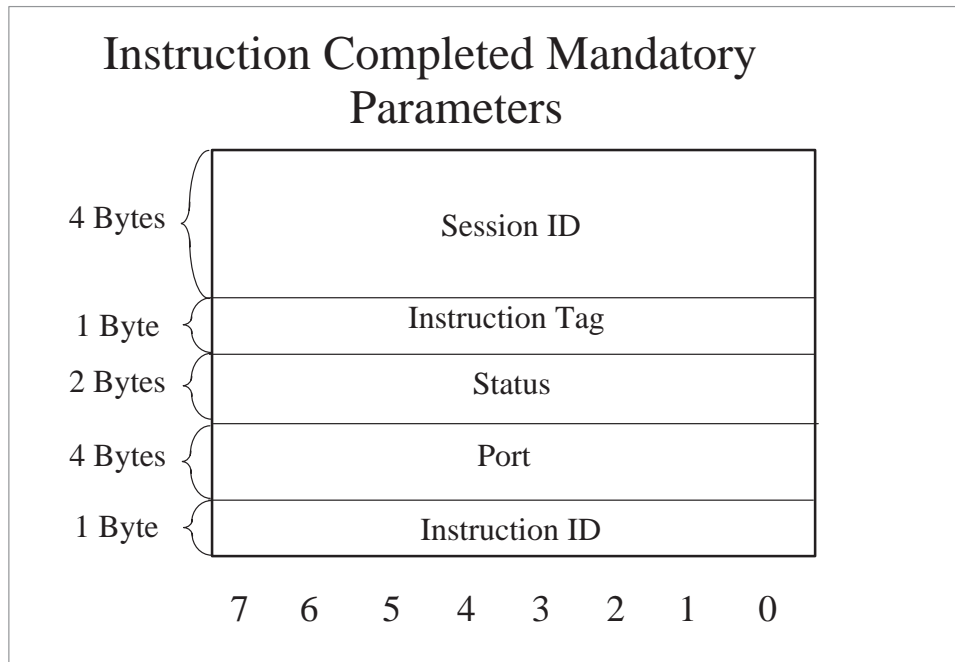
Length: 1 Byte

Contents:

- RESET REASON: 1 Byte – *TYPE: Reset Reason Parameter*

Instruction Completed Mandatory Parameters

The mandatory parameters for a Instruction Completed Event.



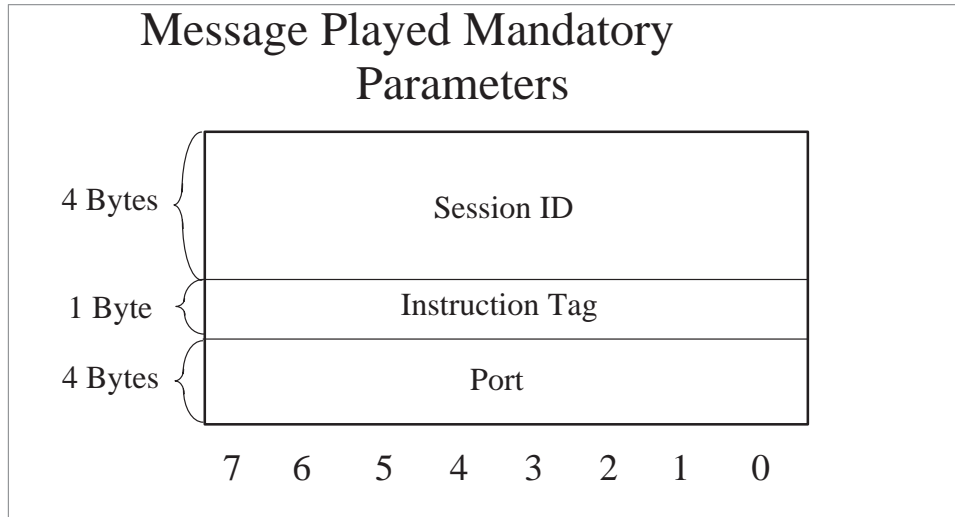
Length: 12 bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag Parameter*
- **STATUS:** 2 Bytes – *TYPE: Port Status Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **INSTRUCTION ID:** 1 Byte – *TYPE: Instruction ID Parameter*

Message Played Mandatory Parameters

The mandatory parameters for a Message Played Event.



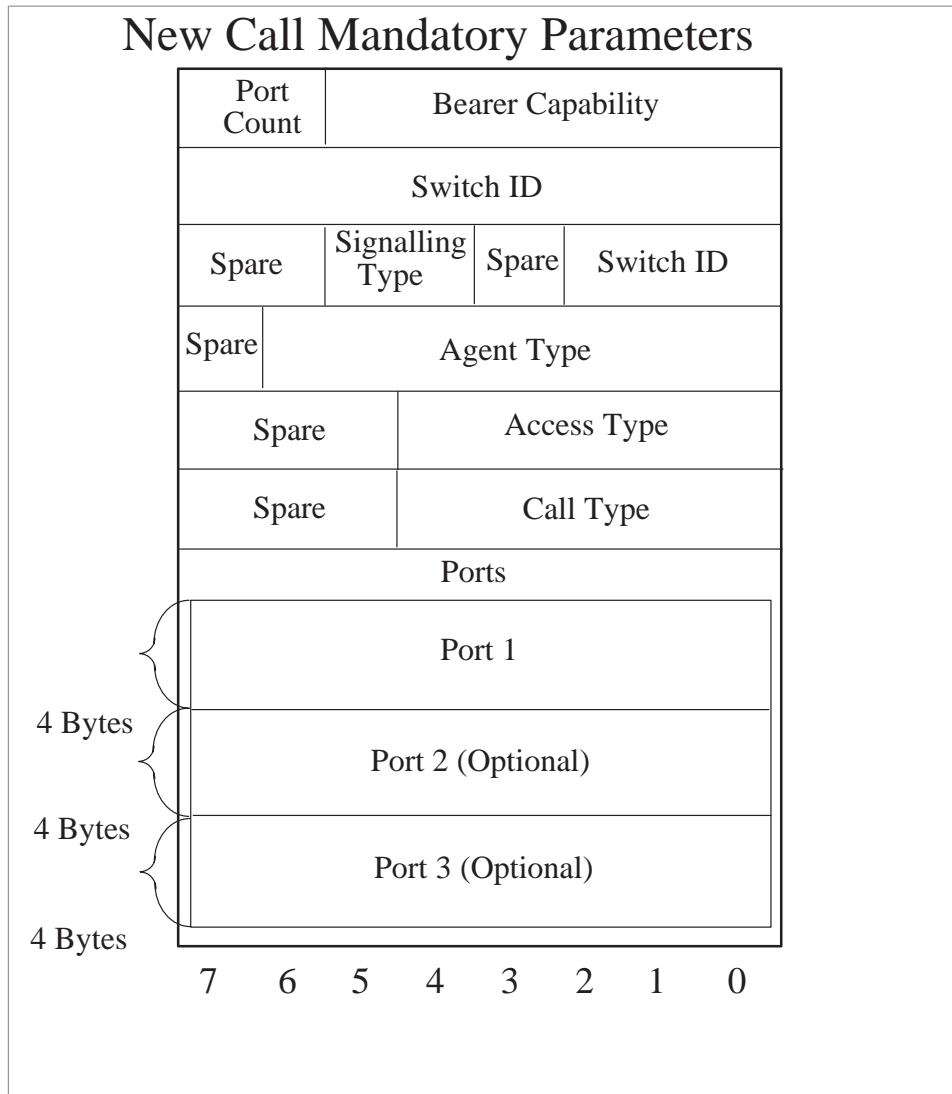
Length: 9 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

New Call Mandatory Parameters

The mandatory parameters for a New Call Event.



Parameter Length: MIN: 10 Bytes MAX: 18 Bytes

Parameter Contents:

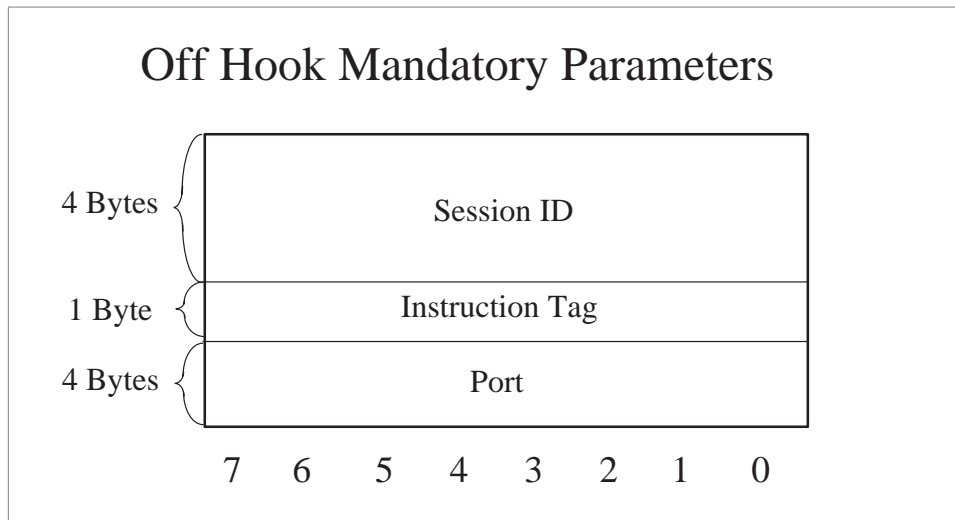
- *BEARER CAPABILITY*: 6 Bits – *TYPE: Bearer Capability Parameter*
- *PORT COUNT*: 2 Bits – *VALUES INCLUDE: 0 to 3.*
- *SWITCH ID*: 7 Bits – *TYPE: Switch ID Parameter*
- *SIGNALLING TYPE* : 1 BYTE – *TYPE: Signalling Type Parameter*
- *AGENT TYPE*: 1 BYTE – *TYPE: Agent Type Parameter*
- *ACCESS TYPE*: 5 Bits – *TYPE: Access Type Parameter*

- CALL TYPE: 5 Bits – *TYPE: Call Types Parameter*
- PORTS: Ports consists of a minimum of 1 port to a maximum of 3 ports.

Port TYPE: Port Info Parameter

Off Hook Mandatory Parameters

The mandatory parameters for an Off Hook Event.



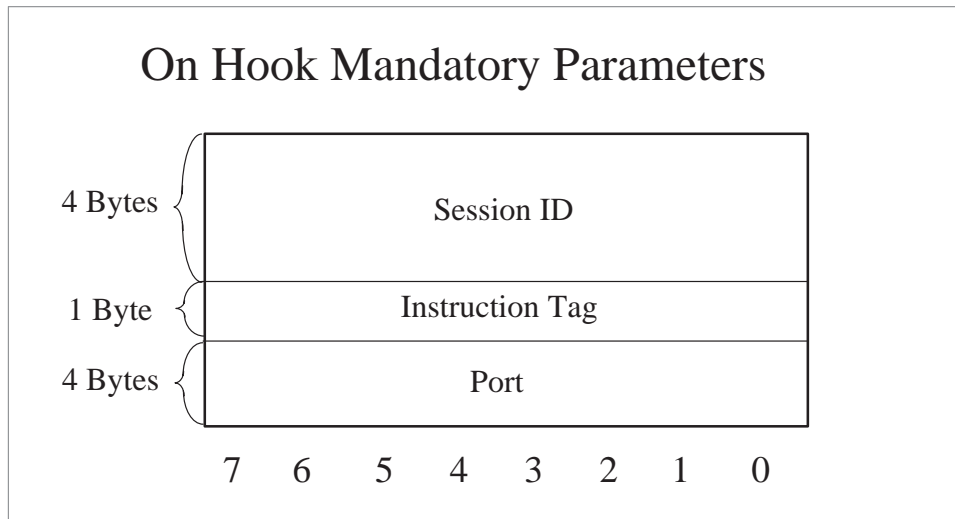
Length: 9 Bytes

Contents:

- SESSION ID: 4 Bytes – *TYPE: Session ID Parameter*
- INSTRUCTION TAG: 1 Byte – *TYPE: Instruction Tag ID Parameter*
- PORT: 4 Bytes – *TYPE: Port Info Parameter*

On Hook Mandatory Parameters

The mandatory parameters for an On Hook Event.



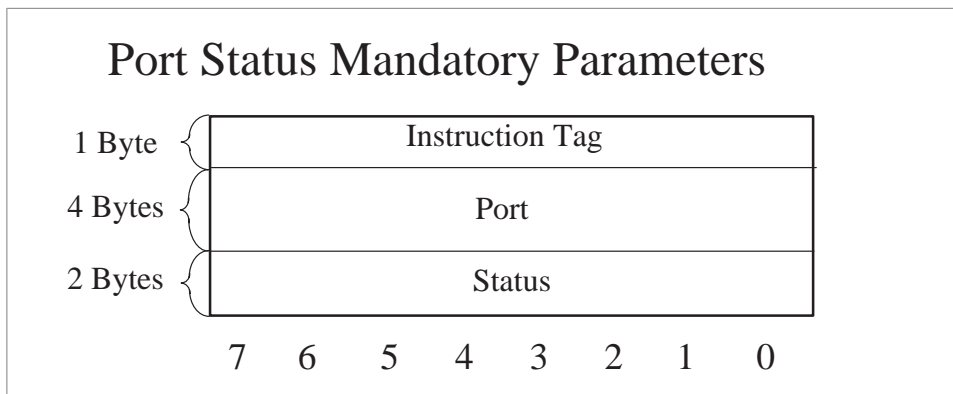
Length: 9 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

Port Status Mandatory Parameters

The mandatory parameters for a Port Status Event.



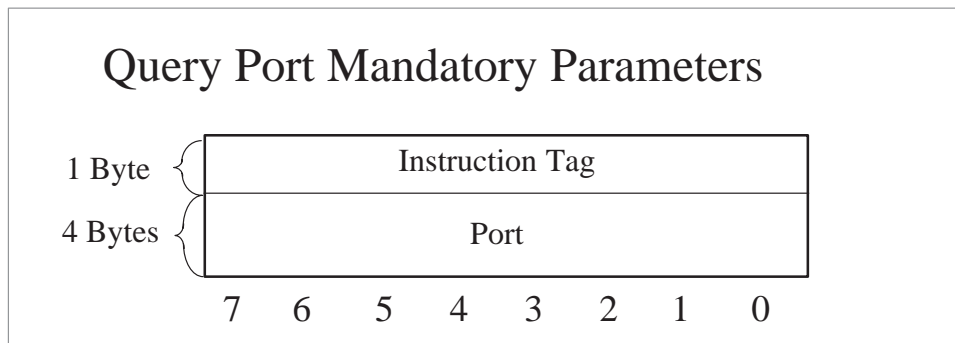
Length: 7 Bytes

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE: Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE: Port Info Parameter*
- *STATUS*: 2 Bytes – *TYPE: Port Status Parameter*

Query Port Mandatory Parameters

The mandatory parameters for a Query Port Event.



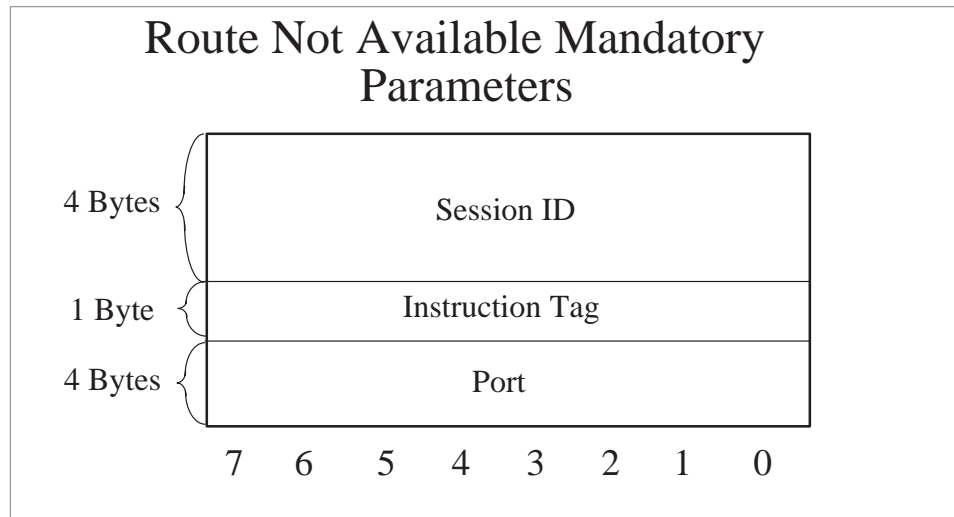
Length: 5 Bytes

Contents:

- *INSTRUCTION TAG*: 1 Byte – *TYPE: Instruction Tag ID Parameter*
- *PORT*: 4 Bytes – *TYPE: Port Info Parameter*

Route Not Available Mandatory Parameters

The mandatory parameters for a Route Not Available Event.



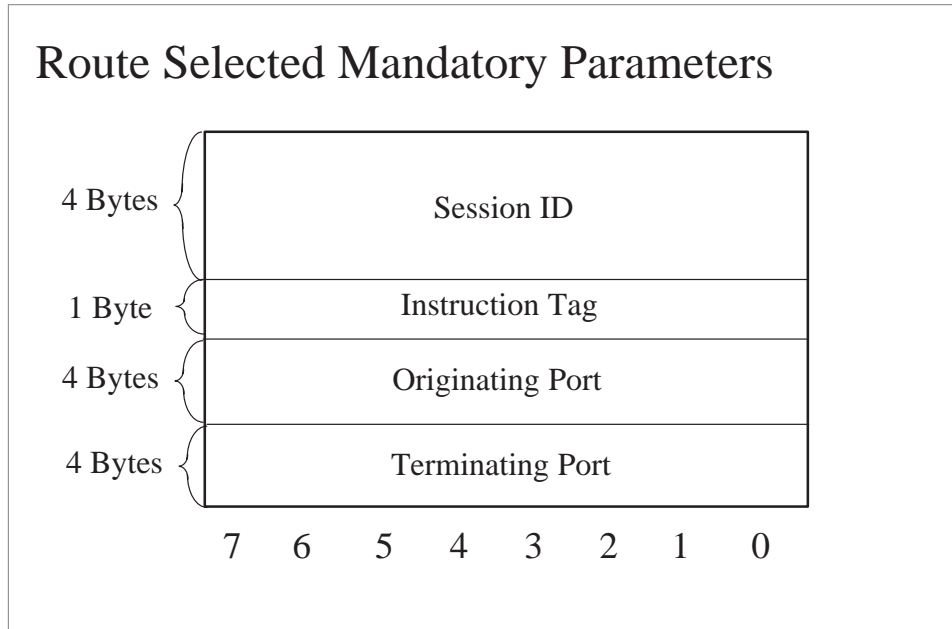
Length: 9 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

Route Selected Mandatory Parameters

The mandatory parameters for a Route Selected Event.



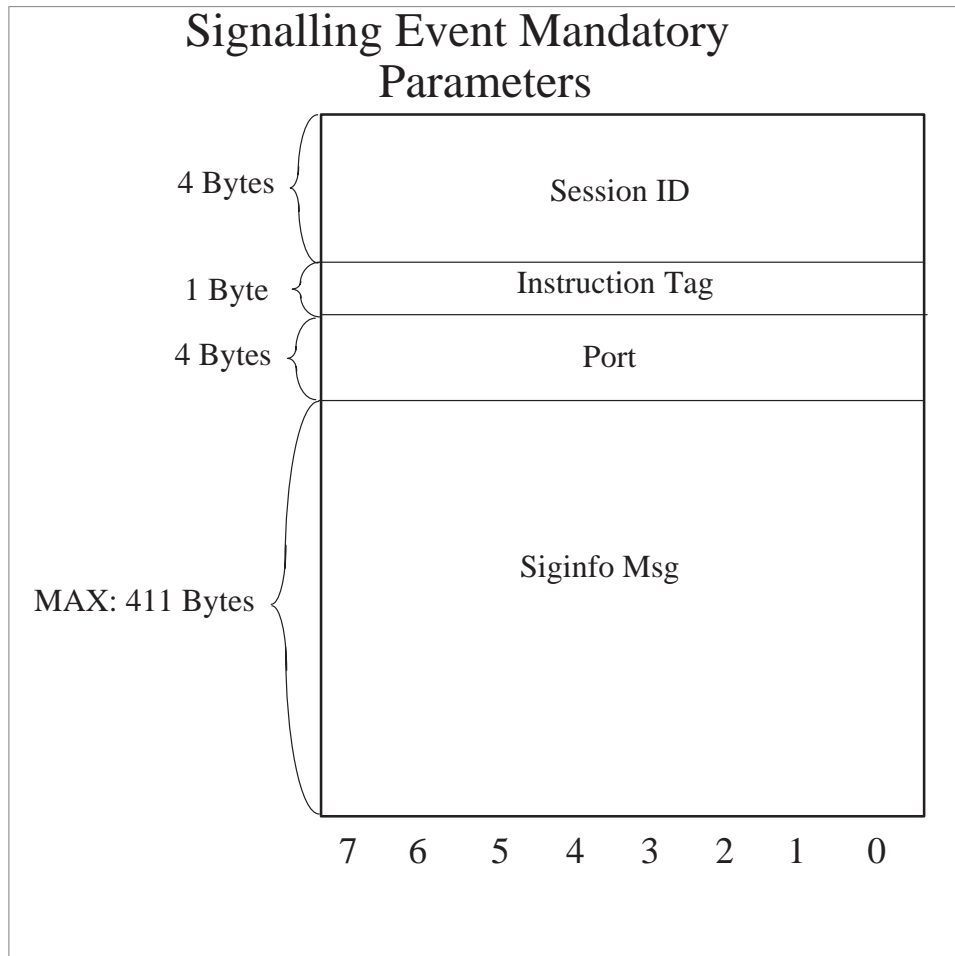
Length: 13 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **ORIGINATING PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **TERMINATING PORT:** 4 Bytes – *TYPE: Port Info Parameter*

Signalling Event Mandatory Parameters

The mandatory parameters for a Signalling Event.



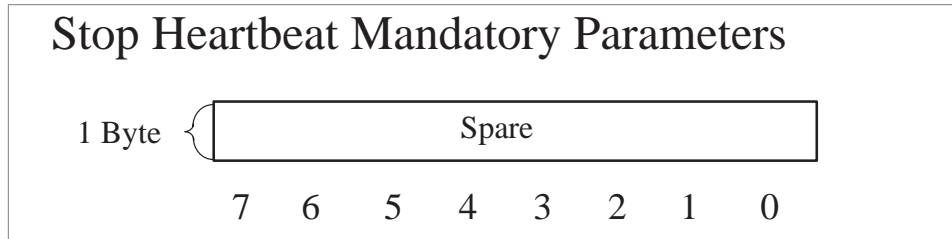
Length: MAX 420 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*
- **SIGINFO MSG:** MAX: 411 Bytes – *TYPE: Signalling Info Parameter*

Stop Heartbeat Mandatory Parameters

The mandatory parameters for an Stop Heartbeat Event.



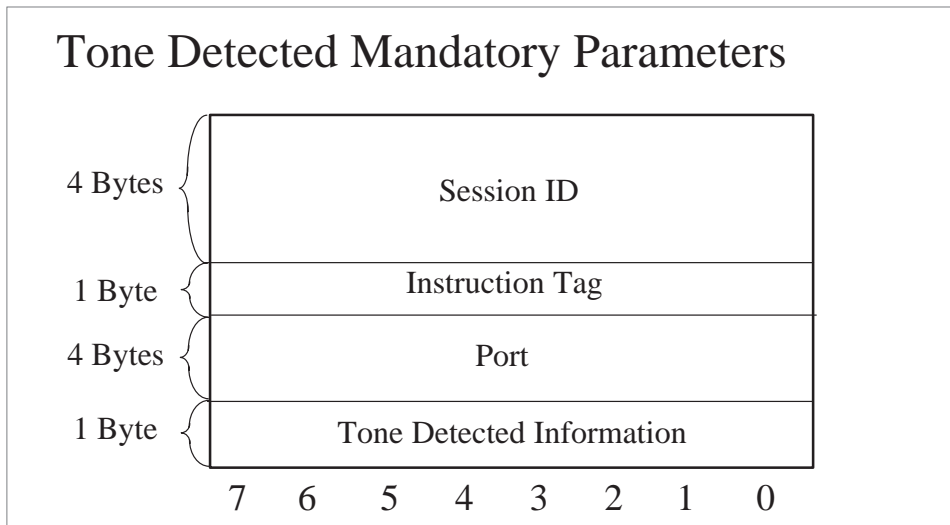
Length: 1 Byte

Contents:

- Spare: 1 Byte – *To make message word aligned.*

Tone Detected Mandatory Parameters

The mandatory parameters for a Tone Detected Event.



Length: 10 Bytes

Contents:

- **SESSION ID:** 4 Bytes – *TYPE: Session ID Parameter*
- **INSTRUCTION TAG:** 1 Byte – *TYPE: Instruction Tag ID Parameter*
- **PORT:** 4 Bytes – *TYPE: Port Info Parameter*

- *TONE DETECTED INFORMATION*: 1 Byte – *TYPE: Tone Detected Parameter*

Appendix A

PSN for UCS DMS-250 switch

This appendix describes the Programmable Service Node (PSN) specifically for the UCS DMS-250 switch. This appendix documents only the differences between the PSN Platform (described in the main chapters of this document) and the PSN for the UCS DMS-250 switch. The PSN for UCS DMS-250 feature builds on top of the strategic PSN platform; it implements UCS DMS-250 specific triggering mechanisms and specific parameters to provide a complete PSN platform for UCS DMS-250 customers.

These systems do not offer the reliability or the capacity required for truly robust performance. Also, the different computing platforms introduced into the network to support these overlay systems introduce maintenance and network management complexities, and increase operations and sustaining costs.

The SCU platform provides service control capabilities. A high-speed data link connects the SCU and the PSN. Possible high speed data links include IEEE 802.3 Ethernet (the current offering), FDDI, ATM, and other protocols.

Entering Server Mode

An existing UCS call follows in-switch logic until it encounters a CAIN Trigger Detection Point (TDP). It can proceed as a CAIN call and query the AIN SCP, or become a PSN call and query the SCU.

Refer to the *UCS DMS-250 NetworkBuilder Application Guide* for more details.

Specifications

For a potential PSN call, the CAIN trigger database is accessed at a TDP based on the Point in Call (PIC). There are two methods to become a PSN call.

- 1 One way to become a PSN call is to datafill the CAIN trigger database tables so that the trigger action is to query the SCU. This is done by datafilling the CAINGRP option on the originating trunk group from table TRKGRP, the ANI in table ANISCUSP, the CAIN_OFFICE_GROUP in table OFCPARM, or by the authcode in table AUTHCODU. For PRI agents, the CAINGRP option must be datafilled in table CALLATTR. For more information on the CAIN Trigger Tables and Office Params, refer to the *UCS DMS-250 NetworkBuilder Application Guide*.

A new action, Query the SCU (QUERYSCU), is added to the CAIN trigger database tables CUSTDP, OFFHKIMM, PRIBCHNL, SIOTRK, and SPECDIG. When this action is obtained from accessing the database, it results in the call entering server mode and becoming a PSN call. Normal UCS DMS-250 call processing actions are suspended when a call becomes a PSN call.

When the QUERYSCU action is encountered, in-switch call processing is suspended and the **New_Call** event notification message is sent to the SCU. If an in-switch error occurs before all digits are present, the call goes to treatment and does not become a PSN call, as it would have under the existing CAIN functionality.

- 2 The second way to become a PSN call is to become an AIN call, query the AIN SCP and receive a response from the AIN SCP to query the SCU.

In both cases, a **New_Call** event notification is sent to the SCU. The information contained in a **New_Call** event notification is as follows:

- Switch ID: from the office parameter ORIG_SWITCH_ID (mandatory)
- Trunk port: trunk group and trunk member information for the port involved in the call when it becomes a PSN call (mandatory)
- Access Type: the access type of the port involved in the call. (Mandatory)
- Call Type: the call type of the call. (Mandatory)
- Bearer Capability: the bearer capability of the call. (Mandatory)
- Signalling Type: whether the agency is PTS 2-wire, PTS 4-wire, SS7 or PRI. (Mandatory)

- Agent Type: the TRKGRP type of the originator. For example, FGD, IMT or DAL. (Mandatory)
- Digits information collected (optional):
 - ANI: from ANI, PANI, or CLID digits received or determined by the UCS DMS-250, including the ID digits.
 - AUTHCODE: The filed or dialed authorization digits.
 - PIN: The Personal Identification Number received by the UCS DMS-250 associated with the AUTHCODE or ANI processing.
 - Account Code: The account code received by the UCS DMS-250 associated with the AUTHCODE, ANI, or MCCSCARD processing.
 - Called Number: The number the UCS DMS-250 would ordinarily use for translations. Example: dialed number.
 - STS, OPart and TPart—The partition information the UCS DMS-250 would normally use for routing.
- Point In Call: the point in call (PIC) / Trigger Criteria combination active when it was determined that the SCU was to be queried (Optional)
- ISUP Index: the ISUP Index of the SS7 originator (Optional)
- COT Required: the COT required field value in Table TRKSGRP for the SS7 originator (Optional)

Signaling protocol

The following sections contain information on the PSN signaling protocol.

TCAP signaling

The indication parameter, *ConnectToScu*, allows the AIN SCP to specify that an agent is to be routed to the SCU. This parameter is added to the CAIN existing TCAP message, *Continue*. If *ConnectToScu* is present in the extension parameter of the *Continue* message, then the UCS DMS-250 terminates CAIN call processing and begins PSN processing by sending the call to the SCU. For more information on this extension parameter, refer to the *UCS DMS-250 NetworkBuilder Application Guide*.

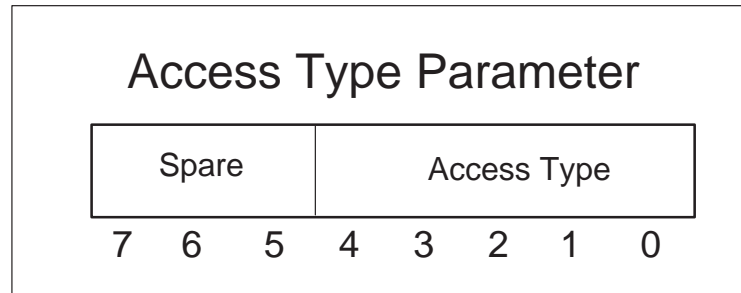
PSN signaling

PSN messages and parameters (in the LOPER format) are described in more detail in the main chapters of this document. Most of the messages and parameters sent to the SCU or received from the SCU do not require modifications for this feature.

The parameters that have different values from the ones defined as the strategic PSN values are in the following sections.

Access Type

This parameter contains the combination of access and trunk type information. This information is sent in the NEW CALL event notification and contains the originating trunk information.



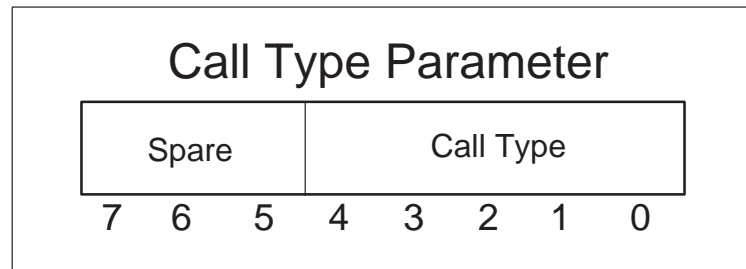
Parameter Length: **1 Byte**

Parameter Contents:

- **ACCESS TYPE** : 5 Bits. The access type of the port, which is encoded as follows:
 - 0 0 0 0 0 : Unused (Spare)
 - 0 0 0 0 1 : PTS FGD
 - 0 0 0 1 0 : SS7 FGD
 - 0 0 0 1 1 : DAL 4-wire
 - 0 0 1 0 0 : DAL 2-wire
 - 0 0 1 0 1 : PRI
 - 0 0 1 1 0 to 1 1 1 1 1 : Unused (Spare)

Call Type

This parameter contains the call type for the call when it is determined by the PSN that the call is to be controlled by the SCU.



Mandatory Parameter for a New Call event.

Parameter Length: **1 Byte**

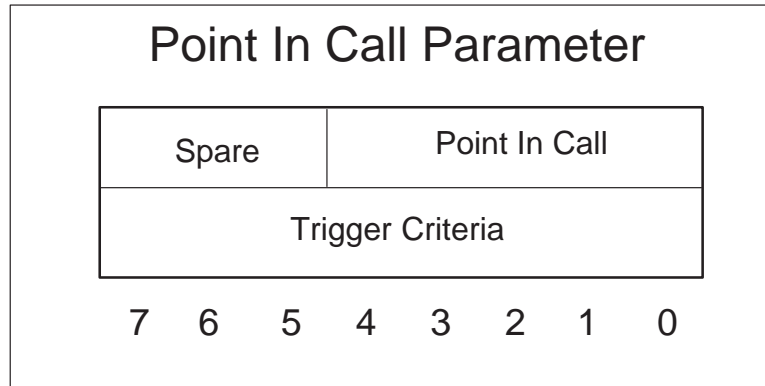
Parameter Contents:

- **CALL TYPE** : 5 Bits. The call type gives information on the type of call being made. The values defined currently include:
 - 0 0 0 0 0 : Undetermined *
 - 0 0 0 0 1 : Onnet *
 - 0 0 0 1 0 : Offnet *
 - 0 0 0 1 1 : Public Speed
 - 0 0 1 0 0 : Private Speed
 - 0 0 1 0 1 : Hotline Speed
 - 0 0 1 1 0 : N00*
 - 0 0 1 1 1 : Zero Plus – Onnet
 - 0 1 0 0 0 : Zero Plus – Offnet
 - 0 1 0 0 1 : INTOA *
 - 0 1 0 1 0 to 1 1 1 1 1 : Spare Values

The values marked with an “*” are the only values that are currently supported.

Point In Call

This parameter contains the point in the call when all criteria were met and the PSN gave the SCU control over the call.



Optional Parameter Tag: **130**

Parameter Length: **2 Byte**

Parameter Contents:

- POINT IN CALL: 5 bits. This field contains the point in call, at which time the PSN determines that the call is a service call and needs to be controlled by the SCU. It is encoded as follows:
 - 0 0 0 0 0 : Not Used
 - 0 0 0 0 1 : Orig Null
 - 0 0 0 1 0 : Authorize Orig Attempt
 - 0 0 0 1 1 : Collect Information
 - 0 0 1 0 0 : Analyze Information*
 - 0 0 1 0 1 : Select Route
 - 0 0 1 1 0 : Authorize Call Setup
 - 0 0 1 1 1 : Send Call
 - 0 1 0 0 0 : Orig Alerting
 - 0 1 0 0 1 : Orig Active
 - 0 1 0 1 0 : Orig Suspended
 - 0 1 0 1 1 : Term Null
 - 0 1 1 0 0 : Authorize Termination

- 0 1 1 0 1 : Select Facility
- 0 1 1 1 0 : Present Call
- 0 1 1 1 1 : Term Alerting
- 1 0 0 0 0 : Term Active
- 1 0 0 0 1 : Term Suspended
- 1 0 0 1 0 to 1 1 1 1 1 : Spare Values

The values marked with an “*” are the only values that are currently supported.

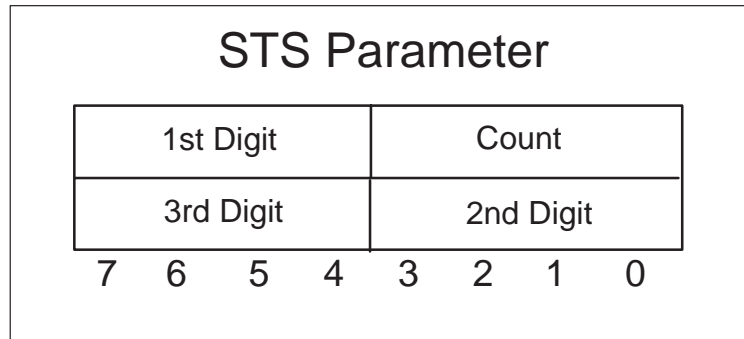
- TRIGGER CRITERIA: 8 Bits. This field is encoded as follows:

- 0 0 0 0 0 0 0 0 : Feature Activator
- 0 0 0 0 0 0 0 1 : Vertical Service Code
- 0 0 0 0 0 0 1 0 : Customized Access
- 0 0 0 0 0 0 1 1 : Customized Intercom*
- 0 0 0 0 0 1 0 0 : NPA
- 0 0 0 0 0 1 0 1 : NPA_NXX
- 0 0 0 0 0 1 1 0 : NXX
- 0 0 0 0 0 1 1 1 : NXX_XXXX
- 0 0 0 0 1 0 0 0 : NPA_NXXXXXXX*
- 0 0 0 0 1 0 0 1 : Country_Code_NPA_NXXXXXXX*
- 0 0 0 1 0 0 0 0 : Offhook Immed
- 0 0 0 1 1 0 0 0 : Net Busy
- 0 0 0 1 1 0 1 1 : Orig Called Party Busy
- 0 0 0 1 1 1 0 1 : Orig No Answer
- 0 0 1 0 0 0 0 0 : Orig Feature Activator
- 0 1 1 0 0 0 0 0 : Channel Setup PRI CLID
- 0 1 1 0 0 0 0 1 : Channel Setup PRI Addr
- 0 1 1 0 0 0 1 0 : Channel Setup PRI N00
- 0 1 1 0 0 0 1 1 : Channel Setup PRI Intl
- 0 1 1 0 0 1 0 0 : Specific Digit String Info*
- 0 1 1 0 0 1 0 1 : Specific Digit String ANI*
- 0 1 1 0 0 1 1 0 : Specific Digit String N00*
- 0 1 1 0 0 1 1 1 : Specific Digit String CIC

- 0 1 1 0 1 0 0 0 : Shared Interoffice CIC
- 0 1 1 0 1 0 0 1 : Shared Interoffice Info
- 0 1 1 0 1 0 1 0 : Shared Interoffice ANI
- 0 1 1 0 1 0 1 1 : Shared Interoffice Addr
- 0 1 1 0 1 1 0 0 : Shared Interoffice N00
- 0 1 1 0 1 1 0 1 : Shared Interoffice Intl
- 0 1 1 0 1 1 1 0 to 1 1 1 1 1 1 1 1 : Unused (Spare)

Serving Translation Scheme

This parameter contains the serving translation scheme (STS) of the call. The switch sends this parameter to the SCU in the New Call event notification message.



Optional Parameter Tag: **131**

Parameter Length: **2 Bytes**

Parameter Contents:

- **COUNT** : 4 Bits: This contains the number of digits in the STS. Valid values include 1 to 3.
- **DIGITS 1, 2, and 3** : Each digit is 4 Bits: This field contains the 3 digits in the TBCD format, which are encoded as follows:
 - 0 0 0 0 : Filler
 - 0 0 0 1 : Digit 1
 - 0 0 1 0 : Digit 2
 - 0 0 1 1 : Digit 3
 - 0 1 0 0 : Digit 4
 - 0 1 0 1 : Digit 5
 - 0 1 1 0 : Digit 6
 - 0 1 1 1 : Digit 7
 - 1 0 0 0 : Digit 8
 - 1 0 0 1 : Digit 9
 - 1 0 1 0 : Digit 0
 - 1 0 1 1 : *
 - 1 1 0 0 : #

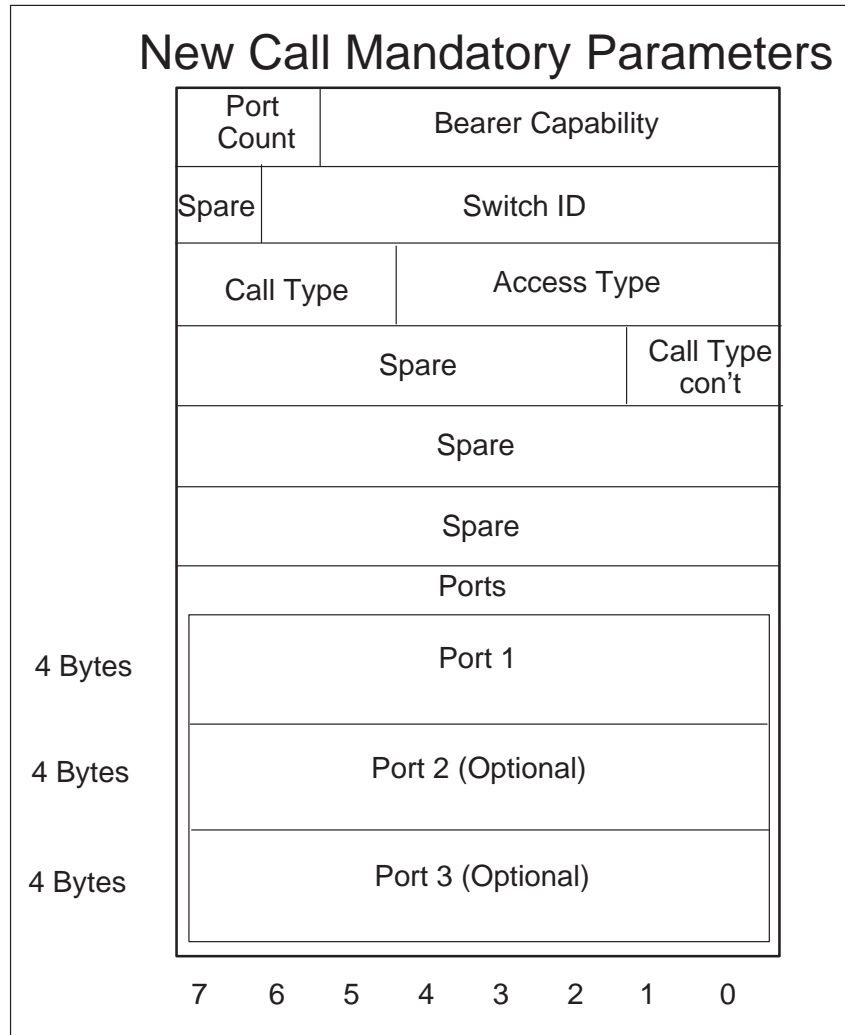
- 1 1 0 1 : D
- 1 1 1 0 : E
- 1 1 1 1 : F

PSN Messaging

PSN messages in the LOPER format are described in Chapters 14, 17, 19, and 21 in this document. Most of the messages sent to the SCU or received from the SCU do not require modifications for this feature. The following sections describe the messages that have different formats from the ones defined as the strategic PSN messages.

New Call Mandatory Parameters

The mandatory parameters for a New Call Event.



Parameter Length: **MIN: 10 Bytes MAX: 18 Bytes**

Parameter Contents:

- *BEARER CAPABILITY*: 6 Bits. *TYPE*: *Bearer Capability Parameter*
- *PORT COUNT*: 2 Bits. *VALUES INCLUDE*: 0 to 3.
- *SWITCH ID*: 7 Bits. *TYPE*: *Switch ID Parameter*
- *ACCESS TYPE*: 5 Bits. *TYPE*: *Access Type Parameter*
- *CALL TYPE*: 5 Bits. *TYPE*: *Call Types Parameter*
- *PORTS*: Ports consists of a minimum of 1 port to a maximum of 3 ports.

Port TYPE: *Port Info Parameter*

New Call Optional Parameters

The following is a list of optional parameters for a New Call Event.

Collected Digits

This parameter contains any digits collected during regular call processing. A maximum of 10 optional Collected Digits parameters will be decoded. After the 10th optional Collected Digits parameter has been decoded if additional optional Collected Digits parameters are dropped.

TYPE: *Digits Collected Parameter*

Serving Translation Scheme (STS)

This parameter contains the serving translation scheme of the call. The switch sends this parameter to the SCU in the New Call event notification message.

TYPE: *STS Parameter*

Point In Call (PIC)

This parameter contains the point in the call when all criteria are met and the PSN gives the SCU control over the call.

TYPE: *Point In Call Parameter*

Flow Control Encountered

This parameter informs the SCU that Flow Control is active and it provides the source that initiated the Flow Control is active.

TYPE: *Flow Control Encountered Parameter*

Signaling Information

This parameter contains any signaling information that is stored during regular call processing.

TYPE: Signaling Info Parameter

This parameter contains the Signaling Information in the standard format that is applicable to the signaling type of the port. This parameter may be used to receive or send signaling information from/to the SCU.

Depending upon the primitive or event notification in which this parameter is transmitted and the signaling type of the associated port, its contents are different. In general, the contents contain parameters that are encoded in the standard format.

For PTS agents, there are no standards used. For SS7 agents, the parameters are encoded based on the TR444 and GR394, and for PRI agents, the TR1268 and ITU-T Q.931 are used to define the parameters.

Signaling Info is included in the primitives and event notification messages as shown in Table 22-1. For details on all the parameters that are supported within the signaling info parameter, refer to the following section in this appendix, “SS7 and PRI Messages and Parameters Supported in SIGINFO Parameter.”

Table 22-1
PTS, SS7, and PRI Messages that are included in the Primitives and Event Notifications

Primitive/Event Notification	PTS Messages	SS7 Messages	PRI Messages
Connect	Digits To Outpulse, Bearer Capability	IAM	SETUP
Disconnect	—	REL	DISC, REL
Transmit Siginfo	Digits To Outpulse, PTS On-Hook, PTS Off-Hook	IAM, ANM, REL, CPG, FAA, FAR, FRJ, PAM, RES, SUS	CONNECT, REL, DISC, FACility, Alert
Off-Hook	—	ANM	CONNECT
—continued—			

Table 22-1
PTS, SS7, and PRI Messages that are included in the Primitives and Event Notifications (continued)

Primitive/Event Notification	PTS Messages	SS7 Messages	PRI Messages
On-Hook	—	REL, RSC	DISC, REL
Signaling Event	Digits Outpulsed Indicator	ACM, CPG, COT, FAA, FAR, FRJ, PAM, RES, SUS	PROGRESS, ALERT, FACILITY, CALL PROCEEDing, RELease COMPlete
—end—			

PSN Agencies

The originating agency is the agency which first enters server mode by triggering through CAIN. PSN supports the following originating agencies:

- DAL (4 wire and 2 wire FXS)
- IMT – PTS and SS7
- FGD – PTS and SS7
- PRI

The subsequent agency is the agency that is brought into server mode by the SCU. PSN supports the following subsequent agencies:

- DAL (4 wire and 2 wire FXS)
- FGB
- FGC
- FGD (PTS and SS7)
- IMT (PTS and SS7)
- PRI

Feature interactions

In this phase of PSN, there are no feature interactions with the existing In-switch features. Once an agent is tagged a PSN agent, the SCU is given full control of this PSN agent. The SCU may then send instructions to control each PSN agent.

The SCU maintains full control of this call until all parties in this call are on-hook or are disconnected by the SCU.

Terminating Agent Bearer Capability (BC) Screening

When a connection is made between any two ports (using either a Connect or a Reconnect primitive), the bearer capability of the two ports is screened for compatibility. (Bearer Capability screening is not supported for multi-party calls that are setup using the Bridge primitive.) Table BCCOMPAT defines the bearer capability pairs which are compatible with one another.

If the bearer capability screening fails, then the connection is not made and an Error Detected event notification message is sent to the SCU.

BC Derivation for Originating Agent

For the port that triggers a New Call event to the SCU, the Bearer Capability is obtained in-switch.

- This value is obtained from the BCNAME field in table TRKGRP.
- For subsequent connections on this port, the SCU may change the port's Bearer Capability (using the optional Originator Bearer Capability parameter in the Connect primitives). If this happens, the new BC is used to check the compatibility.

Bearer Capability derivation for subsequent agent

For all subsequent ports that are added to a PSN call, the Bearer Capability is derived from the Signaling Info parameter which is included in the Connect primitive. Subsequent connection setups on this port use this bearer capability information for BC screening.

- 1 The Bearer Capability information is included in the Signaling Info parameter as the following sub-parameters and is based on the signaling type of the port.
 - The sub-parameter BEARER CAPABILITY for PTS agents.
 - The sub-parameter USER SERVICES INFORMATION for SS7 agents.The sub-parameter BEARER CAPABILITY (in Q.931 format) for PRI agents.
- 2 If the Bearer Capability sub-parameter is not included in the Signaling Info parameter, then the BC is derived from table TRKGRP.
 - For IMT trunks, the Bearer Capability is obtained from the DDI field of table TRKGRP.

- For every other trunk, the Bearer Capability is obtained from the BCNAME field of table TRKGRP.

PSN Optionality

The following sections contain information on PSN optionality control.

PSN Optionality Control Requirements

Software Optionality Control (SOC) functionality allows operating company personnel to select optional groups of one or more features, called options. Customers control these options through the SOC CI level. Nortel Networks supports personnel control passwords for activation and deactivation of the right-to-use (RTU) on all options.

This activity implements a SOC Combination Option, referred to as order number (UPSN0001), which contains a combination of SOC State option functionality and SOC Usage option functionality. Please see *UCS DMS-250 Software Optionality Control (SOC) User's Manual* for more information.

- State option – A state option controls its features by assigning a state of either IDLE or ON to the option through a SOC CI command after the purchase of the proper passwords from Nortel Networks. Software allows use of the feature on the basis of the option's state.
- Usage option – A usage option controls its feature by assigning a limit to the number of resources that the option controls. The three types of limits that can be assigned are:
 - Hard limit – If a hard limit is assigned, no resources may be assigned beyond the limit.
 - Soft limit – If a soft limit is assigned, resources may be allocated beyond the limit.
 - MONITORED limit – If MONITORED is assigned, there is no limit to the number of resources that may be allocated.

Logs are generated when a limit is reached or exceeded. There is also a THRESHOLD value that may be set by operating company personnel so that logs are generated before a limit is reached. Please see Chapter 8, "PSN Logs" for more information. The usage options in this activity will be sent to operating companies with a default hard limit of 0.

The PSN SOC counts the number of active PSN agents. Therefore, if the limit is set to 10,000, this allows for up to 10,000 concurrently active PSN agents. Please note that the count refers to the number of active agents, not calls. For example, if the typical PSN call has an average of 2 active agents, then a customer has purchased approximately 5,000 currently active PSN calls when purchasing a PSN SOC count of 10,000.

Because of the real-time impact of SOC, PSN SOC usage resources are allocated as blocks of 1000 weighted Call Processing Class primitives. This means that the first 1000 messages sent do not notify SOC of usage. The feature tracks the usage internally to 1000 before notifying SOC. This allows the SOC real-time to affect only 1 in every 1000 weighted Call Processing Class primitives which are successfully processed.

SOC resource usage can be verified through the Operational Measurement (OM) UPSNACT group. This OM group contains fields corresponding to Call Processing Class primitives and events which create or destroy PSN agents (that is, new call and disconnect). Each OM group field is incremented when the corresponding primitive or event is successfully processed. For more information on the UPSNPRIM OM Group, see the *UCS DMS-250 Operational Measurements Reference Manual*.

When the PSN SOC option UPSN0001 state is not ON or the PSN SOC option UPSN0001 hard limit is exceeded, a SOC resource is not granted to process a received Call Processing Class primitive or event that is going to create a PSN agent (new call event or connect primitive). When this occurs, a UPSN 100 log is generated, and either the UPSNFAIL OM Group Register PRIMFAIL (for a failed **Connect** primitive) or EVENFAIL (for a failed **New_Call** event) is incremented. For more information on the UPSNFAIL OM Group, see the *UCS DMS-250 Operational Measurements Reference Manual*. For more information on the UPSN log, see the *UCS DMS-250 Logs Reference Manual*.

When the PSN SOC option UPSN0001 state is IDLE, or when the PSN SOC hard limit is exceeded, a new call's attempt to query an SCU for supervision results in a PSN Failure (PSNF) Treatment on the originating agent.

When the PSN SOC option UPSN0001 state is IDLE, or when the PSN SOC hard limit is exceeded, an attempt to route a CAIN SCP supervised agent to an SCU for supervision results in a PSN Failure (PSNF) Treatment on the originating agent.

When the PSN SOC option UPSN0001 state is IDLE, or when the PSN SOC hard limit is exceeded, PSN does not attempt to initiate polling for connection to an SCU.

Since PSN functionality is dependent on either CAIN CUSTDP Trigger functionality or CAIN SPECDDIC Trigger functionality, the PSN SOC option displays a dependency warning message when the PSN SOC state is transitioned to ON when both the CAIN CUSTDP Trigger SOC option (CAIN0500) and CAIN SPECDDIG Trigger SOC option (CAIN0501) are in the IDLE state.

Also, when transitioning the SOC options CAIN0500 or CAIN0501 from ON to IDLE, SOC displays a dependency warning message indicating that the UPSN0001 option is dependent on the CAIN Trigger options and PSN functionality could be affected.

PSN Optionality Control Feature Interaction

This PSN Optionality Control interacts with the following features:

- Software Optionality Control Base feature.
- Carrier-AIN SOC.

See the *UCS DMS-250 Software Optionality Control (SOC) User's Manual*.

PSN Optionality Control interaction with the SOC BASE feature causes minor changes to be visible to SOC. Refer to the *UCS DMS-250 Commands Reference Manual* and the *UCS DMS-250 Logs Reference Manual* for more information.

PSN dependency on the CAIN CUSTDP Trigger feature and the CAIN SPECDIG Trigger feature requires the state transition impact statements for SOC options CAIN0500 and CAIN0501 to provide warning statements indicating that PSN functionality could be affected when transitioning CAIN0500 or CAIN0501 from ON to IDLE. The PSN SOC option also displays a dependency warning message when transitioning from IDLE to ON when both the CAIN CUSTDP Trigger option (CAIN0500) and CAIN SPECDIG Trigger option (CAIN0501) are in the IDLE state.

PSN Billing

The SCU generates billing records for all agents under its control since it has the information about the services it provides. The SCU has the primary responsibility for the billing records.

The UCS DMS-250 switch also generates billing records for agents associated with a PSN call. This is used to supplement the billing records generated by the SCU.

For this release of the PSN, Call Detail Records (CDR) are provided for each agent (trunk) entering server mode operation as follows:

Default CDR

For the initial agent which enters server mode as the result of CAIN triggering, a CDR will be generated with the following fields populated:

ORIGGRP: Originating Trunk Group
ORIGMEM: Originating Trunk Member

ORIGDATE: Origination Date
 ORIGTIME: Origination Time
 DISCDATE: Disconnect Date
 DISCTIME: Disconnect Time
 CRID: Call Reference ID (optional)
 SEQNUM: Sequence Number

Also for the agent originating and entering server mode, CDR fields which had been populated up to the point of server mode entry will retain their values.

For subsequent agents which enter server mode as the result of SCU control, a CDR is generated with the following fields populated:

TERMGRP: Terminating Trunk Group
 TERMMEM: Terminating Trunk Member
 ORIGDATE: Origination Date
 ORIGTIME: Origination Time
 DISCDATE: Disconnect Date
 DISCTIME: Disconnect Time
 CRID: Call Reference ID (optional)
 SEQNUM: Sequence Number

The CDR generated for an initial agent always has that agent's Trunk Group and Trunk Member numbers in the ORIGGRP and ORIGMEM fields. For subsequent agents, the Trunk Group and Trunk Member numbers are in TERMGRP and TERMMEM fields. The ORIGTIME and ORIGDATE will be populated based on the time and date the agent was added to the call, and the DISCDATE and DISCTIME will show the date and time the agent was removed from the SCU call. The SEQNUM ensures that records are not lost in transit to billing systems.

0 00000000: Unknown
 1 001: Asterisk Detected
 110 - 111: Unused (Spare)

Populated CDR

The SCU can overwrite the default value of a CDR field by sending the *BillinInfo* parameter, either as a mandatory or as an optional parameter. There are 17 CDR fields which might be populated by the SCU. Every digit type has a maximum number of digits. The maximum size of the *BillinInfo* parameter is 240 bytes. The following list specifies the 17 digit types and their maximum number of digits supported by the *BillinInfo* parameter:

- Acctcode Digits: a Max of 12 Digits

- Anisp Digits: a Max of 10 Digits
- Billnum Digits: a Max of 23 Digits
- Call Duration Digits: a Max of 5 Digits (Note)
- Carrierid Digits: a Max of 4 Digits
- Calling Ptyaddr Digits: a Max of 15 Digits
- CRID Digits: a Max of 9 Digits
- Dialedno Digits: a Max of 18 Digits
- DNIS Digits: a Max of 15 Digits
- Infodig Digits: a Max of 2 Digits
- Origgrp Digits: a Max of 4 Digits
- Origmem Digits: a Max of 4 Digits
- PIN Digits: a Max of 4 Digits
- Termgrp Digits: a Max of 4 Digits
- Termmem Digits: a Max of 4 Digits
- Termpvn Digits: a Max of 15 Digits
- Univacc Digits: a Max of 10 Digits

Note: The maximum value the SCU can send to populate the Call Duration CDR field is 65499 Seconds (equal to 18.19 Hours), which gets displayed in the CDR fields in 10 ms increments. For example, the value 123 becomes 1. The value is rounded up if the last 2 digits are 50 or above. The value is rounded down if the last 2 digits are 49 or less. For example, 9550 is displayed as 00000096 and 9549 is displayed as 00000095.

One CDR is generated for every agent involved in a server mode call when the trunk is disconnected, goes on-hook or is taken down by the UCS DMS-250 switch due to a fatal error.

PSN Billing Details

The details on how and what billing information is captured depend entirely upon the customer using PSN. The customer may decide to capture some information using the CDRs (Call Detail Records), or may decide to do the billing entirely on the SCU.

Table 22-2 shows the mapping of CDR fields that any primitive that has the *BillinInfo* parameter, either as mandatory or as an optional parameter may populate, and it shows the Digit Type which correlates to each of the CDR fields.

Table 22-2
CDR field name mapping

CDR field name	Digit Type
ACCTCD	Acctcode digits (23)
ANISP	Anisp digits (10)
BILLNUM	Billing num digits (5)
CIC	Carrierid digits (8)
CLGPTYNO	Calling ptyaddr digits (2)
CRID	CRID digits (16)
DIALEDNO	Dialedno digits (26)
DNIS	DNIS digits (21)
INFODIG	Infodig digits (9)
PINDIG	Pin digits (14)
TERMPVN	Termpv digits (18)
UNIVACC	Univacc digits (19)
CALLDUR	Call duration digits (7)
ORIGGRP	Origgrp digits (11)
ORIGMEM	Origmem digits (12)
TERMGRP	Termgrp digits (24)
TERMMEM	Termmem digits (25)

Additional information

The following sections contain additional information on PSN for the UCS DMS-250 switch.

Engineering/Hardware

The existing six-port conference bridge is used to allow more than two ports to be bridged. It is also used to bridge two or more parties to a message.

PSN requires the use of six-port and not three-port conference circuits.

Logs

PSN does not generate any CAIN logs.

The format of logs CDR272 and CDR273 are modified to match the new Call Detail Record (CDR) format. See the *UCS DMS-250 Logs Reference Manual*.

Data schema

CAIN Trigger Database Tables

Changes are made to the CAIN trigger database tables to add the new action QUERYSCU. Refer to the *UCS DMS-250 Data Schema Reference Manual* for additional information.

CAIN Simulator Tables

Changes are made to the CAIN simulator tables to add the new extension parameter, ConnectToScu, to the TCAP Continue message.

Commands

A change is made to the CAINTEST CI command to display the ConnectToScu extension parameter in the Continue message when it is received as a response to a CAIN query or conversation message sent from CAINTEST. The following is an example of the Continue message with the ConnectToScu extension parameter.

>SEND QUERY 1

```
Message sent, waiting for reply
TIME --> RECEIVED RESPONSE IN 3.7 SECONDS
CONTINUE received
Continue call with the SCP
CONNECT_TO_SCU: Y
Mailbox deallocated
```

Several SOC CI commands have been changed by providing SOC for the Programmable Service Node. See the *UCS DMS-250 SOC User's Manual*.

A warning message is displayed if a user attempts to transition UPSN0001 to ON when CAIN0500 and CAIN0501 are in the IDLE state.

A warning message is displayed if a user attempts to transition CAIN0500 to IDLE when UPSN0001 is in the ON state.

A warning message is displayed if a user attempts to transition CAIN0501 to IDLE when UPSN0001 is in the ON state.

See the *UCS DMS-250 Commands Reference Manual* for additional information.

Operational measurements (OM)

A register is added to the CAINTRIG OM group for the new trigger action, QUERYSCU.

The OM group UPSNACT is added for SOC resource usage verification.

A new OM group UPSNFAIL is added for the case when SOC denies processing of a PSN Call Processing Class primitive.

See the *UCS DMS-250 Operational Measurements Reference Manual*. for additional information.

Restrictions/Limitations

The following restrictions/limitations apply to PSN calls:

- Only two-way trunks are supported as either originators or terminators of PSN calls. Outgoing only or incoming only agents are considered invalid for PSN calls and termination or originations to these trunks are not allowed in server mode.
- EDALs and ONALs are not supported as either originators or terminators of PSN calls.
- Trunk agencies that rely on ATD receivers are not supported.
- A UCS call can only be either a CAIN call or a PSN call, but not both at any point in time. A CAIN call can become a PSN call, but CAIN processing is terminated when this occurs.
- AXXESS Agents are not supported as either originators or terminators of PSN calls.
- Two wire trunks (for example, DAL) may “leak” some of the received voice prompts signal into the call processing as user input. When the Bong Tone is recorded starting with a tone that represents # sign, under severe “leak,” the call processing may interpret this as the termination of user digit stream input.
- Two-wire FXO trunks are not supported as either originators or terminators of PSN calls.

SS7 and PRI Messages and Parameters Supported in SIGINFO Parameter

The following diagram provides a list of SS7 messages and parameters that are passed between the PSN and the SCU inside the Signaling INFO parameter.

SS7 Messages

SS7 Messages
Address Complete Message (ACM)
Message Type (M)
Backward Call Indicators (M)
Call Reference
US Network Parm (#FE)
User to User Indicator (#2A)
User to User Information (#20)
Answer Message (ANM)
Message Type (M)
Backward Call Indicators
Call Reference
Carrier Selection
Internetwork Specific ANM
Intranetwork Specific ANM
Network Specific ANM
Operator Information
US Network Parm Name (#FE)
User to User Indicator (#2A)

SS7 Messages
User to User Information (#20)
Call Progress Message (CPG)
Message Type (M)
Event Information (M)
Backward Call Indicators
Party Information
Redirection Number
Supply End-to-End Information Request
Supply End-to-End Information Response
User to User Indicator (#2A)
User to User Information (#20)
Continuity Message
Message Type (M)
Continuity Indicators (M)
Facility Accepted Message (FAA)
Message Type (M)
Facility Indicator (M)
Call Reference
Facility Request Message (FAR)
Message Type (M)
Facility Indicator (M)
Call Reference
Called Party Address

SS7 Messages
Calling Party Number
Charge Adjustment
Charge Number
GD Callid
Operator Information
Originating Line Info
Facility Reject Message (FRJ)
Message Type (M)
Facility Indicator (M)
Cause Indicator (M)
Call Reference
Initial Address Message (IAM)
Message Type (M)
Called Party Number (M)
Calling Party's Category (M)
Forward Call Indicators(M)
Nature of Connection Indicators(M)
User Service Information (M)
Calling Party Address
Carrier Identification
Carrier Selection Information (#EE)
Channel Assignment Map (#25)
Charge Number

SS7 Messages
GD (Authcode)
GD (CallID)
GD (CLLI Admin)
GD (IMT Info)
GD (RLT Treatment Code)
GD (Terminating SWID & TRKGRP)
GD (XFR Operator Queue)
Network Information (#C3)
Network Specific Facilities (NSF)
Network Specific IAM
Operator Information
Operator Service Indicators
Originating Line Information
Supplementary Line Information
Transit Network Selection (TNS)
Pass Along Message (PAM)
Message Type (M)
RPM message [with Reconfig. Ind. (M) & Backward Call Ind.]
Release Message (REL)
Message Type (M)
Cause Indicators (M)
User to User Indicator (#2A)
User to User Information (#20)

SS7 Messages
Reset Circuit Message (RSC)
Message Type (M)
Resume Message (RES)
Message Type (M)
Resume Indicators (M)
Suspend Message (SUS)
Message Type (M)
Suspend Indicators (M)

Q.931 PRI Messages

The following diagram provides a list of the Q.931 PRI Messages:

Q.931 PRI Messages
ALERTing
Message Type (M)
Protocol Discriminator (M)
Progress Indicator
User to User Information
CALL PROCeeding
Message Type (M)
Protocol Discriminator (M)
CONNect
Message Type (M)

Q.931 PRI Messages
Protocol Discriminator (M)
User to User Information
DISConnect
Message Type (M)
Cause (M)
Protocol Discriminator (M)
Cause
User to User Information
FACility
Message Type (M)
Protocol Discriminator (M)
Called Party Number
Called Party Subaddress
PROGress
Message Type (M)
Protocol Discriminator (M)
Progress Indicator (M)
Cause
User to User Information
RELease
Message Type (M)
Protocol Discriminator(M)
Cause

Q.931 PRI Messages
User to User Information
RELease COMPlete
Message Type (M)
User to User Information
SETUP
Message Type (M)
Bearer Capability (M)
Called Party Number (M)
Protocol Discriminator (M)
Business Group
Called Party Subaddress
Calling Party Number
Calling Party Subaddress
Display
Higher Layer Compatibility
Lower Layer Compatibility
Network Specific Facility
Original Called Number
Progress Indicator
Transit Network Selection
User to User Information

New call information collected

The following sections describe new call information collected for PSN.

- Agencies: The following are valid originating agencies for PSN on UCS06. They are classified based on the signaling type. All agencies must be 2-way trunks. New Call is supported on all the agencies provided in the following list:
 - PTS: FGD, IMT, DAL, TIE (MF and DTMF)
 - SS7: FGD, IMT
 - PRI: basic PRI.
- New Call Triggers: For each New Call sent to the SCU, the Trigger Criteria at which the port either (a) sends a message to the SCU and enters the server mode or (b) sends a message to the AIN SCP and later on (based on the SCP response) enters the server mode— is specified.

Table 22-3 provides the possible Trigger Criteria for the PSN agencies provided in the preceding list. For each applicable Trigger Criteria, if an action of QUERYSCU is encountered, then the call is deemed a PSN call and a **New_Call** event notification is sent to the SCU with the information listed in this section.

Table 22-3
Trigger Criteria Applicable for PSN Agencies

PIC	TDP	Trigger	Agency	Trigger Criteria for the Agency
O_Null	Origination_Attempt	Offhook_Immediate	DAL	Refer to Table 22-7
Collect_information	Information_Collected	Shared_Interoffice_Trunk	PTS, FGD	Refer to Table 22-4
			SS7, FGD	Refer to Table 22-4
			PTS, IMT	Refer to Table 22-11
			SS7, IMT	Refer to Table 22-11
			DAL	Refer to Table 22-8
	TIE	Refer to Table 22-8		
		PRI_B_Channel	PRI	Refer to Table 22-14
—continued—				

Table 22-3
Trigger Criteria Applicable for PSN Agencies (continued)

PIC	TDP	Trigger	Agency	Trigger Criteria for the Agency
Analyze_Information	Information_Analyzed	Custom_Dialing_Plan	PTS, FGD	Refer to Table 22-5
			SS7, FGD	Refer to Table 22-5
			PTS, IMT	Refer to Table 22-12
			SS7, IMT	Refer Table 22-12
			DAL	Refer Table 22-9
			TIE	Refer to Table 22-9
			PRI	Refer to Table 22-15
—continued—				

Table 22-3
Trigger Criteria Applicable for PSN Agencies (continued)

PIC	TDP	Trigger	Agency	Trigger Criteria for the Agency
Analyze_Information	Information_Analyzed	Specific_Digit_String	PTS, FGD	Refer to Table 22-6
			SS7, FGD	Refer to Table 22-6
			PTS, IMT	Refer to Table 22-13
			SS7, IMT	Refer to Table 22-13
			DAL	Refer to Table 22-10
			TIE	Refer to Table 22-10
			PRI	Refer to Table 22-16
—end—				

Parameters in the NEW CALL Event Notification

The **New_Call** event notification is sent to the SCU when the PSN determines that the port is to be controlled by the SCU, (when the port enters the server mode). There are several ways that the PSN may determine that the port is to be controlled by the SCU:

- it is determined through the existing CAIN logic (using the existing CAIN triggers) that the port is to be controlled by the SCU (actions QUERYSCU in the CAIN trigger tables SPECDIG and CUSTDP).
- the call queries the CAIN SCP, a response is returned from the CAIN SCP indicating that the call is to be controlled by the SCU.

For each of the cases in the preceding text, the parameters sent in the **New_Call** event notification may be different.

Also, the New Call parameters depend upon the access type of the port, the call type at the time of sending the new call to the SCU, the point in call and the digits information that is available at the time of query.

The following is a detailed list of New Call parameters.

New Call – BEFORE Querying the AIN SCP

This section describes the New Call parameters for ports when the AIN trigger tables indicate that the call is to be controlled by the SCU, and the call has not at any point before this queried the AIN SCP.

The following New Call parameters are applicable to the FGD agent. The Dialling Plan (DP) for FGDs may be one of the following.

The SS7 FGD information is converted to one of the formats shown in the following dial plans before the TDPs are encountered. The specific format that they are converted into depends upon information that is received in the IAM message.

- 1 Pure FGD (with DTMF PIN and Account)
 - a. MF: KP + II + ANI + ST
 - b. MF: KP + Address + ST
 - c. DTMF: PIN + Account
- 2 FGD International
 - a. MF: KP + 1NX + XXX + CCC + ST
 - b. MF: KP + II + ANI + ST
 - c. MF: KP + Address (CC + NN) + ST
 - d. DTMF: (PIN) + (ACCT)
- 3 FGD UA International
 - a. MF: KP +(10-digit ANI) + ST
 - b. MF: KP + 800-NXX-XXXX+ST
 - c. DTMF: AUTH + (PIN) + 011 + CC + NSN + (AC)
- 4 FGD (Cut-thru)
 - a. MF: KP + II + ANI + STP
 - b. DTMF: Address
- 5 FGD Transitional (auth billed)
 - a. MF: KP + II + ANI + STP2/STP3
 - b. DTMF: Authcode + Address

- 6 FGD UA (auth billed)
 - a. MF: KP + II + ANI + ST
 - b. MF: KP + UA (Universal Access Number) + ST
 - c. DTMF: Authcode + Address
- 7 3-Stage Domestic
 - a. MF: KP + OZZ + XXX(X) + ST
 - b. MF: KP + II + ANI + ST
 - c. MF: KP + Address + ST

Table 22-4
Collection_Information, Information_Collected, Shared_Interoffice_Trunk for PTS FGD and SS7 FGD

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-4
Collection_Information, Information_Collected, Shared_Interoffice_Trunk for PTS FGD and SS7 FGD (continued)

Digits Type	New Call Parameters
ADIN	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled key STS)</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-4
Collection_Information, Information_Collected, Shared_Interoffice_Trunk for PTS FGD and
SS7 FGD (continued)

Digits Type	New Call Parameters
ANI	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled key STS)</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-4
Collection_Information, Information_Collected, Shared_Interoffice_Trunk for PTS FGD and SS7 FGD (continued)

Digits Type	New Call Parameters
CIC	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), fcarrierCode for DP g)</p> <p>STS (STS from ANI for DP g)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
INFO	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—end—	

Table 22-5
Analyze_Information, Information_Analyzed, Custom_Dialing_Plan for PTS FGD and SS7 FGD

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-5
Analyze_Information, Information_Analyzed, Custom_Dialing_Plan for PTS FGD and SS7
FGD (continued)

Digits Type	New Call Parameters
XLAADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—end—	

Table 22-6
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS FGD and SS7 FGD

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-6
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS FGD and SS7 FGD
 (continued)

Digits Type	New Call Parameters
ADIN	switchId, portsInTheCall, bearerCapability, signalingType, agentType accessType (pts_fgd, ss7_fgd) callType (from address pretranslations for all the DPs) infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a) infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d) infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c) infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f) STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f) Partition (OPART and TPART (datafilled key STS) Signaling Info (IAM - if SS7)
—continued—	

Table 22-6
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS FGD and SS7 FGD
 (continued)

Digits Type	New Call Parameters
ANI	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
CIC	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), fcarrierCode for DP g)</p> <p>STS (STS from ANI for DP g)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-6
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS FGD and SS7 FGD
 (continued)

Digits Type	New Call Parameters
INFO	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits +ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled key STS)</p> <p>Signaling Info (IAM - if SS7)</p>
—continued—	

Table 22-6
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS FGD and SS7 FGD
 (continued)

Digits Type	New Call Parameters
XLAADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_fgd, ss7_fgd)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), personalIdentificationNumber, accountCodeNumber for DP a)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), for DP b and d)</p> <p>infoCollected (callingPartyAddress (ANI), calledPartyAddress (address), authorizationCode for DP c)</p> <p>infoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), authorizationCode for DP e and f)</p> <p>STS (STS from ANI for DP a, b, and d; STS from authcode for DP c, e, and f)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>Signaling Info (IAM - if SS7)</p>
—end—	

The following New Call parameters are applicable to the DAL and TIE agent (both MF and DTMF signaling types). The Dial Plan for DALs and TIEs may be one of the following. Dial plans (5) and (7) are applicable to TIEs only.

- 1 DTMF: Authcode + Pin + Address + Account (dialed auth first)
- 2 DTMF: Address + Authcode + Pin + Account (dialed auth last)
- 3 DTMF: Pin + Address + Account (Filed auth)
- 4 DTMF: Authcode + Pin + Address (PANI calls – ANIPIN DP)
- 5 MF: KP + Authcode + Pin + Address + Account + ST (auth billed)
- 6 International (DTMF only)
 - DTMF: 011 + (Authcode) + (Pin) + CC + NSN + (Account)
- 7 International (MF only)
 - MF: KP + (AUTH) + (PIN) + 011 + CC + NSN + (Account) + ST

Table 22-7
O_Null, Origination_Attempt, Offhook_Immediate for DAL

Digits Type	New Call Parameters
ADDR	switchId, portsInTheCall, bearerCapability, signalingType, agentType accessType (2w_dal) callType (from address pretranslations) InfoCollected STS (STS from datafilled auth field) Partition (OPART and TPART (datafilled by key STS))
<p>Note: The N/A PIC is triggered by datafill in the trigger table only. Since there is no digit dial for triggering this PIC, there is no Digits Collected either.</p>	

Table 22-8
Collect_Information, Information_Collected, Shared_Interoffice_Trunk for DAL and TIE (DTMF and MF)

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
ADIN	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
<p>Note: There is no CIC-specific behavior for call on DAL or TIE originations.</p>	
<p align="center">—continued—</p>	

Table 22-8
Collect_Information, Information_Collected, Shared_Interoffice_Trunk for DAL and TIE (DTMF and MF) (continued)

Digits Type	New Call Parameters
ANI	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
INFO	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
<p>Note: There is no CIC-specific behavior for call on DAL or TIE originations.</p>	
<p>—end—</p>	

Table 22-9
Analyze_Information, Information_Analyzed, Custom_Dialing_Plan for DAL and TIE (DTMF and MF)

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
XLADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
<p>Note: There is no CIC specific behavior for call on DAL or TIE originations.</p>	
<p align="center">—continued—</p>	

Table 22-10
Analyze_Information, Information_Analyzed, Specific_Digit_String for DAL and TIE (DTMF and MF)

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
ADIN	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
<p>Note: There is no CIC specific behavior for call on DAL or TIE originations.</p>	
<p align="center">—continued—</p>	

Table 22-10
Analyze_Information, Information_Analyzed, Specific_Digit_String for DAL and TIE (DTMF and MF) (continued)

Digits Type	New Call Parameters
ANI	switchId, portsInTheCall, bearerCapability, signalingType, agentType accessType (4w_dal, 2w_dal) callType (from address pretranslations for all the DPs) infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d) STS (STS from ANI (PANI) for DP d) Partition (OPART and TPART (datafilled by key STS))
Note: There is no CIC specific behavior for call on DAL or TIE originations.	
—continued—	

Table 22-10
Analyze_Information, Information_Analyzed, Specific_Digit_String for DAL and TIE (DTMF and MF) (continued)

Digits Type	New Call Parameters
INFO	switchId, portsInTheCall, bearerCapability, signalingType, agentType accessType (4w_dal, 2w_dal) callType (from address pretranslations for all the DPs) InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g) infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d) STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d) Partition (OPART and TPART (datafilled by key STS))
XLAADDR	switchId, portsInTheCall, bearerCapability, signalingType, agentType accessType (4w_dal, 2w_dal) callType (from address pretranslations for all the DPs) InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g) infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d) STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d) Partition (OPART and TPART (datafilled by key STS))
Note: There is no CIC specific behavior for call on DAL or TIE originations.	
—end—	

The following New Call parameters are applicable to the IMT agent. The Dialing Plan (DP) for FGDs may be one of the following:

Note: The SS7 IMT information is converted to one of the formats shown in the following dial plans before the TDPs are encountered. The specific format that they are converted into depends upon information that is received in the IAM message.

- 1 ADDR
 - MF: KP + Addr + ST
 - or
 - DTMF: Addr + (#)
- 2 I3PA
 - MF: KP + FC + PART + Addr + ST
 - or
 - DTMF: FC + PART + Addr + (#)
- 3 SHARED
 - MF: KP + FC + COS + PART + Addr + ST
 - or
 - DTMF: FC + COS + PART + Addr + (#)
- 4 UNIVERSAL ACCESS
 - MF: KP + 800-NXX-XXXX + ST
 - DTMF: Auth + Pin + Addr + Acct
 - or
 - DTMF: 0 + Addr + TCN + Acct
- 5 3-Stage Domestic
 - MF: KP + OZZ + XXX(X) + ST
 - MF: KP + II + ANI + ST
 - MF: KP + Address + ST

Note: Account codes are only supported for TCNs validated out-of-switch.

Table 22-11
Collect_Information, Information_Collected, Shared_Interoffice_Trunk for PTS IMT and SS7
IMT

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
ADIN	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
—continued—	

Table 22-11
Collect_Information, Information_Collected, Shared_Interoffice_Trunk for PTS IMT and SS7
IMT (continued)

Digits Type	New Call Parameters
ANI	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
CIC	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), carrierCode for DP e)</p> <p>STS (STS from ANI for DP e)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
—continued—	

Table 22-11
Collect_Information, Information_Collected, Shared_Interoffice_Trunk for PTS IMT and SS7
IMT (continued)

Digits Type	New Call Parameters
INFO	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
XLAADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
—end—	

Table 22-12
Analyze_Information, Information_Analyzed, Custom_Dialing_Plan for IMT

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
XLAADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
—continued—	

Table 22-13
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS IMT

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
ADIN	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
—continued—	

Table 22-13
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS IMT (continued)

Digits Type	New Call Parameters
ANI	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
CIC	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress (Info Digits + ANI), calledPartyAddress (address), carrierCode for DP e)</p> <p>STS (STS from ANI for DP e)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
—continued—	

Table 22-13
Analyze_Information, Information_Analyzed, Specific_Digit_String for PTS IMT (continued)

Digits Type	New Call Parameters
INFO	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (pts_imt, ss7_imt)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (callingPartyAddress, calledPartyAddress for DP a)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, partitionNumber for DP b and c personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>infoCollected (callingPartyAddress, calledPartyAddress, authorizationCode, personalIdentificationNumber, AccountCode for DP d)</p> <p>STS (STS from auth for DP a, b, and c; STS from AUTH for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p> <p>SignalingInfo (IAM - if SS7)</p>
XLAADDR	<p>switchId, portsInTheCall, bearerCapability, signalingType, agentType</p> <p>accessType (4w_dal, 2w_dal)</p> <p>callType (from address pretranslations for all the DPs)</p> <p>InfoCollected (authorizationCode, calledPartyAddress, personalIdentificationNumber, accountCode for DP a, b, c, e, f, and g)</p> <p>infoCollected (callingPartyAddress (PANI), authorizationCode, personalIdentificationNumber, calledPartyAddress for DP d)</p> <p>STS (STS from auth for DP a, b, c, e, f, and g; STS from ANI (PANI) for DP d)</p> <p>Partition (OPART and TPART (datafilled by key STS))</p>
—end—	

The following New Call parameters are applicable to the PRI agent. The Dial Plan for PRIs may be one of the following:

- 1 DTMF: Auth + Pin + Address + Acct (PRI auth billed, auth dialed)
- 2 DTMF: Address + Acct (PRI auth billed, auth filed with acct)
- 3 DTMF: Address + Pin + Acct (PRI CLID billed, with PIN and Account)
- 4 Plain PRI with an Incoming Q.931 SETUP Message (In the Plain PRI, the following parameters will be present: the Called Party Number (Address) and the Calling Party Number (ANI).)

Table 22-14
Collect_Information, Information_Collected, PRI_B_Channel for PRI

Digits Type	New Call Parameters
ADDR	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP a and b; STS from ANI for DP c and d) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b) infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c) infoCollected (calledPartyAddress, calledPartyAddress (ANI) for DP d)
—continued—	

Table 22-14
Collect_Information, Information_Collected, PRI_B_Channel for PRI (continued)

Digits Type	New Call Parameters
ADIN	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP a and b) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b)
CLID	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from ANI for DP c) signalingInfo (SETUP) InfoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c)
—end—	

Table 22-15
Analyze_Information, Information_Analyzed, Custom_Dialing_Plan for PRI

Digits Type	New Call Parameters
ADDR	<p>switchId, portsInTheCall, bearerCapability</p> <p>accessType (pri)</p> <p>callType (from address pretranslations)</p> <p>STS (STS from auth for DP a and b; STS from ANI for DP c and d)</p> <p>signalingInfo (SETUP)</p> <p>InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a)</p> <p>InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b)</p> <p>infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c)</p> <p>infoCollected (calledPartyAddress, calledPartyAddress (ANI) for DP d)</p>
XLAADDR	<p>switchId, portsInTheCall, bearerCapability</p> <p>accessType (pri)</p> <p>callType (from address pretranslations)</p> <p>STS (STS from auth for DP a and b; STS from ANI for DP c and d)</p> <p>signalingInfo (SETUP)</p> <p>InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a)</p> <p>InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b)</p> <p>infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c)</p> <p>infoCollected (calledPartyAddress, calledPartyAddress (ANI) for DP d)</p>

Table 22-16
Analyze_Information, Information_Analyzed, Specific_Digit_String for PRI

Digits Type	New Call Parameters
ADDR	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP a and b; STS from ANI for DP c and d) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b) infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c) infoCollected (calledPartyAddress, calledPartyAddress (ANI) for DP d)
ADIN	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP a and b) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b)
Note: There is no CIC-specific behavior for calls on PRI origination.	
—continued—	

Table 22-16
Analyze_Information, Information_Analyzed, Specific_Digit_String for PRI (continued)

Digits Type	New Call Parameters
ANI	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP c and d) signalingInfo (SETUP) InfoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c) InfoCollected (calledPartyAddress, callingPartyAddress (ANI) for DP d)
INFO	switchId, portsInTheCall, bearerCapability accessType (pri) STS (STS from auth for DP a and b; STS from ANI for DP c and d) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b) infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c) infoCollected (calledPartyAddress, callingPartyAddress (ANI for DP d)
Note: There is no CIC-specific behavior for calls on PRI origination.	
—continued—	

Table 22-16
Analyze_Information, Information_Analyzed, Specific_Digit_String for PRI (continued)

Digits Type	New Call Parameters
XLAADDR	switchId, portsInTheCall, bearerCapability accessType (pri) callType (from address pretranslations) STS (STS from auth for DP a and b; STS from ANI for DP c and d) signalingInfo (SETUP) InfoCollected (calledPartyAddress, authorizationCode, personalIdentificationNumber, accountCode for DP a) InfoCollected (calledPartyAddress, authorizationCode, accountCode for DP b) infoCollected (calledPartyAddress, callingPartyAddress (ANI), personalIdentificationNumber, accountCode for DP c) infoCollected (calledPartyAddress, callingPartyAddress (ANI for DP d)
Note: There is no CIC-specific behavior for calls on PRI origination.	
—end—	

New Call – AFTER Querying the AIN SCP

Once the call queries the AIN SCP, the SCP may return a Response with “Connect To SCU” parameter in a Continue TCAP message. If this is the case, then the call enters the server mode and a New Call event notification is sent to the SCU.

All the parameters collected before querying the SCP are included in the New Call event notification sent to the SCU. This information does not include signaling information.

List of terms

ACM	Address Complete Message
ANI	Automatic Number Identification
ANM	ANswer Message
AP	Application Processor
ATM	Asynchronous Transfer Mode
BBF	Blue Box Fraud
CDR	Call Detail Record
CM	Computing Module
CPS	Call Prompter Service
CRID	Call Reference IDentifier
DISC	DISConnect Message
DTMF	Dual Tone Multi Frequency

EIU	Ethernet Interface Unit
ESP	Enhanced Services Platform
ETH	Ethernet
FDDI	Fiber Distributed Data Interface
FLIS	Fiber Link Interface Shelf
FPE	Feature Processing Environment
FSM	Finite State Machine
FTS	File Transfer System
IAM	Initial Address Message
IP	Internet Protocol
ISDN	Integrated Services Digital Network
ISUP	ISDN User Part
LOPER	Low Overhead Protocol Encoding Rules
LPP	Link Peripheral Processor
LSB	Least Significant Byte
MCCS	Mechanized Calling Card Services

MS	Message Switch
MSB	Most Significant Byte
MMI	Man Machine Interface
NAV	Network Application Vehicle
NFS	Network File System
PDU	Protocol Data Unit
PER	Protocol Encoding Rules
PIN	Personal Identification Number
PM	Peripheral Module
PPCD	Play Prompt, Collect Digits & Report
PRI	Primary Rate Interface
PSN	Programmable Service Node
REL	RElease Message
RMS	Remote Messaging System
SCP	Service Control Point
SCU	Service Control Unit

SF	Single Frequency
SS7	Signalling System Number 7
STR	Specialized Tone Receiver
SWACT	SWitch ACTivity
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UTR	Universal Tone Receiver

Ordering information

Use the following table for ordering Nortel Networks NTPs (Northern Telecom Publications) and PCLs (Product Content Loads):

Type of product	Source	Phone	Cost
Technical documents (paper or CD-ROM)	Nortel Networks Product Documentation	1-877-662-5669, Option 4 + 1	Yes
Individual NTPs (paper)	Merchandising Order Service	1-800-347-4850	Yes
Marketing documents	Sales and Marketing Information Center (SMIC)	1-800-4NORTEL (1-800-466-7835)	No
PCL software	Nortel Networks	Consult your Nortel Networks sales representative	Yes

When ordering publications on CD

Please have the CD number and software version available, for example, **HLM-2621-001 02.03**.

When ordering individual paper documents

Please have the document name and number available, for example, **297-2621-380, UCS DMS-250 Programmable Service Node (PSN) Application Guide**.

When ordering software

Please have the eight-digit ordering code, for example, **UCSE0012**, as well as the ordering codes for the features you wish to purchase. Contact your Nortel Networks representative for assistance.

Digital Switching Systems
UCS DMS-250
Programmable Service Node (PSN)
Application Guide

Product Documentation—Dept 3423
Nortel Networks
P.O. Box 13010
RTP, NC 27709–3010
1-877-662-5669, Option 4 + 1

Copyright © 1996, 1997, 1999 Northern Telecom,
All Rights Reserved

NORTEL NETWORKS CONFIDENTIAL: The information contained herein is the property of Nortel Networks and is strictly confidential. Except as expressly authorized in writing by Nortel Networks, the holder shall keep all information contained herein confidential, shall disclose the information only to its employees with a need to know, and shall protect the information, in whole or in part, from disclosure and dissemination to third parties with the same degree of care it uses to protect its own confidential information, but with no less than reasonable care. Except as expressly authorized in writing by Nortel Networks, the holder is granted no rights to use the information contained herein.

Information is subject to change without notice. Nortel Networks reserves the right to make changes in design or components as progress in engineering and manufacturing may warrant.

DMS, DMS-250, MAP, NORTEL, NORTEL NETWORKS, NORTHERN TELECOM, NT, and SUPERNODE are trademarks of Nortel Networks.
Publication number: 297-2621-380
Product release: UCS09
Document release: Standard 04.03
Date: August 1999
Printed in the United States of America