Critical Release Notice

Publication number: 297-2621-370
Publication release: Standard 10.03

The content of this customer NTP supports the SN07 (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.

Purple: Applies to new or modified content for SN07 (DMS) that is valid through the current release.

Attention!
Adobe Acrobat Reader 5.0 or higher is required to view bookmarks in color.

Publication History

September 2004

For the SN07 (DMS) release, 10.03, the following changes were added:

Volume 1

Added additional NetworkBuilder-related data schema information to the CAINPARM table to address CR Q00816405.

Volume 2

No changes

Volume 3

Added notes for CAIN parameter TRTMTCD_COMPCODE_ZAPPED_ZERO to address CR Q00816405.

Volume 4

No changes

Volume 5

No changes

September 2003

For the SN06 (DMS) release, 10.02, the following changes were added:

Volume 1

SN06 (DMS) Standard release 10.02. Added LNP_EVALUATE_AFTER_OTC_CIC information per CR Q00 509677-06.

Volume 2

No changes

Volume 3

No changes

Volume 4

No changes

Volume 5

No changes

297-2621–370

Digital Switching Systems

UCS DMS-250

NetworkBuilder Application Guide, Volume 3 of 5

UCS17 Standard 10.01 July 2002

NORTEL NETWORKS™

How the world shares ideas.

Digital Switching Systems

UCS DMS-250

NetworkBuilder Application Guide, Volume 3 of 5

Publication number: 297-2621–370
Product release: UCS17
Document release: Standard 10.01
Date: July 2002

Contents

Volume 1 of 5

Figures

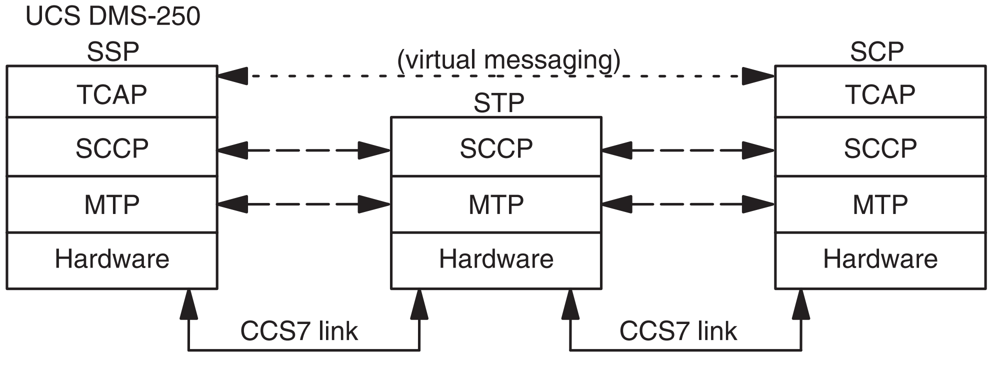
TCAP messaging

Advanced intelligent networking (AIN) uses Transaction Capabilities Application Part (TCAP) messaging for communicating between the service switching point (SSP) and the service control point (SCP). TCAP messages are transported by common channel signaling #7 (CCS7) links.

Figure 1-1 shows the four-layer CCS7 stack that Carrier AIN (CAIN) uses to communicate between the UCS DMS-250 switch and the SCP. The CCS7 stack is based on Open Systems Interconnection (OSI) logic.

ATTENTION
The CCS7 stack must reside on the STP and the SCP for communication links to function properly.

Figure 1-1
Four-layer CCS7 stack



CCS7 – common channel signaling #7
MTP – message transfer part
SCCP – signaling connection control part
SCP – service control point
SSP – service switching point
STP – signal transfer point
TCAP – transaction capabilities application part

The CCS7 stack consists of the following components:

- hardware — UCS DMS-250 switch, STP, or SCP
- message transfer part (MTP) — defines the point-to-point signaling, directs the signal from one piece of hardware to another
- signaling connection control part (SCCP) — defines the end-to-end signaling, directs the signal from the origination to destination
- transaction capabilities application part (TCAP) — provides application message transport

CAIN message types

The following TCAP messages are divided into three categories of messages: call-related messages, non-call related messages, and error messages.

Note: TCAP messages are handled differently for LNP queries and for AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide or UCS DMS-250 CAIN/FlexDial Interactions* for more information.

CAIN call-related messages

Call-related messages are used to drive call processing. At certain points in a call, the AIN application on the switch sends a message to the SCP requesting instructions regarding the disposition of the call in progress (for example, routing information). The SCP may provide the necessary information to the switch and terminate the dialog, or may extend the dialog by requesting additional information from the switch before providing final instructions to the switch. This allows call processing logic to be migrated from the switch to the SCP, with the switch acting as a terminal or connection server to the SCP applications.

CAIN supports the following call-related messages:

- `Call_Info_From_Resource` (UCS DMS-250 switch to SCP)
- `CTR_Clear` (UCS DMS-250 switch to SCP)
- `Failure_Outcome` (UCS DMS-250 switch to SCP)
- `Info_Analyzed` TDP-Request (query) message (UCS DMS-250 switch to SCP)
- `Info_Collected` TDP-Request (query) message (UCS DMS-250 switch to SCP)
- `Network_Busy` TDP-Request (query) or EDP-Request message (UCS DMS-250 switch to SCP)
- `O_Abandon` EDP-Request message (UCS DMS-250 switch to SCP)

- `O_Answer` EDP-Notification message (UCS DMS-250 switch to SCP)
- `O_Called_Party_Busy` TDP-Request (query) or EDP-Request message (UCS DMS-250 switch to SCP)
- `O_Disconnect` EDP-Notification message (UCS DMS-250 switch to SCP)
- `O_Disconnect` EDP-Request message (UCS DMS-250 switch to SCP)
- `O_Feature_Requested` TDP-Request (query) message (UCS DMS-250 switch to SCP)
- `O_Mid_Call` TDP-Request (query) message (UCS DMS-250 switch to SCP)
- `O_No_Answer` TDP-Request (query) or EDP-Request message (UCS DMS-250 switch to SCP)
- `Origination_Attempt` TDP-Request (query) message (UCS DMS-250 switch to SCP)
- `O_Term_Seized` EDP-Notification message (UCS DMS-250 switch to SCP)
- `Resource_Clear` (UCS DMS-250 switch to SCP)
- `Termination_Attempt` TDP-Request message (UCS DMS-250 switch to SCP)
- `Timeout` EDP-Request message (UCS DMS-250 switch to SCP)
- `Acknowledge` (SCP to UCS DMS-250 switch)
- `Analyze_Route` (SCP to UCS DMS-250 switch)
- `Authorize_Termination` (SCP to UCS DMS-250 switch)
- `Call_Info_To_Resource` (SCP to UCS DMS-250 switch)
- `Cancel_Resource_Event` (SCP to UCS DMS-250 switch)
- `Collect_Information` (SCP to UCS DMS-250 switch)
- `Connect_To_Resource` (SCP to UCS DMS-250 switch)
- `Continue` (SCP to UCS DMS-250 switch)
- `Disconnect` (SCP to UCS DMS-250 switch)
- `Disconnect_Leg` (SCP to UCS DMS-250 switch)
- `Merge_Call` (SCP to UCS DMS-250 switch)
- `Originate_Call` (SCP to UCS DMS-250 switch)
- `Send_To_Resource` (SCP to UCS DMS-250 switch)

Call_Info_From_Resource message

The switch sends a **`Call_Info_From_Resource`** message to provide intermediate information from the IP to the SCP during an active IP connection.

CTR_Clear message

The switch sends a **`CTR_Clear`** in response to a conversational **`Connect_To_Resource`** message from the SCP or in response to **`Cancel_Resource_Event`**. Refer to Volume 4, "Conversational processing," for more information.

Failure_Outcome message

The **`Failure_Outcome`** message informs the SCP that a failure event was detected in processing a **`Disconnect_Leg`**, **`Merge_Call`**, or **`Originate_Call`** message.

Info_Analyzed TDP-Request (query) message

The switch sends an **`Info_Analyzed`** TDP-Request (query) message to the SCP when *Specific_Feature_Code*, *Customized_Dialing_Plan*, *Specific_Digit_String*, or *Office_Code* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *Specific_Feature_Code*, *Customized_Dialing_Plan*, *Specific_Digit_String*, or *Office_Code* triggers when ***Info_Analyzed*** is encountered during the **Analyze_Information** PIC. Refer to Volume 1, Chapter 5, "Analyze_Information PIC," for more information.

Info_Collected TDP-Request (query) message

The switch sends an **`Info_Collected`** TDP-Request (query) message to the SCP when *Offhook_Delay*, *Shared_Interoffice_Trunk*, or *PRI_B-Channel* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *Offhook_Delay*, *Shared_Interoffice_Trunk*, or *PRI_B-Channel* trigger when ***Info_Collected*** is encountered during the **Collect_Information** PIC. Refer to Volume 1, Chapter 4, "Collect_Information PIC," for more information.

Network_Busy TDP-Request (query) or EDP-Request message

The switch sends a **`Network_Busy`** TDP-Request (query) message to the SCP when *Network_Busy* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *Network_Busy* trigger when ***Network_Busy*** TDP is encountered during the **Select_Route** PIC. Refer to Volume 1, Chapter 6, "Select_Route PIC," for more information.

When EDPs are active during the **Select_Route** PIC, and the ***Network_Busy*** EDP is encountered, the switch may send a **`Network_Busy`** EDP-Request message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

O_Abandon EDP-Request message

The SCP sends the **`O_Abandon`** EDP-Request message to the switch to indicate that the controlling leg went on-hook before the called party answered.

O_Answer EDP-Notification message

When the ***O_Answer*** EDP is active and the terminating party answers the call, the switch sends an **`O_Answer`** EDP-Notification message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

O_Called_Party_Busy TDP-Request (query) or EDP-Request message

The switch sends an **`O_Called_Party_Busy`** TDP-Request (query) message to the SCP when *O_Called_Party_Busy* trigger criteria are met and the provisioned action is QUERY. Call processing evaluates the *O_Called_Party_Busy* trigger when ***O_Called_Party_Busy*** TDP is encountered during the **Send_Call** PIC. Refer to Volume 1, Chapter 7, "Send_Call PIC," for more information.

When EDPs are active during the **Send_Call** PIC, and the ***O_Called_Party_Busy*** EDP is encountered, the switch may send an **`O_Called_Party_Busy`** EDP-Request message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

O_Disconnect EDP-Notification or EDP-Request message

When the ***O_Disconnect*** EDP is armed as a notification and the calling or called party disconnects, the switch sends an **`O_Disconnect`** EDP-Notification to the SCP. Call processing evaluates the *O_Disconnect* event when the ***O_Disconnect*** EDP is encountered. The ***O_Disconnect*** EDP can only be detected after the call has been answered and the call has entered the **O_Active** or **O_Suspended** PICs. Refer to Volume 1, Chapter 9, "O_Active and O_Suspended PICs," for more information.

When the ***O_Disconnect*** EDP is armed as a request, the switch sends the **`O_Disconnect`** EDP-Request message to the SCP to indicate that the ***O_Disconnect*** requested event was detected.

O_Feature_Requested TDP-Request (query) message

The switch sends an **`O_Feature_Requested`** TDP-Request (query) message to the SCP when *O_Feature_Requested* trigger criteria is met and the provisioned action is FEAT. Call processing evaluates the *O_Feature_Requested* trigger when ***O_Feature_Requested*** is encountered during the **Collect_Information** PIC. Refer to Volume 1, Chapter 4, "Collect_Information PIC," for more information.

O_Mid_Call TDP-Request (query) or EDP-Request message

The switch sends an **`O_Mid_Call`** TDP-Request (query) message to the SCP when *O_IEC_Reorigination* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *O_IEC_Reorigination* trigger when ***O_Mid_Call*** TDP is encountered during the **Send_Call**, **O_Alerting**, **O_Active**, and **O_Suspended** PICs. Refer to Volume 1, Chapter 7, "Send_Call PIC," Volume 1, Chapter 8, "O_Alerting PIC," and Volume 1, Chapter 9, "O_Active and O_Suspended PICs," for more information.

When EDPs are active during the **Send_Call**, **O_Alerting**, or **O_Active** PICs, and the ***O_Mid_Call*** EDP is encountered, the switch may send an **`O_Mid_Call`** EDP-Request message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

O_No_Answer TDP-Request (query) or EDP-Request message

The switch sends an **`O_No_Answer`** TDP-Request (query) message to the SCP when *O_No_Answer* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *O_No_Answer* trigger when ***O_No_Answer*** TDP is encountered during the **O_Alerting** PIC. Refer to Volume 1, Chapter 8, "O_Alerting PIC," for more information.

When EDPs are active during the **O_Alerting** PIC, and the ***O_No_Answer*** EDP is encountered, the switch may send an **`O_No_Answer`** EDP-Request message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

Origination_Attempt TDP-Request (query) message

The switch sends an **`Origination_Attempt`** TDP-Request (query) message to the SCP when *Off_Hook_Immediate* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *Off_Hook_Immediate* trigger when ***Origination_Attempt*** is encountered during the **O_Null** PIC. Refer to Volume 1, Chapter 3, "O_Null PIC," for more information.

O_Term_Seized EDP-Notification message

When the ***O_Term_Seized*** EDP is active and the terminating trunk is seized by the UCS DMS-250 switch, the switch sends an **`O_Term_Seized`** EDP-Notification message to the SCP. Refer to Chapter 3, "Event Processing," for more information.

Resource_Clear message

The switch sends a **`Resource_Clear`** in response to a conversational **`Send_To_Resource`** message from the SCP or in response to **`Cancel_Resource_Event`**. Refer to Volume 4, "Conversational processing," for more information.

Termination_Attempt TDP-Request message

The switch sends a **`Termination_Attempt`** TDP-Request (query) message to the SCP when *Termination_Attempt* trigger criteria is met and the provisioned action is QUERY. Call processing evaluates the *Termination_Attempt* trigger when the ***Termination_Attempt*** TDP is encountered during the **T_Null** PIC. Refer to Volume 1, Chapter 10, “T_Null PIC,” for more information.

Timeout EDP-Request message

When the ***O_Mid_Call*** EDP is active and the Timeout timer expires, the switch sends the **`Timeout`** EDP-Request message to the SCP. Refer to Volume 1, Chapter 9, “O_Active and O_Suspended PICs,” for more information.

Acknowledge message

The **`Acknowledge`** message enables the SCP to respond to informational messages from the switch without requesting additional Call Party Handling operations.

Analyze_Route message

The SCP sends an **`Analyze_Route`** message in response to a request originated by the switch. This message directs the switch to continue call processing under the direction of the SCP. When received in conversation, this message should be accompanied by the **`Request_Report_BCM_Event`** component. Refer to Chapter 10, “Incoming CAIN messages,” and Chapter 3, “Event processing,” for more information.

Authorize_Termination message

The SCP sends the **`Authorize_Termination`** message in response to a **`Termination_Attempt`** TDP-Request message. This message directs the switch to allow the indicated termination to progress through the terminating call model. Refer to Volume 1, Chapter 10, “T_Null PIC,” for more information.

Call_Info_To_Resource message

The SCP sends a **`Call_Info_To_Resource`** message in response to a **`Call_Info_From_Resource`** message received from the switch during an active IP connection.

Cancel_Resource_Event message

The SCP sends a **`Cancel_Resource_Event`** when the switch is processing an outstanding **`Send_To_Resource`** operation initiated by the SCP. The **`Cancel_Resource_Event`** directs the switch to discontinue caller interaction and report to the SCP for further instructions. Refer to Volume 4, “Conversational processing,” for more information.

Collect_Information message

The SCP sends the **`Collect_Information`** message in response to a **`O_Mid_Call`** TDP-Request message. This message directs the switch to continue in-switch call processing as if the call had not sent the request to the SCP. When received in conversation, this message should be accompanied by the **`Request_Report_BCM_Event`** component. Refer to Chapter 10, "Incoming CAIN messages," and Chapter 3, "Event processing," for more information.

Connect_To_Resource message

The SCP sends the **`Connect_To_Resource`** message in response to an **`O_Mid_Call`** TDP-Request message. This message may direct the switch to play a resource, collect digits, or route to an IP. Refer to Chapter 10, "Incoming CAIN messages," or Volume 4, "Conversational processing," for more information.

Continue message

The SCP sends the **`Continue`** message in response to either a request originated by the switch, or a **`CTR_Clear`** message at the ***O_Mid_Call*** EDP. This message directs the switch to continue in-switch call processing as if the call had not sent a request to the SCP. When received in conversation, this message should be accompanied by the **`Request_Report_BCM_Event`** component. Refer to Chapter 10, "Incoming CAIN messages," and Chapter 3, "Event processing," for more information.

Disconnect message

The SCP sends the **`Disconnect`** message in response to a request originated by the switch. This message directs the switch to apply treatment. Refer to Chapter 10, "Incoming CAIN messages," for more information.

Disconnect_Leg message

The **`Disconnect_Leg`** message directs the switch to release a specified leg involved in a call.

Merge_Call message

When the switch receives a **`Merge_Call`** message (in Call Configuration 6) it attempts to allocate a three port conference port. If no conference ports are available, the switch sends a **`Failure_Outcome`** message with a ***`FailureCause`*** of `unavailableResources`.

Originate_Call message

The **`Originate_Call`** message enables the switch to place a stable two-party call on hold and originate an associated call to another party.

Send_To_Resource message

The SCP sends the `Send_To_Resource` message in response to a request from the switch. This message may direct the switch to play a resource, collect digits, or route to an IP. Refer to Chapter 10, “Incoming CAIN messages,” or Volume 4, “Conversational processing,” for more information.

CAIN non-call related messages

Non-call related messages are used to set up event detection points (EDP) for a call in progress, for automatic code gapping, and termination notification.

CAIN supports the following non-call related messages:

- `ACG` (SCP to UCS DMS-250 switch)
- `ACG_Global_Ctrl_Restore` (SCP to UCS DMS-250 switch)
- `ACG_Global_Ctrl_Restore_Success` (UCS DMS-250 switch to SCP)
- `ACG_Overflow` (UCS DMS-250 switch to SCP)
- `Furnish_AMA_Information` (SCP to UCS DMS-250 switch)
- `Request_Report_BCM_Event` (SCP to UCS DMS-250 switch)
- `Send_Notification` (SCP to UCS DMS-250 switch)
- `Termination_Notification` (UCS DMS-250 switch to SCP)

ACG message

The SCP sends an `ACG` message to specify an ACG control to be added, updated, or deleted from the ACG control list. Refer to Chapter 10, “Incoming CAIN messages,” for more information.

ACG_Global_Ctrl_Restore message

The SCP sends an `ACG_Global_Ctrl_Restore` message to request the removal of a large selection of controls from the ACG control list. Refer to Chapter 10, “Incoming CAIN messages,” for more information.

ACG_Global_Ctrl_Restore_Success message

The switch sends an `ACG_Global_Ctrl_Restore_Success` message to the SCP when the switch has successfully completed an ACG Global Restore Request. Refer to Chapter 6, “Outgoing CAIN messages,” for more information.

ACG_Overflow message

The switch sends an `ACG_Overflow` message to the SCP when a new ACG control has been discarded. A new ACG control is discarded when it cannot be added to the ACG control list due to a lack of available spaces. Refer to Chapter 6, "Outgoing CAIN messages," for more information.

Furnish_AMA_Information message

The SCP sends a `Furnish_AMA_Information` message when Bellcore AMA Format (BAF) modules are to be included in the switch CDR. The switch checks the structure of the message, and after examining the message, the switch captures the contents of the billing information in the CDR. The UCS DMS-250 switch supports this non-call related message with the `Analyze_Route` and `Send_to_Resource` call-related messages. Refer to Chapter 10, "Incoming CAIN messages," in this volume for more information.

Request_Report_BCM_Event message

The SCP sends a `Request_Report_BCM_Event` message to indicate to the switch which EDPs should be armed to send an EDP-Request or EDP-Notification to the SCP. Either the `Analyze_Route`, `Continue`, or `Collect_Information` call-related message must be the first component in the conversation package. Refer to Chapter 3, "Event processing," for more information.

Send_Notification message

The SCP sends a `Send_Notification` message to instruct the switch to send a `Termination_Notification` when the call is released. Refer to Chapter 6, "Outgoing CAIN messages," for more information.

Termination_Notification message

The switch may send a `Termination_Notification` when the call is released. This is done in response to a `Send_Notification` message from the SCP. Refer to Chapter 6, "Outgoing CAIN messages," for more information.

CAIN error messages

Error messages are used to respond to error situations, such as receipt of an unexpected communication or the inability of the switch or SCP to carry out a particular function.

CAIN supports the following error messages:

- `Application_Error`
- `Close`
- `Failure_Report`
- `Report_Error`

Application_Error message

The switch or the SCP may send an `Application_Error` message in response to an error detected in the CAIN TCAP application logic. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Close message

The SCP or the switch can send a `Close` message. The switch sends a `Close` message to terminate an open transaction (for example, to end EDP processing).

The SCP sends a `Close` message to indicate a nonfatal unexpected communication error. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Failure_Report message

The switch or the SCP may send a `Failure_Report` message in response to another message when either the switch or the SCP is unable to perform a requested operation. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Report_Error message

The switch or the SCP sends a `Report_Error` message when an error is detected in a response message or when the T1 timer expires, indicating a fatal application error. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Note: The office parameter CAIN_T1_TIMEOUT in table CAINPARM (CAIN Parameter) sets the timeout.

IN/1 message types

The following TCAP messages are divided into three categories of messages: call-related messages, non-call related messages, and error messages.

Note: TCAP messages are handled differently for LNP queries and for AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide or UCS DMS-250 CAIN/FlexDial Interactions* for more information.

IN/1 call-related messages

Call-related messages are used to drive call processing. At certain points in a call, the IN application on the switch sends a message to the SCP requesting instructions regarding the disposition of the call in progress (for example, routing information). The SCP may provide the necessary information to the switch and terminate the dialog, or may extend the dialog by requesting additional information from the switch before providing final instructions to the switch. This allows call processing logic to be migrated from the switch to the SCP, with the switch acting as a terminal or connection server to the SCP applications.

IN/1 supports the following call-related messages:

- **`Play_Announcement`** (SCP to UCS DMS-250 switch)
- **`Connect`** (SCP to UCS DMS-250 switch)
- **`Start`** (UCS DMS-250 switch to SCP)

Play_Announcement message

The SCP sends a **`Play_Announcement`** message when it is unable to respond with routing instructions due to an irregular user action.

Connect message

The SCP sends a **`Connect`** message to provide the switch with call routing directions.

Start message

The switch sends a **`Start`** message when the call encounters a toll-free service trigger interaction and the trigger action is QUERY.

IN/1 non-call related messages

Non-call related messages are used by the SCP to adjust parameters in the switch that may be used to check trigger settings or disable particular triggers, to set up event detection points (EDP) for a call in progress, and for automatic code gapping.

IN/1 supports the following non-call related messages:

- **`ACG`** (SCP to UCS DMS-250 switch)
- **`Termination`** (SCP to UCS DMS-250 switch)
- **`Termination_Information`** (UCS DMS-250 switch to SCP)

ACG message

The SCP sends an **`ACG`** message to specify an ACG control to be added, updated, or deleted from the ACG control list.

Termination message

The SCP sends a **`Termination`** message to request that the switch send a **`Termination_Information`** message when the call is released.

Termination_Information message

The switch sends a **`Termination_Information`** when the call is released. This is done in response to a **`Termination`** message from the SCP.

IN/1 error messages

Error messages are used to respond to error situations, such as receipt of an unexpected communication or the inability of the switch or SCP to carry out a particular function.

IN/1 supports the following error messages:

- **`Reject`**
- **`Report_Error`**
- **`Return_Error`**

Reject message

Either the switch or the SCP can send a **`Reject`** message when protocol errors are encountered in received messages.

Report_Error message

The switch sends a **`Report_Error`** message if a received response message indicates the call should be completed through an IXC that does not serve the originating LATA, or if it detects an error in an **`ACG`** component of the response message.

Return_Error message

The SCP sends a **`Return_Error`** message when it cannot provide routing instruction because of improper or invalid data in a query message.

Global title use

In order to allow partitioning of large databases, global title types are available: CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, and CAIN_OFCD_GT.

There are two global title formats possible. The global title format is determined by the value of the global title indicator. A global title indicator of 0010 causes an additional octet to be encoded containing the numbering plan and encoding scheme for the global title digits.

Although Bellcore states that the STP only has to support the 0010 global title indicator, CAIN supports both the 0001 and 0010 global title indicators. The following parameters in table CAINPARM are used to indicate the format for each of the corresponding CAIN global titles:

- CLID_GT_FORMAT

- ADDR_GT_FORMAT
- FEAT_GT_FORMAT
- OFCD_GT_FORMAT

If the parameter is set to IMPLICIT, a global title indicator of 0100 is used. If the parameter is set to ENHANCED, a global title indicator of 0001 is used.

CAIN_CLID_GT

The calling line identification global title translation type (CAIN_CLID_GT) allows convenient TCAP traffic segregation based on the caller's geographic location, as indicated by the caller's CLID, ANI, or home number plan area (NPA), as datafilled on the originating agency.

The contents of the global title address is dependent on the nature of address, as follows:

- When the nature of address is NATL, the global title address is the same as the address transmitted in the ***`CallingPartyID`*** parameter.
- When the nature of address is INTL, the switch derives a 3-digit NPA from in-switch datafill for use with this global title type.

CAIN provides either a 10-digit CLID or ANI from FGD and PRI agents. For DAL, IMT, AXXESS agents, a 3-digit NPA is derived from in-switch datafill.

Note: Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information on AXXESS agents.

When the switch sends a message to the SCP, the called party address indicator instructs the STP to perform global title translations.

The switch obtains the global title translation type for the local type CAIN_CLID_GT in tables C7GTT and C7GTTYPE. The encoding scheme is determined by the CLID_GT_FORMAT parameter in table CAINPARM.

CAIN_ADDR_GT

The address global title translation type (CAIN_ADDR_GT) allows traffic segregation based on the called party's identification.

CAIN provides the address digits collected from the originating agency.

The switch obtains the global title translation type for the local type CAIN_ADDR_GT in tables C7GTT and C7GTTYPE. The encoding scheme is determined by the CLID_GT_FORMAT parameter in table CAINPARM.

Note: If querying on an IMT with an I3PA dialing plan, the info and partition digits will be included in the global title.

CAIN_FEAT_GT

The feature global title translation type (CAIN_FEAT_GT) allows traffic segregation based upon a provisionable feature ID.

The contents of the global digits features may be provisioned in table CAINPARM for the CAIN_DEFAULT_GT, CAIN_DEFAULT_OVERFLOW_GT, and ACG_OVERFLOW_GT, and in the individual trigger tables using the ACGOVFLGT, GT, or T1OVFLGT option.

CAIN_OFCD_GT

The office code global title translation type allows traffic segregation for local number portability (LNP). Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on LNP.

ACG overflow

The ACG overflow (ACGOVFLGT) option, or ACG_OVERFLOW_GT parameter in table CAINPARM may provide additional global titles for ACG overflow queries.

T1 timeout overflow

The T1 overflow (T1OVFLGT) option or CAIN_DEFAULT_OVERFLOW_GT parameter in table CAINPARM may provide additional global titles for T1 overflow queries.

Restrictions and limitations

The following limitations apply to CAIN TCAP messaging.

- TCAP messages are only supported on CCS7 links.
- Only single unit data SCCP messages are supported. Segmented messages are not supported. (Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.)
- CAIN is based on ANSI Issue 2 TCAP (ANSI T1.114-1992).
- TCAP dialog portion is not supported.
- If an IN/1 SCP receives a CAIN TCAP message, decoding of the message will fail.
- If a CAIN SCP receives an IN/1 TCAP message, decoding of the message will fail.

Note: Other limitations and restrictions apply to CAIN TCAP messaging for LNP queries and for AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide or UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Errors

The following error types can be detected in association with TCAP messaging:

- transaction protocol errors
- component protocol errors
- application errors
- failures

Transaction protocol errors

Transaction protocol errors are detected when the transaction portion of a TCAP message does not conform to the defined protocol. Table 1-1 shows the protocol error and how it is handled by call processing.

Table 1-1
Transaction protocol errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Unrecognized transaction identifier	Conversation	CAIN201	Yes	Abort package sent to SCP
Unrecognized transaction identifier	Response	CAIN201	No	None
Underivable transaction identifier	Query or conversation	CAIN201	No	None
Unrecognized package type, transaction identifier is derived	N/A	CAIN201	Yes	Abort package sent to SCP
Unrecognized package type, transaction identifier is underivable	N/A	CAIN201	No	None
Incorrect transaction portion, underivable transaction identifier	Query or conversation	CAIN201	No	None

—continued—

Table 1-1
Transaction protocol errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
Incorrect transaction portion, underivable transaction identifier	Response	CAIN201	No	None
Badly structured transaction portion, underivable transaction identifier	Query or conversation	CAIN201	No	None
Badly structured transaction portion, underivable transaction identifier	Response	CAIN201	No	None
Permission to release problem	QWOP CWOP	CAIN200 CAIN201	Yes	Error is reported as an application error to the SCP
Unrecognized error	Any	CAIN200 CAIN201	No	None

—end—

Component protocol errors

Component protocol errors are detected when the component portion of a TCAP message does not conform to the defined protocol. Table 1-2 shows the component error and how it is handled by call processing.

Table 1-2
Component protocol errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Badly structured component portion	Unidirectional or Response	CAIN201	No	None
Badly structured component portion	Query or conversation	CAIN201	Yes	If Query package, see note 1; if conversation, note 2

Note 1: An “Unexpected Communication” fatal application error is generated. Refer to the “Fatal application error” section in this chapter for more information.

Note 2: A response package with a reject component is sent to the SCP.

—continued—

Table 1-2
Component protocol errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
Incorrect component portion	Unidirectional or Response	CAIN201	No	None
Incorrect component portion	Query or conversation	CAIN201	Yes	If Query package, see note 1; if conversation, note 2
Unrecognized component	Unidirectional or Response	CAIN201	No	None
Unrecognized component	Query or conversation	CAIN201	Yes	If Query package, see note 1; if conversation, note 2
Unrecognized correlation identifier	Unidirectional or Response	CAIN201	No	None
Unrecognized correlation identifier	Query or conversation	CAIN201	Yes	If Query package, see note 1; if conversation, note 2
Unrecognized operation code	Unidirectional or Response	CAIN200 CAIN201	No	None
Unrecognized operation code	Query or conversation	CAIN200 CAIN201	Yes	Note 1
Missing mandatory parameter	Unidirectional or Response	CAIN201	No	None
Missing mandatory parameter	Query or conversation	CAIN201	Yes	If Query package, see note 1; if conversation, note 2

Note 1: An “Unexpected Communication” fatal application error is generated. Refer to the “Fatal application error” section in this chapter for more information.

Note 2: A response package with a reject component is sent to the SCP.

—end—

Application errors

There are two types of application errors: fatal and nonfatal.

Note: The fatal and nonfatal application errors tables are not all-inclusive.

Fatal application errors

Fatal application errors are detected when NetworkBuilder is unable to continue the operation. The call can still be processed in-switch. Table 1-3 shows the fatal application error and how it is handled by call processing.

Table 1-3
Fatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Unexpected parameter sequence	Unidirectional	CAIN201	No	None
Unexpected parameter sequence	Conversation	CAIN201	Yes	Note 3
Unexpected parameter sequence	Response	CAIN201	Yes	Note 4
Erroneous data received	Unidirectional	CAIN201	No	None
Erroneous data received	Conversation	CAIN201	Yes	Note 3
Erroneous data received	Response	CAIN201	Yes	Note 4
Unexpected communication	Unidirectional	CAIN201	No	None
Unexpected communication	Query	CAIN201	Yes	Note 3
Unexpected communication	Conversation	CAIN201	Yes	Note 3

Note 1: A problem was encountered while sending the query message to the SCP.

Note 2: Tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision the error action; OFFCCODE defaults to ROUTE, but does not require provisioning; all other triggers perform an error action of TREAT.

Note 3: The error is reported to the SCP in a response package as an **`Application_Error`** in a Return Error component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Note 4: The error is reported to the SCP in a unidirectional package as a **`Report_Error`** in an Invoke (Last) component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—continued—

Table 1-3
Fatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
Unexpected communication	Response	CAIN201	Yes	Note 4
`Close` message received in response to a call-related message	Unidirectional	CAIN201	No	None
`Close` message received in response to a call-related message	Conversation	CAIN201	Yes	Note 3
`Close` message received in response to a call-related message	Response	CAIN201	No	None
`Failure_Report` received	Unidirectional	CAIN201	No	None
`Failure_Report` received	Query	CAIN201	Yes	Error reported in an Abort package
`Failure_Report` received	Conversation	CAIN201	Yes	Error reported in Reject component and Response package
`Failure_Report` received	Response	CAIN201	No	None
`Application_Error` message received	Unidirectional	CAIN201	No	None
`Application_Error` message received	Query	CAIN201	Yes	Error reported in an Abort package

Note 1: A problem was encountered while sending the query message to the SCP.

Note 2: Tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision the error action; OFFCCODE defaults to ROUTE, but does not require provisioning; all other triggers perform an error action of TREAT.

Note 3: The error is reported to the SCP in a response package as an `Application_Error` in a Return Error component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Note 4: The error is reported to the SCP in a unidirectional package as a `Report_Error` in an Invoke (Last) component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—continued—

Table 1-3
Fatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
`Application_Error` message received	Conversation	CAIN201	Yes	Error reported in a Reject component and Response package
`Application_Error` message received	Response	CAIN201	No	None
`Report_Error` message received	Unidirectional	none	No	None
`Report_Error` message received	Query	CAIN201	Yes	Error reported in an Abort package
`Report_Error` message received	Conversation	CAIN201	Yes	Note 3
`Report_Error` message received	Response	CAIN201	No	None
T1 timer expired	N/A	CAIN201	Yes	Note 4
Incorrect outgoing error operation code from call processing	N/A	CAIN201	No	None
Invalid transaction state during call processing error processing	N/A	CAIN201	No	None
Invalid event received from application	N/A	CAIN201	No	None

Note 1: A problem was encountered while sending the query message to the SCP.

Note 2: Tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision the error action; OFFCCODE defaults to ROUTE, but does not require provisioning; all other triggers perform an error action of TREAT.

Note 3: The error is reported to the SCP in a response package as an `Application_Error` in a Return Error component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Note 4: The error is reported to the SCP in a unidirectional package as a `Report_Error` in an Invoke (Last) component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—continued—

Table 1-3
Fatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
Invalid operation received from application	N/A	CAIN201	No	None
Message driver error (Note 1)		CAIN200	Note 3	ERRACT in trigger table (Note 2)
Missing mandatory parameter in a Response package		CAIN201	No	ERRACT in trigger table (Note 2)
Missing mandatory parameter in a Conversation package		CAIN201	Yes	ERRACT in trigger table (Note 2) and switch sends a Reject

Note 1: A problem was encountered while sending the query message to the SCP.

Note 2: Tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision the error action; OFFCCODE defaults to ROUTE, but does not require provisioning; all other triggers perform an error action of TREAT.

Note 3: The error is reported to the SCP in a response package as an **`Application_Error`** in a Return Error component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Note 4: The error is reported to the SCP in a unidirectional package as a **`Report_Error`** in an Invoke (Last) component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—end—

Nonfatal application errors

Nonfatal application errors are detected when NetworkBuilder detects minor TCAP errors. However, the call can still be handled by NetworkBuilder call processing. Table 1-4 shows the nonfatal application errors and how they are handled by call processing.

Table 1-4
Nonfatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Unexpected communication (`Send_To_Resource` message with Play Announcement)	Conversation	CAIN101	No	Note
No local transaction identifier for message	Unidirectional, Response, or Abort	CAIN101	No	None
Unrecognized parameter	Response	CAIN101	No	Switch ignores all remaining parameters in the message
Unable to decode extension parameters	Response	CAIN101	No	None
Unable to decode optional parameters	Response	CAIN101	No	None
Unable to encode extension parameters	Response	CAIN101	No	None
Unable to encode optional parameters	Response	CAIN101	No	None
Unable to encode/decode extension parameters (CAIN0200 SOC option not enabled.)	N/A	CAIN102	No	Extension parameters are ignored

Note: The error is reported to the SCP in a response package with a Return Result component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—continued—

Table 1-4
Nonfatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
`Send_To_Resource` requires Response package	Conversation	CAIN100	Yes	Note
`Continue` received in a CC other than CC2	N/A	CAIN101	No	switch processes message as if it were an `Acknowledge`; all parameters are ignored

Note: The error is reported to the SCP in a response package with a Return Result component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

—end—

Caller abandon

Caller abandon occurs when the calling party disconnects while the switch is waiting for an SCP response. Table 1-5 provides information that describes how caller abandon is handled in this situation.

Table 1-5
Caller abandon while waiting for the SCP

Error type	Log generated	Reported to SCP?	Error action performed
Message timer expires after caller abandon	CAIN201	Yes	`Report_Error` sent in a Unidirectional package
SCP response received in a Response package after caller abandon	none	No	None
`Send_To_Resource` received in a Conversation package after caller abandon	none	Yes	`Resource_Clear` sent with ***`ClearCause`*** of `userAbandon`

Note: For information on caller abandon scenarios, refer to Chapter 10, “Incoming CAIN message parameters.”

—continued—

Table 1-5
Caller abandon while waiting for the SCP (continued)

Error type	Log generated	Reported to SCP?	Error action performed
CAIN_T1_TIMEOUT occurs after caller abandon	CAIN201	Yes	Switch sends a **`Report_Error`** in a Unidirectional package
`Connect_To_ Resource` received in a Conversation package after caller abandon	none	Yes	**`CTR_Clear`** sent with ***`ClearCause`*** of `userAbandon`
All other messages received in a Conversation package after caller abandon	none	Yes	**`Close`** with ***`CloseCause`*** of `callTerminated`

Note: For information on caller abandon scenarios, refer to Chapter 10, “Incoming CAIN message parameters.”

—end—

Failure errors

Failure errors indicate resource allocation problems. Table 1-6 shows the failure error and how it is handled by call processing.

Table 1-6
Resource errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Lack of VAMP resources	Unidirectional or Response	CAIN200 VAMP202	No	None
Lack of VAMP resources	Query or conversation	CAIN200 VAMP202	Yes	Note

Note: The error is reported to the SCP in a response package as a Failure_Report with a Return Error component. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Associated logs and OMs

Logs

CAIN100, CAIN101, CAIN102, CAIN200, CAIN201, CAIN300, VAMP202

Operational measurements

CAINMSGR, CAINMSGS, CAINMGR2, CAINOM, VTCAPSNT, VTCAPRCV, VTCAPERR

Automatic Code Gapping

Automatic Code Gapping (ACG) is a method used to reduce the number of outgoing queries from the switch to the SCP.

CAIN ACG

CAIN ACG blocks outgoing queries that contain certain Global Titles. The SCP provides the switch with the Global Titles to be blocked by sending an ACG message containing an ACG control. An ACG control is the element used to decide which queries to block. An ACG control consists of:

- a Global Title Address (GTA)
- a Translation Type (TT)

The switch stores ACG controls in an ACG control list. The switch searches the ACG control list before sending a query message to determine if that query should be blocked.

There are two types of controls that the SCP can send:

- SCP Overload Controls
- Subsystem Management System (SMS) Originated Control Codes

The SCP automatically requests that the switch activate SCP Overload Controls when the SCP detects itself to be overloaded. Table 2-1 shows an example of an SCP control list.

Table 2-1
SCP control list example

GTA	TT	Gap Interval	Gap Duration
214	cain_addr_gt	4 seconds	64 seconds
214684	cain_addr_gt	16 seconds	128 seconds
2146840000	cain_clid_gt	3 seconds	infinity

The SMS is a provisioning and data management system on the SCP that can generate and send messages to the switch. An operator can use the SMS to manually generate SMS Originated Control Codes (SOCCs). SOCCs are controls that restrict certain queries at the switch. Table 2-2 shows an example of a SOCC control list.

Table 2-2
SOCC control list example

GTA	TT	Gap Interval	Gap Duration
972	cain_feat_gt	infinity	32 seconds
2146849999	cain_addr_gt	0 seconds	256 seconds
2146840000	cain_ofcd_gt	0.10 seconds	infinity

Note: Except where noted, ACG functions apply to both CAIN ACG and CAIN LNP ACG. For more LNP-specific ACG information, refer to the *UCS DMS-250 Local Number Portability Application Guide*.

SCP Overload Controls

The SCP Overload Control provides a GTA consisting of 1 to 10 digits (for CAIN). An incoming ACG message to the SSP contains:

- the control
- a ***ControlCauseIndicator*** parameter
- a ***GapDuration*** parameter
- a ***GapInterval*** parameter

ControlCauseIndicator for SCP Overload Controls

The ***ControlCauseIndicator*** parameter specifies whether the control is an SCP Overload Control or a SOCC. The ***ControlCauseIndicator*** parameter also contains the number of digits to which the control is applied. For SCP Overload Controls, the number of digits will be from 1 to 10. Refer to Chapter 12, “Incoming CAIN message parameters,” for more information on the ***ControlCauseIndicator*** parameter.

GapDuration for SCP Overload Controls

The SSP sends the ***`GapDuration`*** parameter to specify the time (in seconds) that an ACG control should remain in effect before it is removed automatically by the SSP. For more information on the ***`GapDuration`*** parameter, refer to Chapter 12, "Incoming CAIN message parameters."

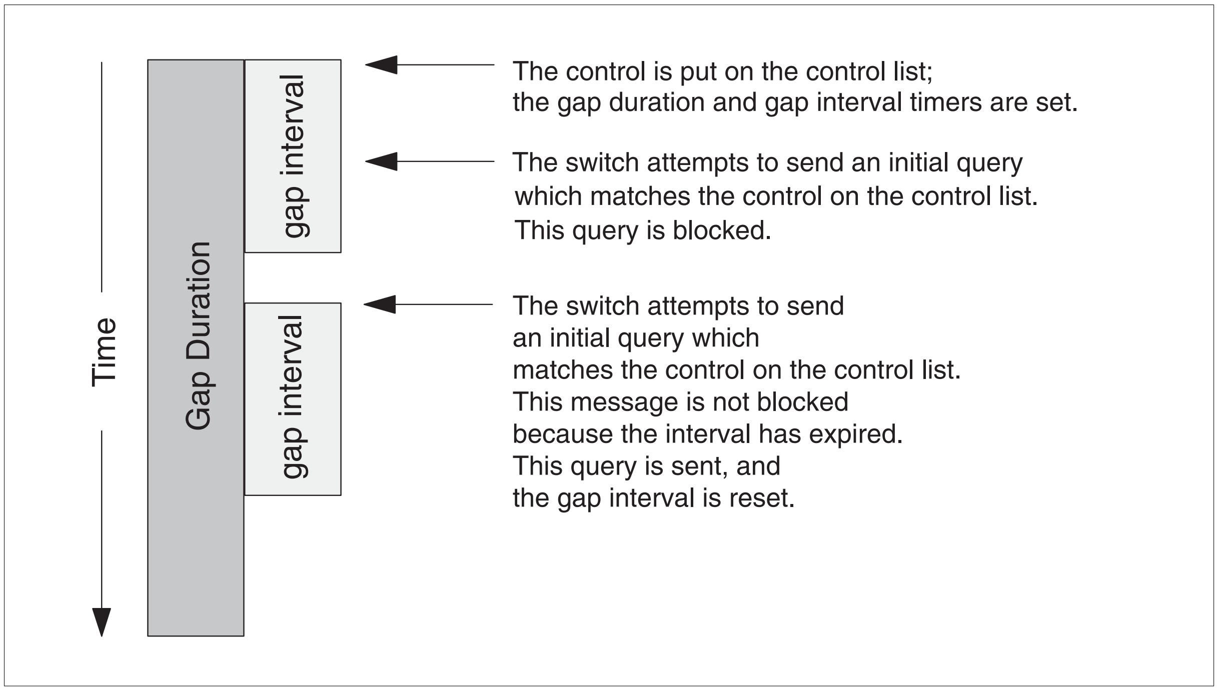
GapInterval for SCP Overload Controls

The ***`GapInterval`*** parameter specifies the minimum amount of time (in seconds) that the SSP must wait before sending another initial query message with the same GT that is under control. Only one query is allowed when the interval expires. For SCP Overload Controls, the Gap Interval will be a Private Gap Interval. For more information on the ***`GapInterval`*** parameter, refer to Chapter 12, "Incoming CAIN message parameters."

Gap duration and gap interval interaction

Figure 2-1 illustrates the relationship between gap intervals and a corresponding gap duration.

Figure 2-1
Gap duration and gap interval interaction


Time
Gap Duration
gap interval
gap interval
The control is put on the control list;
the gap duration and gap interval timers are set.
The switch attempts to send an initial query
which matches the control on the control list.
This query is blocked.
The switch attempts to send
an initial query which
matches the control on the control list.
This message is not blocked
because the interval has expired.
This query is sent, and
the gap interval is reset.

CAIN0900 Auto Code Gapping SOC

The CAIN0900 SOC provides software optionality control for SCP Override Controls. If the CAIN0900 SOC is idle, ACG messages containing SCP Override Controls are ignored. This prevents any SCP Override Controls from being added to the control list. If the SOC is idle and an attempt to add an SCP Override Control is made, a CAIN102 log is generated.

SMS Originated Code Control (SOCC)

A SOCC provides a GTA made up of 1 to 10 digits. The control also contains:

- a ***ControlCauseIndicator*** parameter
- a ***GapDuration*** parameter
- a ***GapInterval*** parameter

These three parameters provide further information on the behavior of the control.

ControlCauseIndicator for SOCCs

The ***ControlCauseIndicator*** parameter specifies whether the control is an SCP Overload Control or a SOCC. The ***ControlCauseIndicator*** parameter also contains the number of digits to which the control is applied. Refer to Chapter 12, “Incoming CAIN message parameters,” for more information on the ***ControlCauseIndicator*** parameter.

GapDuration for SOCCs

The SSP sends the ***GapDuration*** parameter to specify the time (in seconds) that an ACG control should remain in effect before it is removed automatically by the SSP. For more information on the ***GapDuration*** parameter, refer to Chapter 12, “Incoming CAIN message parameters.”

GapInterval for SOCCs

The ***GapInterval*** parameter specifies the average amount of time (in seconds) that the SSP must wait before sending another initial query message with the same GT that is under control. Only one query is allowed when the interval expires. For SOCCs, the Gap Interval will be a National Gap Interval. For more information on the ***GapInterval*** parameter, refer to Chapter 12, “Incoming CAIN message parameters.”

Code controls initiated by operators

ACG controls may be initiated by an operator at the SSP using the ACGCNTRL CI command. When an operator initiates a control, it is placed on the SOCC control list. The CI interface can be used to modify or remove any item on the SOCC control list, whether the item was manually initiated or received in a SOCC ACG message from the SCP.

Entries on the SOCC control list which are initiated by an operator cannot be modified or removed by an ACG message from the SCP.

The operator interface cannot be used to administer controls on the SCP Overload Control list, except for the ACG Global Control Restoral capability.

Note: For more information on the ACGCNTRL CI command, refer to the *UCS DMS-250 Commands Reference Manual*.

Zero-Gap Controls

SOCC controls include "Zero-Gap" controls. Zero-Gap controls are indicated by a "no0Seconds" setting for the ***`GapInterval`*** parameter. Zero-Gap controls provide control exclusion for specific control entries. For example, if a Zero-Gap control is set for a given GTA and TT, any initial query messages that contain that same GTA and TT will be sent to the SCP even if there is another, non Zero–Gap control present to block it from being sent to the SCP. (See "Control Precedence" for more information on control interactions.)

CAIN0901 Manual Code Gapping SOC

The CAIN0901 Manual Code Gapping SOC provides software optionality control for SOCC ACG controls. If the CAIN0901 Manual Code Gapping SOC is in the idle state, no modifications are made to the SOCC control list, whether by CI interface or due to any ACG message received at the switch from the SCP. If the SOC is idle and an attempt to add an SCP control is made, a CAIN102 log is generated.

CAIN0901 has no effect on SCP Overload Controls.

ACG Control Mechanics

ACG Controls are initiated on the SSP when an ACG message is received from the SCP, or when manually initiated at the SSP with the ACGCNTRL CI command. When a control is initiated, both the gap duration and the gap interval timers are started for that control. The gap interval present in the ACG message is an average value specified by the SCP. The actual gap interval used to set the gap timer is selected randomly by the SSP using a uniform distribution. For SCP Overload Control gap intervals, the gap interval value chosen must lie between 90% and 110% of the SCP Overload Control Private Gap Interval value. For SOCC control gap intervals, the gap interval value is chosen must lie between 50% and 150% of the SOCC National Gap Interval value. The random selection is repeated when the gap interval timer is reset.

After the control is initiated, and until the gap interval timer expires, all CAIN initial query attempts with matching GTAs and TTs are blocked from querying and must take some other action (determined by the application). For CAIN, the call takes the error action defined for the trigger which led to the query attempt. When a query is blocked, a Variable AIN Messaging

Platform (VAMP) 301 log is generated, indicating the control that blocked the query. After the gap interval timer expires, the next query attempt with matching GTA and TT is not blocked and is processed normally. The gap interval timer is then reset (and randomized) for another blocking period. This process continues until the control is removed or the gap duration timer expires.

If the switch receives an ACG message containing a control that is already initiated (and the control was not initiated by an operator), the control list is updated with the new information for that control. When the list is updated, the new gap interval and gap duration are applied and the timers are restarted. If the switch receives an ACG message containing a control that is already initiated but was initiated manually with the ACGCNTRL CI command, the switch ignores the request to update that control.

A control removal message is received in an ACG message with a ***`GapInterval`*** value of `removeGapControl`.

An ACG message containing a TT which is not supported by CAIN is ignored. Currently CAIN_ADDR_GT, CAIN_CLID_GT, CAIN_FEAT_GT, and CAIN_OFCD_GT are supported by CAIN.

When a query is blocked by ACG, CAIN may attempt an overflow query to a new global title. CAIN first checks the TRIGGER table to see if the ACGOVFLGT option is set (the ACGOVFLGT option specifies a new global title). If the ACGOVFLGT option is not set, then CAIN checks the ACG_OVERFLOW_GT parameter in table CAINPARM for a new global title (if NIL is set, then there can be no requery). If the query is blocked by ACG again, or if no overflow global title is datafilled, the switch generates a CAIN200 log stating that the query was blocked by an ACG control and the call will take the error action defined for the trigger associated with the query. If the error action is TREAT, the treatment defined by the parameter ACG_TREATMENT in table CAINPARM is applied to the call. Use of an overflow global title is not attempted when an OIECREO or OFFCCODE query is blocked by an ACG control.

When an ACG control is encountered by a query message and call processing performs a requery using a new global title, the query message will include the `acgRequery` extension parameter. When present, the `acgRequery` extension parameter indicates to the SCP that the current query is a requery message resulting from the original query being blocked by an ACG control.

Control List Synchronization

The SCP keeps its own image of the switch's control list. In order to help keep the lists synchronized, the switch sends certain messages to let the SCP know the status of the switch's control list.

When an ACG control is encountered by a query message (that is, the query matches the GTA and TT of the control), and the gap interval has expired or the control has a Zero-Gap interval (SOCC controls only), the query message will include the ***ACGEncountered*** parameter. The ***ACGEncountered*** parameter contains the type of control (SCP Overload Control or SOCC) and the number of digits in the control.

Once a query encounters an ACG control, further controls are not checked for that query. In other words, a query can only encounter one control.

Manual queries (for example, queries sent by the CAINTEST CI tool) are not subject to ACG controls for gapping. However, the control list is still checked to see if the query would have encountered a control. If a control would have applied, the query message will include the ***ACGEncountered*** parameter. Any ACG messages that are received in response to a manual query will be processed as if they had been received in response to a query generated by call processing.

Outgoing queries that are not initial query messages are not subject to ACG controls for gapping. They will, however, check the control list to see if a control would have applied. If a control would have applied, the query message will contain the ***ACGEncountered*** parameter. Subsequent messages that are not query messages (that is, any messages for a call sent after conversation has been initiated, EDP Requests, EDP Notifications, Resource Clear, and so forth) will not check the control list to see if a control would apply, and will therefore not contain the ***ACGEncountered*** parameter.

When the SCP receives a query message with the ***ACGEncountered*** parameter, the SCP is expected to check its image of the switch control list for the corresponding control. If the control does not exist in its image the SCP should include an ACG component in the response message to remove the control at the SSP.

Control Precedence

A precedence must be given for selecting controls from the control list for a query to match on. This is necessary because a query may match on more than one control in the control list. Table 2-3 provides a control list example.

Table 2-3
Control list example

GTA	TT	Control List	Gap Interval	Gap Duration
214684	cain_addr_gt	SOCC	no0Seconds	no32Seconds
214684	cain_addr_gt	SOCC	no30Seconds	no128Seconds
214684	cain_addr_gt	SCP	no4Seconds	infinity
214	cain_addr_gt	SOCC	no1Seconds	no32Seconds

A query containing a GTA of 2146841234 and TT of CAIN_ADDR_GT could match on all of the control entries. In order to determine which control takes precedence, priorities are given to the different types of controls when searching the control list for a match.

All Zero-Gap controls on the SOCC control list are checked first. If a match is not detected, the search continues, proceeding from more-specific to less-specific number of digits. If the number of digits is the same between the SOCC and SCP lists, the SOCC list is given precedence.

In the above example the controls are listed in order of precedence, therefore the Zero-Gap SOCC control would be encountered.

ACG Message Validation

When an ACG message is received from the SCP, the switch performs certain internal data consistency checks. When a violation is detected, the ACG request is discarded and a CAIN101 log is generated stating that CAIN was unable to decode a mandatory parameter. If a violation is detected in a response or conversation message, the violation is reported back to the SCP. If a violation is detected in a unidirectional message, it is not reported back to the SCP. The following validations are made on ACG messages:

- If the ***ControlCauseIndicator*** parameter indicates the control is an SCP Overload Control, then the gap must be marked as a Private Gap Interval.
- If the ***ControlCauseIndicator*** parameter indicates the control is a SOCC, then the gap must be marked as a National Gap Interval.
- The TT must be datafilled in table C7GTTYPE.
- The TT must be one of the global title types supported by CAIN (currently CAIN_ADDR_GT, CAIN_CLID_GT, CAIN_FEAT_GT, and CAIN_OFCD_GT).

Note: The check for a CAIN supported global title type is not performed when a control is added using the ACGCNTRL CI command.

ACG_Global_Control_Restore message

The switch can remove controls individually or all at once from the ACG control lists. Removal of all the controls at once is done with an ACG Global Restore Request. The request can come from either the SCP (in a **`ACG_Global_Ctrl_Restore`** message), or from an operator using a CI command.

While a control list is being reset, it is placed in an inactive state (no new ACG messages are processed), until the deletions are completed.

When the ACG Global Restore Request has been successfully completed, an **`ACG_Global_Ctrl_Restore_Success`** message is sent to the SCP and a VAMP303 log is generated. An **`ACG_Global_Ctrl_Restore_Success`** message is not sent when the list is reset with the ACGCONTRL CI command.

The **`ACG_Global_Ctrl_Restore`** request is a non-call related query message sent by the SCP in an invoke-last component.

Note: All controls are removed according to this message regardless of the SCP that initiated the control (including an LNP SCP).

Control List Overflow

It is possible for a control list to have no available spaces when a new **`ACG`** message is received by the switch. When this happens, the control contained within the **`ACG`** message is discarded, and a unidirectional **`ACG_Overflow`** message is sent to the SCP. The **`ACG_Overflow`** message indicates whether the control that overflowed was an SCP Overload Control or a SOCC.

If the response message (that contained the discarded control) contains a **`Send_Notification`** request, then a **`Termination_Notification`** is sent in addition to the **`ACG_Overflow`** message.

Note: Neither the **`ACG_Overflow`** message nor the `acgRequery` extension parameter are supported by CAIN LNP.

Global Outgoing Control

Global Outgoing Control (GOC) provides a method of gapping outgoing requests to all SCP destinations. This control can only be initiated through the ACGCNTRL CI command. For more information on the ACGCNTRL CI command, refer to the *UCS DMS-250 Commands Reference Manual*.

The GOC must provide a global gap interval value, and information on the interaction between the GOC and individual ACG controls. Table 2-4 contains the possible global gap interval values.

Table 2-4
GOC Gap Intervals

Time Range (seconds)
0 –900
infinity

The interaction between the GOC and the individual ACG controls can either be OVERRIDE or PREPROCESS. The interaction is specified through the ACGCNTRL CI command. When the interaction is OVERRIDE, the GOC overrides all individual ACG controls, including Zero-Gap controls.

When the interaction is PREPROCESS, individual ACG controls (including Zero-Gap controls) will be processed and applied first. However, if no individual control applies then the GOC will be applied to the message.

Note: The ACGCNTRL command can have a drastic effect on the number of queries sent to all SCPs, and should only be used when necessary.

IN/1 ACG

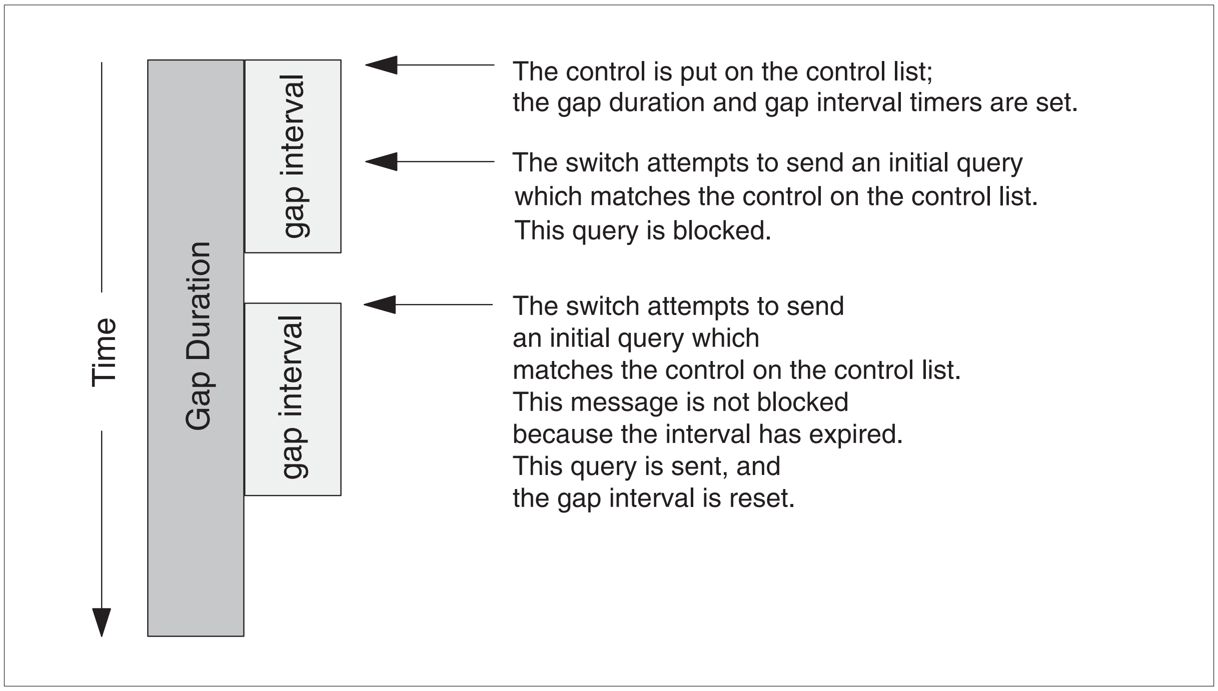
Note: The following information applies only to TR-NWT-000533 IN/1 ACG.

The switch initiates a TR-NWT-000533 IN/1 ACG control when it receives a **`Connect`** or **`Play_Announcement`** message with an **`ACG`** component. If the call cannot be completed because an error occurs while processing either the **`Connect`** or **`Play_Announcement`** message, then the **`ACG`** component is not processed.

The SCP uses ACG to reduce the number of TCAP queries involving a specific control number (identified by the **`ACG`** component ***`Digits`*** parameter). This control number must be the dialed N00 number used in the TCAP query in order for the ACG control to be properly administered on subsequent calls.

The SCP specifies a duration of time (identified by the **`ACG`** duration indicator), during which the switch can send a limited number of queries to the SCP for a specific NYY control number. This duration of time is referred to as the gap duration. The SCP also specifies a gap interval (identified by the `ACG` gap indicator). The gap interval specifies the length of time that the switch cannot sends queries to the SCP. Refer to Figure 2-2 for an illustration of the interaction between the control gap duration and gap interval.

Figure 2-2
Gap duration and gap interval interaction


Time
Gap Duration
gap interval
gap interval
The control is put on the control list; the gap duration and gap interval timers are set.
The switch attempts to send an initial query which matches the control on the control list. This query is blocked.
The switch attempts to send an initial query which matches the control on the control list. This message is not blocked because the interval has expired. This query is sent, and the gap interval is reset.

For the duration of an ACG control, the actual length of time that elapses before the switch can send a query is specified by the gap value. The gap value is a randomized length of time, between 90 and 110% of the gap interval for the control. The gap value is randomized each time the gap interval is reset.

The number of ACG controls which are available is based on the provisioning of ACG resources in table VAMPTRID. More than one ACG control may apply to a particular call (that is, there can be both a 6 and 10-digit control that applies to one call). The control on the longer code should be applied first. If the call successfully passes this first control, then the gap timer of the first control should be reset. The call should then be subjected to the control on the shorter code.

If a TCAP query cannot be sent due to an existing ACG control, the call is terminated to the ACG treatment (specified by the ACG_TREATMENT parameter in table CAINPARM).

Following the receipt and processing of an **ACG** component, the call which received ACG controls is routed according to the other received components. If a response message requests an ACG control for a code which is already under ACG control, the new control should replace the existing control, regardless of the cause. If there is no **ACG** component in the response message associated with a call whose 6 or 10-digit code is already under control, the control is removed. If both a 6-digit and a 10-digit control exist on the number, and there is no **ACG** component in the response message, both controls should be removed. If both a 6-digit and 10-digit control exist on the number, and a control of one code-length is returned in an **ACG** component, then the control of the second code-length is removed.

TR-533 ACG Cause to Treatment Mapping

This feature allows receiving a spare control cause value in an IN/1 ACG message. This feature also allows mapping control cause values to a defined treatment.

When an ACG control blocks a call, the Control Cause Indicator (CCI) is mapped to the treatment value stored in the ACG_TREATMENT tuple of table CAINPARM. If ACG_TREATMENT is UNDT (undefined treatment), then AINF is applied. The treatment is used to fill the CDR TRTMT field.

The TR-533 ACG Cause to Treatment Mapping feature extends this functionality by providing a mechanism for specifying different treatments for each CCI value. Table ACGMAP allows provisioning the hexadecimal value of the CCI against a particular treamtment. This enables the provisioning of spare values, #06 to #FF, as well as the TR-NWT-000533 defined CCI values.

The TR-NWT-000533 values for ACG CCI are as follows:

- 0000 0000 – not used
- 0000 0001 – vacant code
- 0000 0010 – out of band
- 0000 0011 – database overload
- 0000 0100 – destination mass calling
- 0000 0101 – SMS initiated
- 0000 0110 – Spare
- ... – Spare
- 1111 1111 – Spare

Restrictions and limitations

The ACG control list is not maintained over an ONP, SWACT, or restarts.

VAMP ACG only applies to applications built on VAMP.

Event processing

ATTENTION
Extension parameters and certain EDPs require various SOC options in order to function. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information.

The switch provides the ability for normal call processing to be interrupted at various points in call (PIC) according to the Bellcore AIN 0.2 Call Model. At each of these PICs, detection points (DPs) may be reached. Two types of detection points may be encountered, trigger detection points (TDPs) or event detection points (EDPs).

TDPs are points in call processing when a trigger is examined. If the conditions associated with that trigger are satisfied, then the action indicated by the trigger is taken. When the switch encounters a PIC with a TDP, it consults in-switch logic and datafill to determine if normal in-switch call processing should be interrupted. When the in-switch call flow is interrupted, the switch may send a TCAP package (message) to the SCP requesting instructions for processing the current call.

For EDPs to be encountered, an SCP response may include a list of EDPs to activate (“arm”) along with a call-related **`Analyze_Route`**, **`Continue`**, or **`Collect_Information`** message. After the **`Analyze_Route`**, **`Continue`**, or **`Collect_Information`** is processed, the call follows its standard logic, ignoring all TDPs and triggers, until a newly armed EDP is encountered or EDPs are deactivated. With EDP support, the transaction remains open, allowing the SCP to retain call information rather than having the switch resend the call information every time a query (TDP-Request) is sent to the SCP for the call.

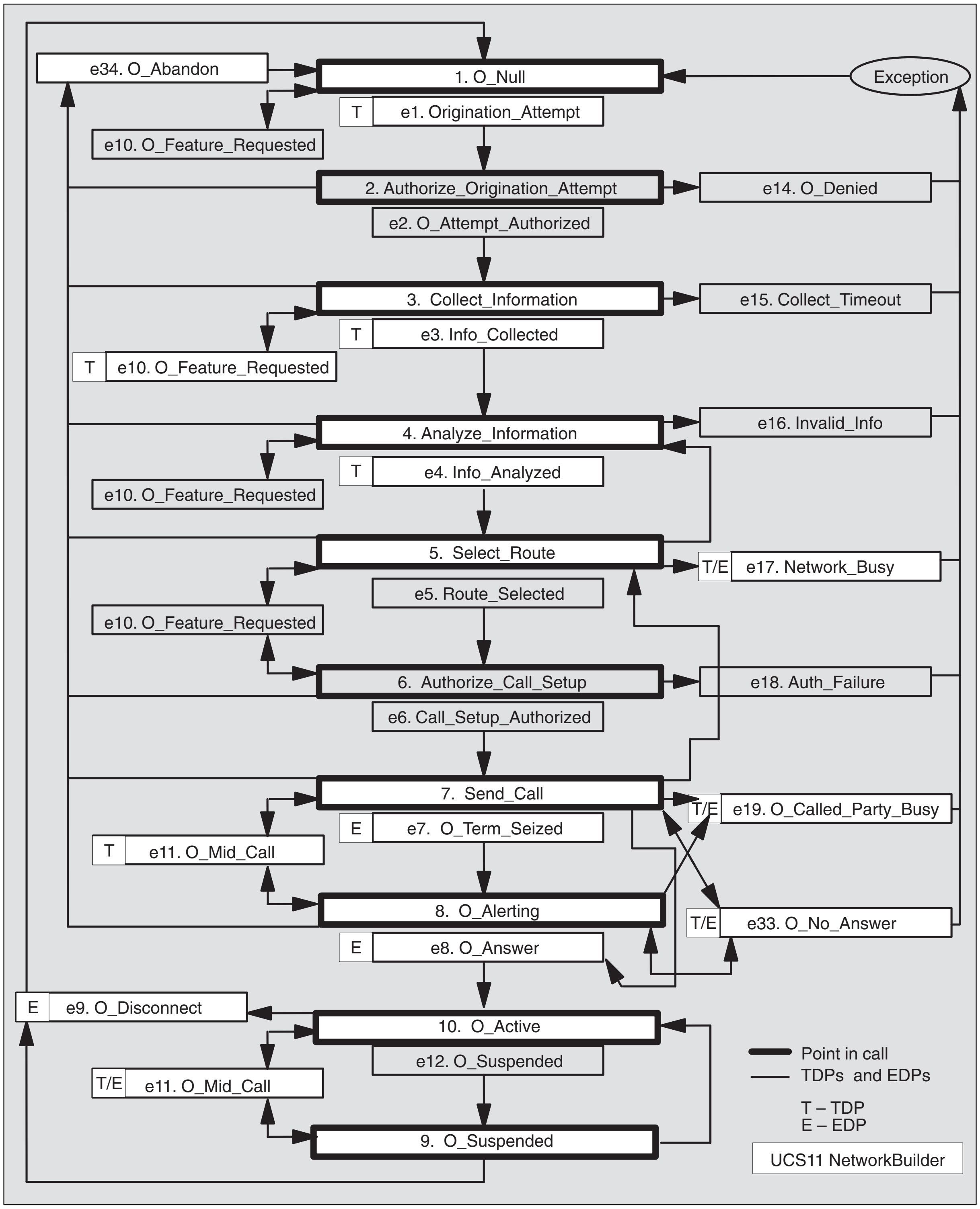
NetworkBuilder supports the following EDPs:

- ***Network_Busy***
- ***O_Abandon***
- ***O_Answer***

- ***O_Called_Party_Busy***
- ***O_Disconnect***
- ***O_Mid_Call***
- ***O_No_Answer***
- ***O_Term_Seized***

Figure 3-1 shows where these EDPs fit into the Carrier-AIN 0.2 originating call model.

Figure 3-1
NetworkBuilder originating call model



In order to arm EDPs, the SCP sends the switch a call-related component (**`Analyze_Route`**, **`Continue`**, or **`Collect_Information`**) in a conversational TCAP package along with a **`Request_Report_BCM_Event`** non-call related component. The presence of the **`Request_Report_BCM_Event`** component causes the switch to associate a next event list (NEL) with the open transaction. The NEL is a list of EDPs which are armed to send an EDP-Request or EDP-Notification to the SCP (or possibly perform another action) when reached. EDP-Notification messages are sent in TCAP format to the SCP without interrupting the call flow or waiting for a response. The SCP may respond to an EDP-Request with the same messages it can use to respond to a TDP-Request (query) message of the same type. Once an EDP-Request is sent or all active EDPs in the call model have been passed, EDPs are deactivated and TDPs may be encountered.

Note: For information on provisioning the SCP simulator for EDPs, refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information.

Limitations and restrictions

Although *Office_Code* triggering is allowed while EDPs are active, EDPs may not be armed in response to an LNP query. If the response to an LNP query includes a **`Request_Report_BCM_Event`** component, the switch generates a CAIN100 log, sends a **`Close`** message to the SCP with the ***`CloseCause`*** parameter value of `unexpectedCommunication`, and processes the call-related component normally. The **`Close`** message is sent in the same transaction as the LNP query. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on LNP call processing.

Reevaluation of the *Office_Code* trigger at the ***Info_Analyzed*** TDP is allowed while an EDP is active. Reevaluation of all other triggers at the ***Info_Analyzed*** TDP is not allowed while EDPs are active. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on LNP call processing.

EDP call processing

Under normal circumstances, EDPs are armed when a **`Request_Report_BCM_Event`** component is received with a call-related component in a multiple-component conversation package in response to a TDP- or EDP-Request message.

The following conditions prevent EDPs from being armed when a **`Request_Report_BCM_Event`** component is received:

- A fatal application error occurs while processing the call-related component. Whenever a fatal application error occurs, if the current trigger has a datafilled error action in the trigger table, that action is taken. If the current detection point (DP) is an EDP or no error action is included in the trigger table, the call receives AINF treatment and is disconnected.
- The **`Request_Report_BCM_Event`** component is received as the first component in the package, or the first component is not call-related. This is a fatal application error.
- The events requested are not permitted by the CAIN SOC options. CAIN SOC option (CAIN0602) controls the *Network_Busy*, *O_Term_Seized*, *O_Called_Party_Busy*, *O_Answer*, and *O_No_Answer* events. CAIN SOC option (CAIN0800) controls the *Timeout* and *O_Disconnect* events. If either SOC is idle when the switch receives a **`Request_Report_BCM_Event`** component requesting an event controlled by the SOC, a non-fatal application error occurs, and the switch generates a CAIN102 log. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information on SOC.

  Note: If the **`Request_Report_BCM_Event`** component received does not request any events which are permitted by SOC, the switch sends a **`Close`** message to the SCP with a ***`CloseCause`*** parameter value of `unexpectedCommunication`. The call-related component is processed normally, regardless of SOC options CAIN0602 and CAIN0800.

- The **`Request_Report_BCM_Event`** component is received along with an **`Analyze_Route`** component containing the `classOfSvc` extension parameter and the class-of-service screening fails. A CAIN100 log is generated and a **`Close`** message is sent to the SCP with ***`CloseCause`*** value `eDPsCompleted`. The call is routed according to the COSUS index which failed screening. Refer to Chapter 12, “Incoming CAIN message parameters,” for more information on the `classOfSvc` extension parameter.

EDP call processing (continued)

- A fatal application error occurs while processing the **`Request_Report_BCM_Event`** component. Refer to “Fatal application errors,” page 3-45 for more information.
- The package was received in response to an **`O_No_Answer`** TDP-Request or EDP-Request, but the called party answered before the package was received. A **`Close`** message is sent to the SCP with ***`CloseCause`*** parameter value `calledPartyAnswered`. Neither component is processed.
- The **`Request_Report_BCM_Event`** component is received in response to an LNP query. A CAIN100 log is generated and a **`Close`** message is sent to the SCP with a ***`CloseCause`*** value of `unexpectedCommunication`.

If none of the above conditions apply, the switch processes the call-related component and arms the specified EDPs. The conversation transaction is kept open. Call flow continues normally until one of these EDPs is reached or all active EDPs are passed over. With the exception of the ***Info_Analyzed*** TDP being encountered for *Office_Code* triggering, no TDPs in the originating call model are encountered while EDPs are active, and any remaining triggers at the TDP at which EDPs are activated are passed over. TDPs in the terminating call model can be encountered while EDPs are active in the originating call model.

Through the ***`EDPRequest`*** and ***`EDPNotification`*** parameters in the **`Request_Report_BCM_Event`** component, the SCP requests the switch to arm EDPs. If an EDP-Notification is sent, no response is expected and the call flow continues normally. If an EDP-Request message is sent, the NEL is discarded and the conversation transaction is maintained. However, if an EDP-Request received by the SCP and the EDP is not armed, a CAIN100 log is generated for the ***O_Abandon***, ***O_Disconnect***, ***switchHookFlash*** and ***Timeout*** EDPs.

EDP-Request messages count against the limit for the number of request messages that can be sent during a call, set by the MAX_NUM_SERIAL_TRIGGERS tuple in table CAINPARM. If this limit is reached and an EDP-Request action is indicated, the IGNORE action is taken instead.

Through extension parameters in the **`Request_Report_BCM_Event`** component or table CAINXDFT, the ***Network_Busy***, ***O_Called_Party_Busy***, and ***O_No_Answer*** EDPs may be armed to send an EDP-Request, to route advance, or to do nothing, depending upon routing criteria. The NEL is maintained when any action other than the sending of an EDP-Request is taken.

EDP call processing (continued)

The switch deactivates all EDPs when it determines that all active EDPs in the call model have been passed or cannot be reached. When this occurs, the switch closes the conversation transaction by sending a **`Close`** message to the SCP. Optional parameters ***`UserID`*** and ***`BearerCapability`*** are included in this message, as well as ***`CloseCause`*** with a value of `eDPsCompleted` (if in-switch logic determines that no active EDPs are reachable) or `callTerminated` (if the call is taken down or sent to treatment). After EDPs have been deactivated, the switch may again encounter TDPs in the originating call model.

Note: When an EDP-Notification is sent to the SCP in a conversation package and no more active EDPs are reachable, the switch immediately sends a response package containing a **`Close`** message.

It is possible to send an LNP query while EDPs are active. When this occurs, the EDPs which are active before sending the LNP query remain active when the response to the query is received. The LNP query and response have no effect on the call's next event list (NEL).

For the Take-back and Transfer feature, to control the call the SCP must be notified of an event involving a party disconnecting from the call. During CC4-CC11, inclusive, an open transaction must be maintained with the SCP. If an open transaction is not maintained, the switch detects an error and provides final treatment procedures. The switch also generates a CAIN200 log with a description field of "Transaction closed in inappropriate CC." For further information, refer to the "Quick-reference to handling of unarmed events" section in Chapter 4.

Queuing switch events for Take-back and Transfer functionality

The switch queues AIN events when any of the following conditions are true:

- another AIN event for the CC is pending in the queue
- Timer T1 is set, indicating that a response from the SCP is pending
- the switch is processing an SCP response

The switch processes a queued AIN event associated with a CC when both of the following conditions are met:

- the switch has completed processing an SCP response
- T1 timer is not set (the switch is not awaiting a response)

When the SCP receives a message resulting from an event that was queued, it must determine whether it is appropriate in the context of the CC resultant of the previously processed message. Such a situation is possible only in call

EDP call processing (continued)

configurations with two call segments (that is, CC4, CC5, CC6, CC7, CC10, and CC11). Currently, the AIN events that can be queued are *O_Abandon*, *O_Disconnect*, *Failure_Outcome*, and *Switch_Hook_Flash*.

Note: If a party goes on-hook during a **`Connect_To_Resource`** interaction, the switch immediately disconnects that party's leg. At the end of the **`Connect_To_Resource`** interaction, the switch sends a **`CTR_Clear`** message with a ***`ClearCause`*** of `userAbandon`. This is the only instance in which the switch processes a subsequent event before completely processing a previous event.

Unreachable EDPs

The switch uses the following rules to determine that EDPs are unreachable:

- When the ***O_Term_Seized*** detection point is passed, all EDPs are unreachable unless the ***Network_Busy***, ***O_Answer***, ***O_Called_Party_Busy***, ***O_Disconnect***, ***O_Mid_Call*** for either the *Switch_Hook_Flash* or the *Timeout* event, or ***O_No_Answer*** EDP is armed.
- When the ***O_Answer*** detection point is passed, all EDPs except ***O_Mid_Call*** for either the *Switch_Hook_Flash* or the *Timeout* event, and ***O_Disconnect*** are unreachable.
- When the call is sent to treatment or taken down before reaching the **O_Active** PIC, all EDPs are unreachable.
- When the ***O_Disconnect*** detection point is passed, all EDPs are unreachable.
- When the ***O_Active*** detection point is passed, the ***O_Abandon*** EDP is unreachable.
- The *Switch_Hook_Flash* event is unreachable unless the call is in CC2, CC4, CC6, or CC10.

  Note: If an unsupported EDP is armed, the next event list (NEL) remains active until an EDP-Request is sent or until the switch uses one of the above rules to determine that all EDPs are unreachable.

Feature interactions

Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information on EDP interactions with the Long Call Duration timer for AXXESS calls.

All active EDPs on all calls are deactivated when a norestart switch active (SWACT) is performed. The SCP is not notified.

EDP call processing (continued)

Although there are no EDPs currently supported in the terminating call model, the *Termination_Attempt* trigger may be evaluated while EDPs are active in the originating call model.

Since reorigination ends the call and starts a new one, the switch sends a **`Close`** message to the SCP with a ***`CloseCause`*** parameter value of `callTerminated` if the EDPs are active when the switch receives reorigination indication. If the **`O_Disconnect`** event is active when the switch receives reorigination indication, the switch sends an **`O_Disconnect`** EDP-Notification prior to the **`Close`** message.

If reorigination occurs while an **`Info_Analyzed`**, **`Network_Busy`**, **`O_Called_Party_Busy`**, or **`O_No_Answer`** query message is being processed at the SCP, the switch closes the transaction associated with the query and T1 timing for the query continues. Under these conditions, when a **`Request_Report_BCM_Event`** component is received in a conversation package, a **`Close`** message in a response package with a ***`CloseCause`*** value of `callTerminated` is sent.

EDPs can never be encountered in an SS7 RLT Third-party call scenario. In an SS7 RLT Redirect call scenario, EDPs may be encountered after the call redirects.

For any call employing the N00 take-back and transfer service (refer to "Chapter 4. Call Configurations" for description), the switch performs the following actions when a message is received from the SCP:

- clears the NEL
- cancels the T1 timer

Network_Busy EDP

This EDP occurs at the **Select_Route** PIC. The conditions that cause this detection point to be reached are described in detail in Volume 1, Chapter 6, “Select_Route PIC.” If EDPs are active when the ***Network_Busy*** detection point is reached, the ***Network_Busy*** EDP is encountered. If there is no active NEL when this detection point is reached, the ***Network_Busy*** TDP is encountered.

When the ***Network_Busy*** EDP is encountered and found to be active, there are several actions the switch may take. If no `networkBusyActions` extension parameter was received with the **`Request_Report_BCM_Event`** component and there is no `networkBusyActions` extension parameter datafilled in CAINXDFT for the active CAIN group, a **`Network_Busy`** EDP-Request message is sent to the SCP. If an extension parameter value is used, the action to be taken depends upon whether or not this detection point was reached due to a busy direct-termination route and whether or not there are additional routing choices available. The routing conditions that can apply at ***Network_Busy*** are RTEAVAIL, RTESDONE, and TERMRTE_GNCT, discussed in Volume 1, Chapter 6, “Select_Route PIC.” The appropriate field in the extension parameter determines what action to take.

When a **`Network_Busy`** EDP-Request message is sent to the SCP, the NEL is cleared and the conversation transaction is maintained. From this point forward, call processing proceeds as if a **`Network_Busy`** TDP-Request message were sent.

After the NEXTRTE or NEXTCNRTE action is taken, all active EDPs remain active, and the conversation transaction remains open. It is possible that the ***Network_Busy*** EDP will be evaluated again during this call. If the call reaches a detection point at which it can be determined that the ***Network_Busy*** EDP and all other active EDPs are unreachable, the switch clears the NEL at that point and sends a **`Close`** message to close the conversation transaction.

If an IGNORE action is indicated, call processing proceeds as if the ***Network_Busy*** detection point were not reached. The call is sent to treatment because of the busy condition. The UCS DMS-250 switch clears the NEL and sends a **`Close`** message to close the conversation transaction.

O_Abandon EDP

The ***O_Abandon*** EDP can occur at any of the following PICs: **Collect_Information**, **Analyze_Information**, **Select_Route**, **Send_Call**, and **O_Alerting**. This detection point may only be encountered as an EDP, during CC4 or CC6. Call processing encounters this EDP when the controlling leg goes on-hook before the called party has answered. If the ***O_Abandon*** EDP is active when call processing reaches this detection point, the switch sends an **`O_Abandon`** EDP-Request to the SCP. The messages supported in response to ***O_Abandon*** are **`O_Disconnect`**, **`Disconnect_Leg,`** **`Merge_Call`**.

For information on arming restrictions for the ***O_Abandon*** EDP, refer to "Unreachable EDPs" on page 3-8, and "Event arming restrictions" on page 3-45.

Refer to Chapter 10, "Incoming CAIN messages," for specific information regarding response messages, and Chapter 12, "Incoming CAIN message parameters," for detailed descriptions of the response parameters.

EDP call processing
O_Answer EDP (end)

O_Answer EDP

The ***O_Answer*** EDP occurs at the **O_Alerting** PIC. This detection point may only be encountered as an EDP. It is encountered when the terminating party answers the call. If the ***O_Answer*** EDP is active when this detection point is reached, an **`O_Answer`** EDP-Notification is sent to the SCP and call processing continues normally. The O_No_Answer timer, if running, is canceled. Also, if the *Timeout* event is requested, the switch starts the TimeoutTimer when the call is answered.

If neither the ***O_Mid_Call*** nor ***O_Disconnect*** EDPs are active when **`O_Answer`** is passed, the switch clears the NEL and sends a **`Close`** message to the SCP after processing this EDP.

O_Called_Party_Busy EDP

The ***O_Called_Party_Busy*** EDP occurs at the **Send_Call** PIC. The conditions that cause this detection point to be reached are described in Volume 1, Chapter 7, "Send_Call PIC." If EDPs are active when the ***O_Called_Party_Busy*** detection point is reached, then it is encountered as an EDP. If there is no active NEL when this detection point is reached, the ***O_Called_Party_Busy*** TDP is encountered.

When this EDP is encountered and found to be active, there are several actions the switch may take. If no `oCalledPartyBusyActions` extension parameter was received with the **`Request_Report_BCM_Event`** component and there is no `oCalledPartyBusyActions` extension parameter datafilled in CAINXDFT for the active CAIN group, an **`O_Called_Party_Busy`** EDP-Request message is sent to the SCP. If an extension parameter value is used, the action to be taken depends upon whether or not there are additional routing choices available. The appropriate value in the extension parameter determines what action to take.

When an **`O_Called_Party_Busy`** EDP-Request message is sent to the SCP, the NEL is cleared and the conversation transaction is maintained. From this point forward, call processing proceeds as if an **`O_Called_Party_Busy`** TDP-Request message were sent.

After the NEXTRTE or NEXTCNRTE action is taken, all active EDPs remain active, and the conversation transaction remains open. It is possible that the ***O_Called_Party_Busy*** EDP will be evaluated again during this call. If the call reaches a detection point at which it can be determined that the ***O_Called_Party_Busy*** EDP and all other active EDPs are unreachable, the switch clears the NEL at that point and sends a **`Close`** message to close the conversation transaction.

If an IGNORE action is indicated, call processing proceeds as if the ***O_Called_Party_Busy*** detection point were not reached. The call is sent to treatment because of the busy condition. The UCS DMS-250 switch clears the next event list (NEL) and sends a **`Close`** message to close the conversation transaction.

O_Disconnect EDP

The ***O_Disconnect*** EDP occurs at the **O_Active** and **O_Suspended** PICs. This detection point can only be encountered as an EDP. The ***O_Disconnect*** EDP can be armed to respond to the *O_Disconnect* event with either an EDP-Notification message or an EDP-Request message.

Call processing detects the *O_Disconnect* event when a party disconnects. It can be detected only after the call has been answered and the call has entered the **O_Active** or **O_Suspended** PICs.

Note: Call processing does not detect the *O_Disconnect* event in SS7 RLT scenarios when an operator disconnects from the calling party to route the calling party to another party.

When reorigination indication is received, call processing encounters the ***O_Disconnect*** EDP before the ***O_Mid_Call*** TDP. Also, call processing encounters the ***O_Disconnect*** EDP when the switch takes down a call in the **O_Active** or **O_Suspended** PICs.

O_Disconnect EDP-Notification

The switch arms the ***O_Disconnect*** EDP to send an **`O_Disconnect`** EDP-Notification message when the switch receives a valid **`Request_Report_BCM_Event`** component containing an ***`EDPNotification`*** parameter with the value `oDisconnect`.

If the switch arms the ***O_Disconnect*** EDP to send an **`O_Disconnect`** EDP-Notification message, and call processing detects the *O_Disconnect* event, the following events occur:

- The switch sends an **`O_Disconnect`** EDP-Notification to the SCP.
- If the *Timeout* event is active, the switch cancels the Timeout timer.
- The switch sends a **`Close`** message to the SCP with a ***`CloseCause`*** value of `callTerminated`. The switch expects no response from the SCP.
- When call processing detects the *O_Disconnect* event due to the switch receiving reorigination indication during a two-party call, call processing encounters the ***O_Mid_Call*** TDP immediately after the switch sends the **`Close`** message to the SCP.

O_Disconnect EDP-Request

If the switch arms the ***O_Disconnect*** EDP to send an **`O_Disconnect`** EDP-Request message, the switch builds and sends the **`O_Disconnect`** EDP-Request message when the *O_Disconnect* event is detected.

EDP call processing
O_Disconnect EDP (end)

The switch only arms the **`O_Disconnect`** EDP as an EDP-Request in multi-party call scenarios. Refer to Volume 3, Chapter 4, “Call configuration model,” for more information on multi-party calls.

The following messages are supported in response to the **`O_Disconnect`** EDP-Request message:

- **`Connect_To_Resource`** message
- **`Disconnect`** message
- **`Disconnect_Leg`** message
- **`Merge_Call`** message

Refer to Chapter 10, “Incoming CAIN messages,” for specific information regarding response messages, and Chapter 12, “Incoming CAIN message parameters,” for detailed descriptions of the response parameters.

O_Mid_Call EDP

The ***O_Mid_Call*** EDP occurs at the **O_Active** and **O_Suspended** PICs. The switch can arm the ***O_Mid_Call*** EDP to detect either the *Timeout* event or the *Switch_Hook_Flash* event. If the SCP requests that the switch arm the ***O_Mid_Call*** EDP to detect both the *Timeout* and *Switch_Hook_Flash* events, the switch arms the *Timeout* event, and ignores the request for the *Switch_Hook_Flash* event.

Timeout event

The switch arms the *Timeout* event when the switch receives a valid **`Request_Report_BCM_Event`** component containing an ***`EDPRequest`*** parameter with the value `Timeout`. The switch can arm the *Timeout* event when a call is in CC1 or CC2. If the call is in a different CC, the switch ignores the request to arm the *Timeout* event and still arms the other requested events.

Call processing detects the *Timeout* event when the TimeoutTimer expires. This timer specifies how many minutes a call can be active before encountering the *Timeout* event.

If the switch has armed the *Timeout* event, the UCS DMS-250 switch starts the TimeoutTimer when the called party goes off-hook. The length of the timer is determined by one of the following values:

- the value in the ***`TimeoutTimer`*** parameter received in the **`Request_Report_BCM_Event`** component that armed the EDP
- the value of the TIMEOUT_TIMER parameter in table CAINPARM if the switch did not receive a valid ***`TimeoutTimer`*** parameter

The TimeoutTimer can be restarted if the switch rearms the *Timeout* event, after the switch sends a **`CTR_Clear`** message at the ***O_Mid_Call*** EDP. The timer's duration is specified by the ***`TimeoutTimer`*** parameter received in the second **`Request_Report_BCM_Event`** component, or from the value of the TIMEOUT_TIMER parameter in table CAINPARM.

When the TimeoutTimer expires, the switch sends a **`Timeout`** EDP-Request message to the SCP in a conversation package. The **`Disconnect`** and **`Connect_To_Resource`** messages are supported in response to a **`Timeout`** EDP-Request. When the switch receives the **`Disconnect`** message, it is processed normally, and the switch releases the called party as if the originator had disconnected. When the switch receives a response other than the **`Disconnect`** or **`Connect_To_Resource`** message, a fatal application error occurs; the calling party is sent to AIN Final (AINF) treatment, and the called party is released.

Note: The CAIN0801 SOC must be enabled in order to allow a **`Connect_To_Resource`** response to the **`Timeout`** EDP-Request.

For more information on the **`Connect_To_Resource`** message, refer to Chapter 10, “Incoming CAIN messages,” or Volume 4, Chapter 3, “CTR-Connections.”

Note: When the switch sends a **`CTR_Clear`** message while the call is at the *Timeout* event, the **`Disconnect`**, **`Continue`**, and **`Connect_To_Resource`** response messages are supported. Although the CAIN protocol allows the **`Analyze_Route`** and **`Collect_Information`** messages in response to **`CTR_Clear`**, a fatal application error occurs when one of these messages is received at the *Timeout* event.

Refer to Chapter 10, “Incoming CAIN messages,” for specific information regarding response messages, and refer to Chapter 12, “Incoming CAIN message parameters,” for detailed descriptions of the response parameters.

Switch_Hook_Flash event

The switch arms the *Switch_Hook_Flash* event when a valid **`Request_Report_BCM_Event`** component is received containing an ***`EDPRequest`*** parameter with the value `switchHookFlash`. The switch can arm the *Switch_Hook_Flash* event when a call is in CC1, CC2, CC4, CC5, CC6, CC7, or CC10. If the call is in a different CC, the switch ignores the request to arm the *Switch_Hook_Flash* event and still arms the other requested events.

Call processing detects the *Switch_Hook_Flash* event when one of the following actions occur:

- the passive leg (***`LegID`*** 1) presses the asterisk “*” key for 40 milliseconds during call configuration (CC) 2 (stable two-party call)
- the controlling leg (***`LegID`*** 0) presses the asterisk “*” key for 40 milliseconds during CC4 (three-party call in the setup phase), CC6 (party on hold), or CC10 (stable multi-party call)

The `shfelegs` extension parameter indicates which call leg or legs the switch monitors for a switch hook flash. Refer to Chapter 12, “Incoming CAIN message parameters,” for more information on the `shfelegs` extension parameter.

EDP call processing
O_Mid_Call EDP (end)

When the *Switch_Hook_Flash* event is detected, the switch sends an ***O_Mid_Call*** EDP-Request message. The following messages are supported in response to an ***O_Mid_Call*** EDP-Request message for the *Switch_Hook_Flash* event:

- **`Connect_To_Resource`** message
- **`Disconnect`** message
- **`Disconnect_Leg`** message
- **`Merge_Call`** message
- **`Originate_Call`** message

Refer to Chapter 10, "Incoming CAIN messages," for specific information regarding response messages, and refer to Chapter 12, "Incoming CAIN message parameters," for detailed descriptions of the response parameters.

If the *Switch_Hook_Flash* event is requested and detected after the MAX_NUM_SERIAL_TRIGGERS setting has been exceeded, the UCS DMS-250 switch takes the following actions:

- ignores the detection of the event
- generates a CAIN905 log for the detection
- generates a CAIN302 log, indicating that the call has exceeded the MAX_NUM_SERIAL_TRIGGERS setting
- generates a CAIN907 log, indicating that the switch has responded to the event as though it had not been requested

Note: In-switch reorigination and the *O_IEC_Reorigination* trigger are disallowed once the switch has armed the *Switch_Hook_Flash* event.

Limitations and restrictions

The detection of *Switch_Hook_Flash* event is not supported on PRI trunks due to hardware restrictions that prevent provisioning of a Specialized Tone Receiver (STR) card on the trunk.

For the **`Analyze_Route`** and **`Collect_Information`** messages, the request to arm the *Switch_Hook_Flash* event is ignored if the *Timeout* event is also requested.

O_No_Answer EDP

This EDP occurs at the **O_Alerting** PIC. It is encountered when the O_No_Answer timer expires. This timer specifies a time limit in which the called party should answer the call. The time limit is set and begins counting down at the ***O_Term_Seized*** detection point.

If EDPs are active when the ***O_No_Answer*** detection point is reached, the ***O_No_Answer*** EDP is encountered. If there is no active NEL when this detection point is reached, the ***O_No_Answer*** TDP is encountered.

When this EDP is encountered and found to be active, there are several actions the switch may take. If no `oNoAnswerActions` extension parameter was received with the **`Request_Report_BCM_Event`** component and there is no `oNoAnswerActions` extension parameter datafilled in CAINXDFT for the active CAIN group, an **`O_No_Answer`** EDP-Request message is sent to the SCP. If an extension parameter value is used, the action to be taken depends upon whether or not there are additional routing choices available. The appropriate value in the extension parameter determines what action to take.

After the NEXTRTE or NEXTCNRTE action is taken, all active EDPs remain active, and the conversation transaction remains open. It is possible that the ***O_No_Answer*** EDP will be evaluated again during this call. If the call reaches a detection point at which it can be determined that the ***O_No_Answer*** EDP and all other active EDPs are unreachable, the switch clears the NEL at that point and sends a **`Close`** message to close the conversation transaction.

If an IGNORE action is indicated, call processing proceeds as if the ***O_No_Answer*** detection point were not reached. The originator continues to hear ringing tone. All active EDPs remain active, and the conversation transaction stays open. If the call reaches a detection point at which it can be determined that the ***O_No_Answer*** EDP and all other active EDPs are unreachable, the switch clears the NEL and sends a **`Close`** message to close the conversation transaction.

EDP call processing
O_No_Answer EDP (end)

Note: Expiration of the Long Call Duration timer before the ***O_No_Answer*** EDP is reached does not directly cause the NEL to be discarded and the conversation transaction to be closed. Non–AXXESS calls utilizing EDPs, however are always sent to treatment when Long Call Duration timer expires. In this case, the switch clears the NEL and sends a **`Close`** message to close the conversation transaction. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for information on EDP interactions with the Long Call Duration timer for AXXESS calls.

O_Term_Seized EDP

The ***O_Term_Seized*** EDP occurs at the **Send_Call** PIC. This detection point may only be encountered as an EDP. It is encountered when a terminating trunk is seized on the UCS DMS-250 switch. The terminating trunk is considered seized when the UCS DMS-250 switch has:

- located an idle trunk member in the terminating trunk group.
- outpulsed the appropriate information on the terminating trunk.
- made a two-way network connection.
- supervised the the originating and terminating trunks.

According to Bellcore specifications, ***O_Term_Seized*** is detected when the terminating party is alerted of the call and audible ringing is provided to the calling party. Due to technical complexity, NetworkBuilder treats all terminators as conventional trunks in regards to the ***O_Term_Seized*** event. This means that no detection of alerting is performed before this event is encountered.

If the ***O_No_Answer*** EDP is armed when this detection point is reached, the O_No_Answer timer is started. The following precedence order is used to determine the time limit to be set:

1 Value of the ***`ONoAnswerTimer`*** optional parameter received in the **`Request_Report_BCM_Event`** component.

2 Value of the OANSTIME option in table CAINGRP for the last CAIN group that was used when sending a TDP-Request.

3 Value of the O_NO_ANSWER_TIMER field in table CAINPARM.

If the ***O_Term_Seized*** EDP is active when this detection point is reached, an **`O_Term_Seized`** EDP-Notification is sent to the SCP and call processing continues normally.

If neither the ***O_Called_Party_Busy***, ***Network_Busy***, ***O_Answer***, ***O_No_Answer***, ***O_Mid_Call***, nor ***O_Disconnect*** EDP is active when ***O_Term_Seized*** is passed, the switch clears the NEL and sends a **`Close`** message to the SCP after processing this EDP.

EDP messages

Message Protocols

There are two types of TCAP transactions allowed in the Bellcore AIN specifications: non-persistent and persistent transactions. A non-persistent transaction consists of a single query and response message pair. A persistent transaction includes conversation messages in addition to the query and response message pair. In AIN 0.2, there are two types of persistent (conversation) transactions: those involving a **`Send_To_Resource`** or **`Connect_To_Resource`** operation and those involving a NEL.

When the switch receives a **`Request_Report_BCM_Event`** component in a conversation package, the first component in the package must be one of the following call-related components: **`Analyze_Route`**, **`Continue`**, or **`Collect_Information`**. If a fatal application error occurs while processing either component, the contents of the entire package are discarded. If the package meets all necessary requirements, call processing continues in the same conversation transaction until an armed EDP is reached and an EDP-Notification or EDP-Request message is sent to the SCP.

The switch discards the NEL when one of the following events occur:

- An EDP-Request is sent to the SCP. The conversation transaction remains open. However, during Takeback and Transfer, the NEL is not cleared until a message is received from the SCP. At that time, the switch overwrites the NEL with the EDP arming request received in the message. If no arming requests are received, then a **`Close`** message is sent and the NEL is discarded.
- The switch sends a **`Close`** message in an Invoke component, in a response package to the SCP. When this occurs, the conversation transaction is closed.
- The switch receives a message from the SCP while the NEL is active. This is a fatal Unexpected Message application error. The conversation transaction is closed.

Table 3-1 describes the EDP and event messages.

Table 3-1
EDP and event messages

Message	Description
`Close`	The switch sends this message to terminate EDP processing. This message may also be sent by the SCP as an error message. The parameters used in the message vary depending on the function of the message. Refer to Chapter 10, "Incoming CAIN messages," for information on using the **`Close`** as an error message.
`Failure_Outcome`	The **`Failure_Outcome`** message informs the SCP that a failure event was detected in processing a **`Disconnect_Leg`**, **`Merge_Call`**, or **`Originate_Call`** message.
`Network_Busy`	EDP-Request message that may be sent to the SCP when the ***Network_Busy*** EDP is active and the ***Network_Busy*** EDP is reached. The parameters and extension parameters used in the message vary depending on the function of the message. Refer to Volume 1, Chapter 6, "Select_Route PIC," for information.
`O_Abandon`	The switch sends the **`O_Abandon`** EDP-Request message to the SCP to indicate that the controlling leg went on-hook before the called party answered.
`O_Answer`	EDP-Notification message sent to the SCP when the ***O_Answer*** EDP is active and the terminating party answers the call. Refer to Volume 1, Chapter 8, "O_Alerting PIC," for information.
`O_Called_Party_Busy`	EDP-Request message that may be sent to the SCP when the ***O_Called_Party_Busy*** EDP is active and the ***O_Called_Party_Busy*** EDP is reached. The parameters and extension parameters used in the message vary depending on the function of the message. Refer to Volume 1, Chapter 7, "Send_Call PIC," for information.
`O_Disconnect`	This message may be sent either as an EDP-Notification or an EDP-Request. The switch sends the **`O_Disconnect`** message to the SCP when a party disconnects after the call has been answered. Refer to Volume 1, Chapter 9, "O_Active and O_Suspended PICs," for information.

—continued—

EDP messages (continued)

Table 3-1
EDP and event messages (continued)

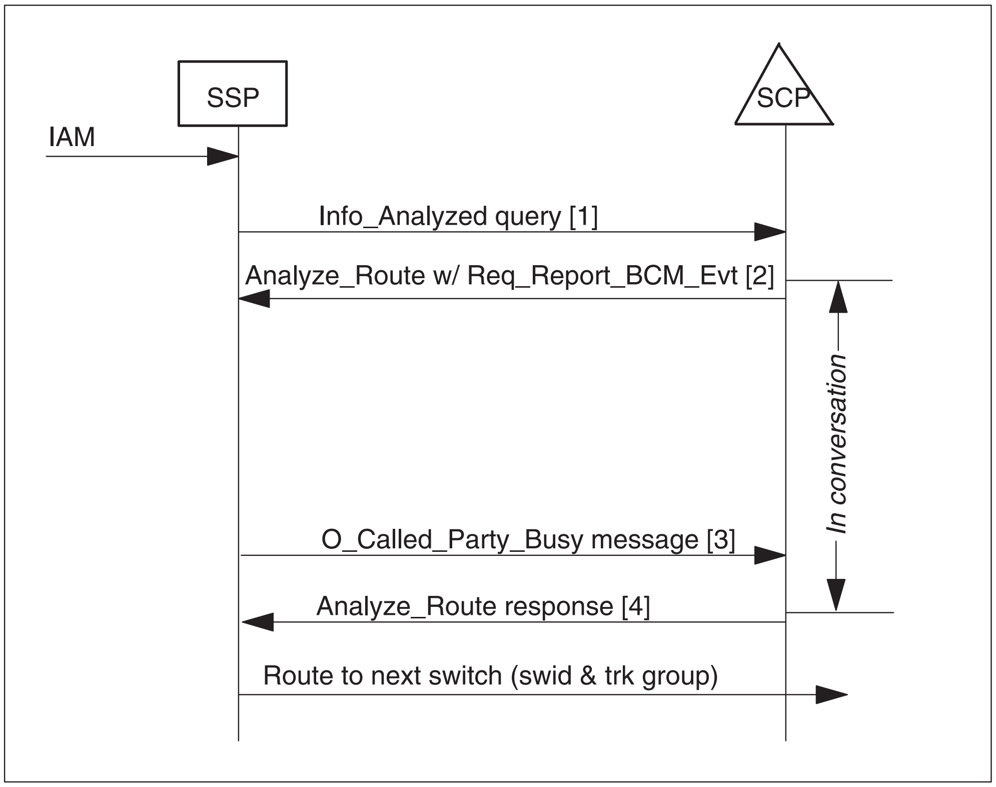
Message	**Description**
`O_Mid_Call`	Informs the SCP that the *Switch_Hook_Flash* event has occurred. ***Note:*** This message is also sent when a CAIN call meets the trigger criteria for an *O_IEC_Reorigination* trigger. Refer to Volume 1, Chapter 7, “Send_Call PIC,” and Volume 1, Chapter 8, “O_Alerting PIC,” for more information.
`O_No_Answer`	EDP-Request message that may be sent to the SCP when the ***O_No_Answer*** EDP is active and the ***O_No_Answer*** EDP is reached. The parameters and extension parameters used in the message vary depending on the function of the message. Refer to Volume 1, Chapter 8, “O_Alerting PIC,” for information.
`O_Term_Seized`	EDP-Notification message sent to the SCP when the ***O_Term_Seized*** EDP is active and a terminating trunk has been seized by the UCS DMS-250 switch. Refer to Volume 1, Chapter 7, “Send_Call PIC,” for information.
`Request_Report_BCM_Event`	Non-call related component that when present causes the switch to associate a next event list (NEL) with the open transaction. The NEL contains a list of EDPs armed to send an EDP-Request or EDP-Notification to the SCP (or perform another action) when reached.
`Timeout`	EDP-Request message sent to the SCP when the TimeoutTimer expires. Refer to Volume 1, Chapter 9, “O_Active and O_Suspended PICs,” for information.

—end—

EDP messaging scenarios

Figures 3-2 and 3-3 provide examples that are intended to clarify the switch to SCP interactions that may occur in calls involving EDPs. Many other scenarios are possible.

Figure 3-2
FGD call with EDP-Request


SSP
SCP
IAM
Info_Analyzed query [1]
Analyze_Route w/ Req_Report_BCM_Evt [2]
In conversation
O_Called_Party_Busy message [3]
Analyze_Route response [4]
Route to next switch (swid & trk group)

1 The switch sends an **`Info_Analyzed`** TDP-Request message to the SCP.

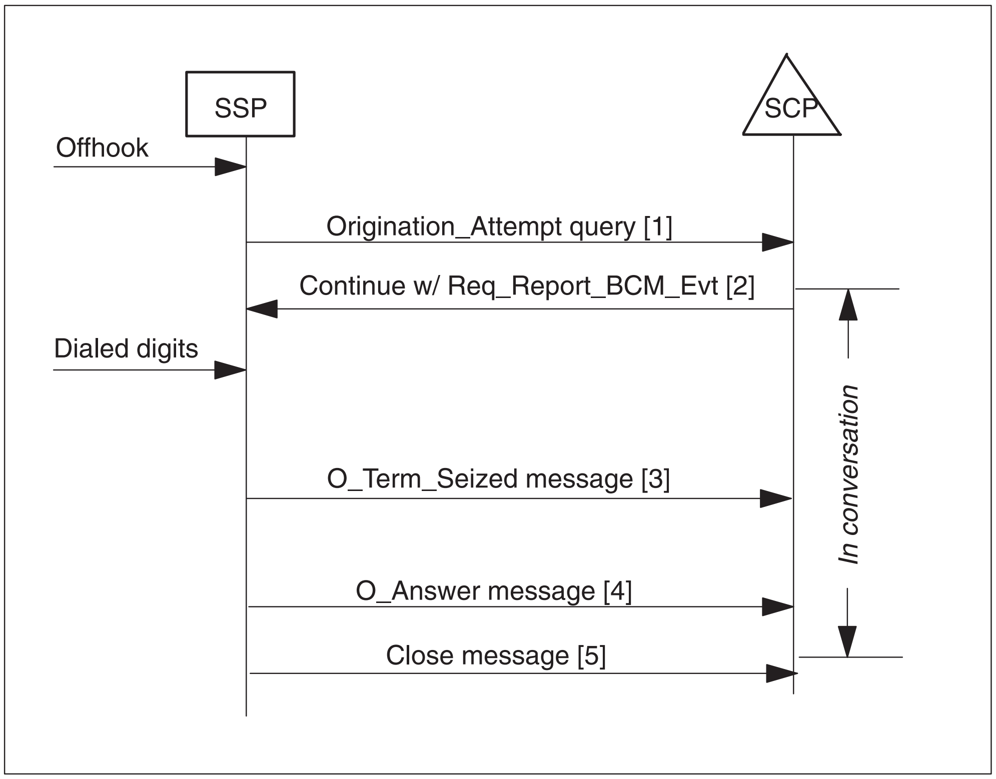
2 The SCP sends the switch a conversation package containing an **`Analyze_Route`** component followed by a **`Request_Report_BCM_Event`** component which arms the ***O_Called_Party_Busy*** EDP.

3 The switch attempts to send the originator to the route specified in the **`Analyze_Route`** component but the called party is busy. The switch sends an **`O_Called_Party_Busy`** EDP-Request in a conversation package and deactivates EDPs.

4 The SCP provides the following information in a (non-conversational) response package containing an **`Analyze_Route`** component only: ***`PrimaryTrunkGroup`*** (switch ID/trunk group)

EDP messages (continued)

Figure 3-3
DAL call with EDP-Notifications


SSP
SCP
Offhook
Origination_Attempt query [1]
Continue w/ Req_Report_BCM_Evt [2]
Dialed digits
O_Term_Seized message [3]
O_Answer message [4]
Close message [5]
In conversation

1 The switch sends an **`Origination_Attempt`** TDP-Request message to the SCP.

2 The SCP sends the SSP a conversation package containing a **`Continue`** component followed by a **`Request_Report_BCM_Event`** component which arms the ***O_Term_Seized*** and ***O_Answer*** EDPs.

3 After the originator dials an address and the terminator is seized, the switch sends an **`O_Term_Seized`** EDP-Notification to the SCP in a conversation package. EDPs are not deactivated.

4 After the terminator answers, the switch sends an **`O_Answer`** EDP-Notification to the SCP in a conversation package. EDPs are not deactivated.

5 Since no more EDPs are reachable after ***O_Answer***, the switch sends a **`Close`** message to the SCP in a response package and deactivates EDPs.

Use

The UCS DMS-250 switch sends the `Close` message to the SCP to terminate EDP processing. This message may also be used as an error message. Refer to Chapter 10, “Incoming CAIN messages” for more information.

Message parameters

Table 3-2 describes the `Close` message parameters.

Table 3-2
Close message parameters

Parameter	Usage	Definition
UserID	Optional	This parameter specifies the network identity of the caller. This value is based on the switch ID and the trunk group’s ADNUM value in table CLLI.
BearerCapability	Optional	This parameter specifies the bearer capability of the call.
CloseCause	Required	This parameter indicates the reason the transaction is being closed.
ExtensionParameter	Optional	This parameter indicates if extension parameters are sent with the `Close` message. Extension parameters are not sent with the `Close` message at this time.

Associated logs

CAIN901, VAMP902

Associated OMs

CAINMSGS

EDP messages
Network_Busy

Use

If EDPs are active when the ***Network_Busy*** detection point (DP) is reached, ***Network_Busy*** is encountered as an EDP and the switch may send a **`Network_Busy`** EDP-Request message to the SCP.

Message parameters

A **`Network_Busy`** EDP-Request is sent to the SCP in a conversation package with a component type of Invoke_Last. Table 3-3 defines the parameters and usage requirements for the parameters the **`Network_Busy`** EDP-Request may contain. Parameters used are based on the function of the message. Refer to Volume 1, Chapter 6, "Select_Route PIC," for information on the parameters for the **`Network_Busy`** TDP-Request message.

Table 3-3
Network_Busy EDP-Request message parameters

Parameter	Usage	Definition
`UserID`	Required	This parameter contains the network identity of the originating agent.
`BearerCapability`	Required	This parameter Contains the bearer capability of the call when the message is built.
`ExtensionParameter`	Optional	***`ExtensionParameter`*** requires the CAIN0200 SOC option.
`busyRoute`	Optional	This parameter contains the number of routing attempts for a call at the *Network_Busy* event. When an outpulse number is available, it is included as the routing number. If the call was routed using a trunk group and SWID from the ***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, or ***`SecondAlternateTrunkGroup`*** parameter, the route index is included.
`termTrunkInfo`	Optional	This parameter contains the terminating trunk group number, trunk type, and trunk member number.

—continued—

Table 3-3
Network_Busy EDP-Request message parameters (continued)

Parameter	Usage	Definition
NotificationIndicator	Optional	This parameter contains the value of FALSE to identify the message type as Request.
BusyCause	Optional	This parameter contains the cause value received from the SS7 or PRI agent. If the route list is exhausted, this parameter contains the value `noCircuitAvailable`
—end—		

Associated logs

CAIN100, CAIN200, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages
O_Abandon (end)

Use

The switch sends the `O_Abandon` EDP-Request message to the SCP during the second call segment. This message indicates that the controlling leg went on-hook before the called party answered.

Message parameters

Table 3-4 describes the `O_Abandon` message parameters.

Table 3-4
O_Abandon message parameters

Parameter	Usage	Definition
UserID	Required	This parameter contains the network identity of the originating agent.
BearerCapability	Required	This parameter contains the bearer capability of the call when the message is built.
CcID	Optional	This parameter specifies the current call configuration of the call.
PointInCall	Optional	This parameter contains the current point in call.
ExtensionParameter	Optional	No extension parameters are supported for this message.
NotificationIndicator	Optional	This parameter contains the value of FALSE to identify the message type as Request.

Associated logs

CAIN100, CAIN905, VAMP902

Associated OMs

CAINAGOM, CAINMSGS

Use

The **`O_Answer`** EDP-Notification message is sent to the SCP when the ***O_Answer*** EDP is active and the terminating party answers the call.

Message parameters

Table 3-5 describes the **`O_Answer`** EDP-Notification message parameters.

Table 3-5
O_Answer EDP-Notification message parameters

Parameter	Usage	Definition
`UserID`	Required	This parameter contains the network identity of the originating agent.
`BearerCapability`	Required	This parameter contains the bearer capability of the call when the message is built.
`NotificationIndicator`	Optional	This parameter contains the value TRUE to identify the message type as Notification.
`ExtensionParameter`	Optional	***`ExtensionParameter`*** requires the CAIN0200 SOC option.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.

Associated logs

CAIN100, CAIN200, CAIN901, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages O_Called_Party_Busy

Use

If EDPs are active when the ***O_Called_Party_Busy*** detection point (DP) is reached, ***O_Called_Party_Busy*** is encountered as an EDP and the switch may send a **`O_Called_Party_Busy`** EDP-Request message to the SCP.

Message parameters

An **`O_Called_Party_Busy`** EDP-Request is sent to the SCP in a conversation package with a component type of Invoke_Last. Table 3-6 defines the parameters and usage requirements for the parameters the **`O_Called_Party_Busy`** EDP-Request may contain. Parameters used are based on the function of the message. Refer to Volume 1, Chapter 7, "Send_Call PIC," for information on the parameters for the **`O_Called_Party_Busy`** TDP-Request message.

Table 3-6
O_Called_Party_Busy EDP-Request message parameters

Parameter	Usage	Definition
UserID	Required	This parameter contains the network identity of the originating agent.
BearerCapability	Required	This parameter contains the bearer capability of the call when the message is built.
BusyCause	Optional	This parameter contains the cause value received from the SS7 or PRI agent; or if the terminating agent is a DAL or AXXESS with ONNETTRK=Y, the cause is set to `userbusy`.
NotificationIndicator	Optional	This parameter contains the value FALSE to identify the message type as Request.
ExtensionParameter	Optional	***ExtensionParameter***s require the CAIN0200 SOC option

—continued—

Table 3-6
O_Called_Party_Busy EDP-Request message parameters (continued)

Parameter	Usage	Definition
`busyRoute`	Optional	This extension parameter contains the number of routing attempts for a call at the *O_Called_Party_Busy* event. When an outpulse number is available, it is included as the routing number. If the call was routed using a trunk group and SWID from the ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, or ***SecondAlternateTrunkGroup*** parameter, the route index is included.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.
—end—		

Associated logs

CAIN100, CAIN200, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages O_Disconnect

Use

The switch sends the **`O_Disconnect`** message to the SCP to indicate that the *O_Disconnect* event was detected. The switch may send this message either as an EDP-Notification or an EDP-Request, depending on the way in which the ***O_Disconnect*** EDP was armed. For further information on ***O_Disconnect*** EDP processing, refer to the "O_Disconnect EDP" section in this chapter.

Message parameters

Table 3-7 describes the **`O_Disconnect`** message parameters.

Table 3-7
O_Disconnect message parameters

Parameter	Usage	Definition
`UserID`	Required	This parameter contains the network identity of the originating agent.
`BearerCapability`	Required	This parameter contains the bearer capability of the call when the message is built.
`NotificationIndicator`	Optional	This parameter identifies the message type as notification or request.
`ExtensionParameter`	Optional	***`ExtensionParameter`***s require the CAIN0200 SOC option.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.
`connectTime`	Optional	This extension parameter indicates how much time has elapsed since the call was answered, in minutes, seconds, and tenths of seconds
`LegID`	Optional	This parameter identifies a leg in a call segment.
`PointInCall`	Optional	This parameter contains the current point in call.
Note: The **`O_Disconnect`** EDP-Request message does not support any ***`ExtensionParameter`***s.		
—end—		

Associated logs

CAIN100, CAIN200, CAIN901, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages O_Mid_Call

Use

The switch sends the **`O_Mid_Call`** EDP-Request message to the SCP to indicate that a *Switch_Hook_Flash* event has occurred.

Message parameters

Table 3-8 describes the **`O_Mid_Call`** EDP-Request message parameters.

Table 3-8
O_Mid_Call EDP-Request message parameters

Parameter	Usage	Definition
`UserID`	Required	This parameter contains the identity of the user which invokes the service (Note)
`BearerCapability`	Required	This parameter contains the bearer capability of the call when the message is built.
`CcID`	Optional	This parameter contains the call configuration identifier.
`NotificationIndicator`	Optional	This parameter contains the value FALSE to identify the message type as Request.
`ExtensionParameter`	Optional	***`ExtensionParameter`***s require the CAIN0200 SOC option.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.

Note: The switch populates the ***`UserID`*** parameter with the identity of the user which invokes the service. Currently, only the terminator is capable of subscribing to the *Switch_Hook_Flash* event. Therefore, the switch populates the ***`UserID`*** parameter with the value for the terminator during call configuration (CC) 2 (stable two-party call). In call configurations following CC2, the switch populates the ***`UserID`*** parameter with the value for the controlling leg. Refer to Chapter 4, “Call Configurations,” for more information.

—end—

Associated logs

CAIN100, CAIN200, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages
O_No_Answer

Use

The `O_No_Answer` EDP-Request message may be sent to the SCP when EDPs are active, the ***O_No_Answer*** detection point (DP) is reached, and ***O_No_Answer*** is encountered as an EDP. Parameters used are based on the function of the message. Refer to Volume 1, Chapter 8, "O_Alerting PIC," for information on the parameters for the `O_No_Answer` TDP-Request message.

Message parameters

An `O_No_Answer` EDP-Request message is sent to the SCP in a conversation package with a component type of Invoke_Last. Table 3-9 defines the `O_No_Answer` EDP-Request message parameters. Parameters used are based on the function of the message. Refer to Volume 1, Chapter 8, "O_Alerting PIC," for information on the parameters for the `O_No_Answer` TDP-Request message.

Table 3-9
O_No_Answer EDP-Request message parameters

Parameter	Usage	Definition
UserID	Required	This parameter contains the network identity of the originating agent.
BearerCapability	Required	This parameter contains the bearer capability of the call when the message is built.
NotificationIndicator	Optional	This parameter contains the value FALSE to identify the message type as Request.
ExtensionParameter	Optional	***ExtensionParameters*** require the CAIN0200 SOC option.

—continued—

EDP messages
O_No_Answer (end)

Table 3-9
O_No_Answer EDP-Request message parameters (continued)

Parameter	Usage	Definition
`busyRoute`	Optional	This extension parameter contains the number of routing attempts for a call at the *O_No_Answer* event. When an outpulse number is available, it is included as the routing number. If the call was routed using a trunk group and SWID from the ***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, or ***`SecondAlternateTrunkGroup`*** parameter, the route index is included.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.
—end—		

Associated logs

CAIN100, CAIN200, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

EDP messages
O_Term_Seized (end)

Use

The `O_Term_Seized` EDP-Notification message is sent to the SCP when the ***O_Term_Seized*** EDP is active and a terminating trunk has been seized by the UCS DMS-250 switch.

Message parameters

Table 3-10 describes the `O_Term_Seized` EDP-Notification message parameters.

Table 3-10
O_Term_Seized EDP-Notification message parameters

Parameter	Usage	Definition
UserID	Required	This parameter contains the network identity of the originating agent.
BearerCapability	Required	This parameter contains the bearer capability of the call when the message is built.
NotificationIndicator	Optional	This parameter contains the value TRUE identifying the message type as Notification.
ExtensionParameter	Optional	***ExtensionParameter***s require the CAIN0200 SOC option.
termTrunkInfo	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.

Associated logs

CAIN100, CAIN200, CAIN901, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

Use

The SCP sends a **`Request_Report_BCM_Event`** non-call related component in a conversational TCAP package along with a call-related **`Acknowledge`**, **`Analyze_Route`**, **`Collect_Information`**, **`Continue`**, **`Disconnect_Leg`**, **`Merge_Call`**, or **`Originate_Call`** component. The presence of the **`Request_Report_BCM_Event`** component causes the switch to associate a next event list (NEL) with the open transaction. The NEL is a list of events for which the switch sends an EDP-Request or EDP-Notification message to the SCP (or possibly perform another action) when detected.

Note: The **`Continue`** message is only allowed to include the **`Request_Report_BCM_Event`** component in response to **`Origination_Attempt`**, **`Info_Analyzed`**, or **`CTR_Clear`** messages.

Message parameters

Table 3-11 describes the **`Request_Report_BCM_Event`** message parameters.

Table 3-11
Request_Report_BCM_Event parameters

Parameter	Usage	Definition
`EDPRequest`	Optional	This parameter specifies an event for which the switch sends an EDP-Request when detected.
`EDPNotification`	Optional	This parameter specifies an event for which the switch sends an EDP-Notification when detected.
`ONoAnswerTimer`	Optional	This parameter indicates the value, in seconds, of the switch originating no answer (O_No_Answer) timer, used to determine when the ***O_No_Answer*** EDP is encountered.
`ExtensionParameter`	Optional	***`ExtensionParameters`*** require the CAIN0200 SOC option.
`networkBusyActions`	Optional	This extension parameter specifies the action to take when the ***Network_Busy*** EDP is encountered.

—continued—

EDP messages
Request_Report_BCM_Event (continued)

Table 3-11
Request_Report_BCM_Event parameters (continued)

Parameter	Usage	Definition
`oCalledPartyBusyActions`	Optional	This extension parameter specifies the action to take when the ***O_Called_Party_Busy*** EDP is encountered.
`oNoAnswerActions`	Optional	This extension parameter specifies the action to take when the ***O_No_Answer*** EDP is encountered.
`edpBuffer`	Optional	If present, this extension parameter indicates that digits should be buffered when an EDP-Request is sent.
`shfelegs`	Optional	This extension parameter specifies which leg or legs the switch must monitor for a switch-hook flash.
TimeoutTimer	Optional	This parameter specifies the value, in minutes, of the switch terminating timeout timer, used to determine when the *Timeout* event is detected
—end—		

Request_Report_BCM_Event extension parameters

NetworkBuilder supports five extension parameters for the **`Request_Report_BCM_Event`** component so that the SCP has the ability to specify any of the following tasks to the switch:

- what action to take based on routing criteria (Note)
- whether buffering of digits should occur when the switch sends an EDP-Request (Note)
- which leg or legs the switch must monitor for a switch-hook flash

Note: These extension parameters for EDPs are the equivalent of the routing criteria fields and the BUFFER option in trigger tables for the *Network_Busy*, *O_Called_Party_Busy*, and *O_No_Answer* triggers. By using these extension parameters, EDPs have the same functionality as their corresponding triggers.

These extension parameters may be datafilled in table CAINXDFT to provide the switch default extension parameter values for a given CAIN group. The switch uses the values datafilled in table CAINXDFT when a

`Request_Report_BCM_Event` component is received without an extension parameter which is included in table CAINXDFT for the active CAIN group. If a given extension parameter is included in both the **`Request_Report_BCM_Event`** component and the CAINXDFT tuple for the active CAIN group, the value received from the SCP in the **`Request_Report_BCM_Event`** component is used.

Note: When the SCP returns a **`Request_Report_BCM_Event`** component which arms an event without a corresponding EDP action extension parameter (`networkBusyActions`, `oCalledPartyBusyActions`, `oNoAnswerActions`), it is instructing the switch to send the EDP-Request when the EDP (***Network_Busy***, ***O_Called_Party_Busy***, ***O_No_Answer***) is encountered. The use of the table CAINXDFT allows the switch to override the action indicated by the SCP in this case. In order to prevent the SCP from being overridden by the switch, the SCP would need to send the extension parameter with all fields having the value REQUEST.

For additional information on **`Request_Report_BCM_Event`** extension parameters, refer to Chapter 12, “Incoming CAIN message parameters.”

EDP messages
Request_Report_BCM_Event (continued)

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 3-12 shows fatal application errors that can occur after a **`Request_Report_BCM_Event`** component is returned from the SCP.

Table 3-12
Request_Report_BCM_Event fatal application errors

Error type	Definition	Log generated	Reported to SCP?	Error action performed
Unexpected message sequence	The first component is a non-call related component.	CAIN200	Yes	ERRACT in trigger table (Note)
Unexpected message	Message not received with allowed component. (Refer to “Event arming restrictions” section in this chapter for more information.)	CAIN200	Yes	ERRACT in trigger table (Note)
Erroneous data value	***`EDPRequest`*** or ***`EDPNotification`*** parameter is invalid. (Refer to “Event arming restrictions” section in this chapter for more information.)	CAIN200	Yes	ERRACT in trigger table (Note)
Erroneous data value	No events are armed.	CAIN200	Yes	ERRACT in trigger table (Note)

Note: Trigger tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision an error action; AINF is used for all other triggers and events.

—end—

Event arming restrictions

The **`Request_Report_BCM_Event`** component can accompany the following call-related components:

- **`Acknowledge`**
- **`Analyze_Route`**
- **`collect_Information`**
- **`Continue`**

> ***Note:*** The **`Continue`** message is only allowed to include a **`Request_Report_BCM_Event`** component when the switch receives it in reply to an **`Origination_Attempt`**, **`Info_Analyzed`**, or **`CTR_Clear`**

- **`Disconnect_Leg`**
- **`Merge_Call`**
- **`Originate_Call`**

According to the type of call-related SCP response received with the **`Request_Report_BCM_Event`** component, only certain events may be present in the ***`EDPNotification`*** and ***`EDPRequest`*** parameters. If all the events requested in the **`Request_Report_BCM_Event`** message are either ignored or unreachable, the switch sends a **`Close`** message to the SCP with a ***`CloseCause`*** of `eDPsCompleted`.

Figure 3-4 summarizes event arming restrictions. The restrictions are dependent on the type of call-related SCP response received with the **`Request_Report_BCM_Event`** component.

EDP messages
Request_Report_BCM_Event (continued)

Figure 3-4
Event arming restrictions

Event \ message type	Acknowledge	Analyze_Route	Collect_Information	Continue (Note 1)	Continue (Note 2)	Disconnect_Leg	Merge_Call	Originate_Call
Network_Busy	X	R	X	X	X	X	X	X
O_Abandon	R	RI	RI	X	X	R	R	R
O_Answer	X	N	N	N	X	X	X	X
O_Called_Party_Busy	X	R	R	R	X	X	X	X
O_Disconnect	R	Z	Z	N	N	R	R	R
O_No_Answer	X	R	R	R	X	X	X	X
O_Term_Seized	X	N	N	N	X	X	X	X
Switch_Hook_Flash (Note 3)	R	R	R	X	X	R	R	R
Timeout	X	R	R	X	R	X	X	X

LEGEND
RI - Request Ignored; event may only be present in the **EDPRequest** parameter of the **Request_Report_BCM_Event** component, but the switch will ignore the request to arm the event
R– event may be requested in the **EDPRequest** parameter of the **Request_Report_BCM_Event** component
X– event cannot be requested in either the **EDPRequest** or the **EDPNotification** parameter of the **Request_Report_BCM_Event** component
N– event may be requested in the **EDPNotification** parameter of the **Request_Report_BCM_Event** component
Z – request to arm the event as an **EDPRequest** is ignored but a request to arm the event as an **EDPNotification** is allowed
Note 1: **Continue** received after an ***Origination_Attempt*** or ***Info_Analyzed***
Note 2: **Continue** received after a **CTR_Clear**
Note 3: For the **Analyze_Route** and **Collect_Information** messages, the request to arm the *Switch_Hook_Flash* event is ignored if the *Timeout* event is also requested.

Nonfatal application errors

Table 3-13 shows nonfatal application errors that can occur while the **`Request_Report_BCM_Event`** component is being processed.

Table 3-13
Request_Report_BCM_Event nonfatal application errors

Error type	Definition	Log generated	Reported to SCP?	Error action performed
Unexpected message	Message received in reply to an LNP query	CAIN100	Yes	**`Close`** message is sent to the SCP with a ***`CloseCause`*** parameter value of `unexpected Communication`.
Unexpected message	Class-of-service screening failed in the **`Analyze_Route`** component	CAIN100	Yes	**`Close`** message is sent to the SCP with a ***`CloseCause`*** parameter value of `eDPsCompleted`.
Unexpected parameter	***`ONoAnswerTimer`*** or ***`TimeoutTimer`*** parameter received without requesting the corresponding event	CAIN100	No	Parameters are ignored.
Unexpected parameter	SOC for extension parameters (CAIN0200) is idle. See Note 2.	CAIN102	No	Extension parameters are ignored.

Note 1: A **`Close`** message is sent to the SCP with a ***`CloseCause`*** parameter value of `unexpectedCommunication` if the **`Request_Report_BCM_Event`** component received does not request any events which are permitted by SOC options CAIN0602, CAIN0800, or CAIN0802.
Note 2: Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information on SOC.

—continued—

EDP messages
Request_Report_BCM_Event (end)

Table 3-13
Request_Report_BCM_Event nonfatal application errors (continued)

Error type	Definition	Log generated	Reported to SCP?	Error action performed
Unexpected communication	*Network_Busy*, *O_Term_Seized*, *O_Called_Party_Busy*, *O_Answer*, and/or *O_No_Answer* events are requested, but SOC CAIN0602 is idle. See Note 2.	CAIN102	Refer to Note 1.	Refer to Note 1.
Unexpected communication	*Timeout* and/or *O_Disconnect* events are requested, but SOC CAIN0800 is idle. See Note 2.	CAIN102	Refer to Note 1.	Refer to Note 1
Unexpected communication	*O_Abandon* and/or *Switch_Hook_Flash* events are requested, but SOC CAIN0802 is idle. See Note 2.	CAIN102	Refer to Note 1.	Refer to Note 1.

Note 1: A **`Close`** message is sent to the SCP with a ***`CloseCause`*** parameter value of `unexpectedCommunication` if the **`Request_Report_BCM_Event`** component received does not request any events which are permitted by SOC options CAIN0602, CAIN0800, or CAIN0802.
Note 2: Refer to Volume 5, Chapter 5, "NetworkBuilder SOC functionality," for more information on SOC.

—end—

Associated logs

CAIN100, CAIN101, CAIN102, CAIN200, VAMP901

Note: For more information on logs, refer to the *UCS DMS-250 Logs Reference Manual*.

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Use

The switch sends the **`Timeout`** EDP-Request message to the SCP during the **O_Active** or **O_Suspended** PICs when the TimeoutTimer expires.

Message parameters

Table 3-14 describes the **`Timeout`** EDP-Request message parameters.

Table 3-14
Timeout EDP-Request message parameters

Parameter	Usage	Definition
`UserID`	Required	This parameter contains the network identity of the originating agent.
`BearerCapability`	Required	This parameter contains the bearer capability of the call when the message is built.
`CcID`	Optional	This parameter contains the call configuration identifier.
`ExtensionParameter`	Optional	***`ExtensionParameter`***s require the CAIN0200 SOC option.
`termTrunkInfo`	Optional	This extension parameter contains the terminating trunk group number, trunk type, and trunk member number.
`connectTime`	Optional	This extension parameter indicates how much time has elapsed since the call was answered, in minutes, seconds, and tenths of seconds.
`NotificationIndicator`	Optional	This parameter identifies the message type as request.

Associated logs

CAIN100, CAIN200, CAIN905, VAMP902

Associated OMs

CAINMSGS, CAINAGOM, CAINTRIG

Call configuration model

The call configuration (CC) model describes a call's various states or stages. Each call configuration in the model provides a view of both call processing (the point in the call) and connection processing (the number and orientation of the call legs present in the call). The switch and the SCP use these views during the course of call party handling to describe events and transitions within the call.

Call party handling refers to the transitions made between call configurations. Transitions occur due to

- end-user actions (for example, a call's party presses the star (*) key for 40 milliseconds or disconnects).
- switch call processing (for example, translations or switch-based features).
- call connection view processing of SCP response messages (for example, **`Connect_To_Resource`** message processing).

This chapter provides the following information:

- an overview of the concepts involved in the call configuration model and diagrams
- detailed descriptions of each call configuration
- an example of a transfer call using the call configuration model
- errors the switch may detect while a call is in a CC
- a quick-reference to the call configuration model

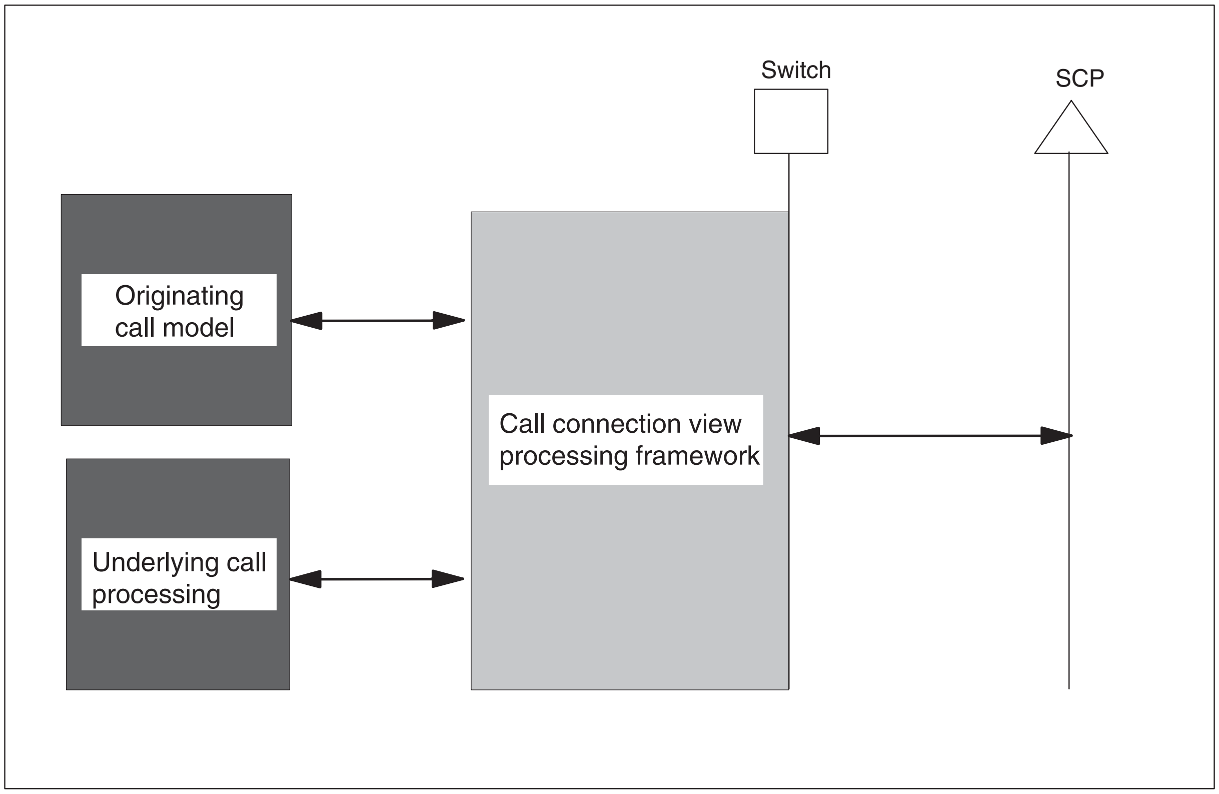
Call connection view processing

The call connection view processing framework allows the switch

- to process SCP response messages relating to the call configuration.
- to provide the N00 take-back and transfer service.

This framework works with the originating call model and call processing frameworks. Figure 4-1 shows how the three frameworks work together.

Figure 4-1
Connection view processing framework


Switch
SCP
Originating call model
Call connection view processing framework
Underlying call processing

Call configurations diagrams explained

Diagrams show and explain the status of the connections within the call configurations.

A call configuration diagram has four parts:

- connection points
- call legs
- call segments
- call segment association

Connection points

Connection points are the points where the switch has connected the call legs. The call configuration diagram uses a circle to represent the connection points.

Call legs

Call legs are the trunks or lines that connect the parties in a call. The call configurations diagrams show the call legs as solid or broken lines.

Call segments

A call segment is a part (segment) of a call. The call segment contains zero to three call legs and a connection point.

A call has one or two segments. Whether the call has one or two call segments depends on how many connection points the call has. If the call has one connection point, the call has one call segment. If the call has two connection points, the call has two call segments. A call has a maximum of two connection points, so a call can have only two call segments. Figure 4-2 shows a diagram of a call configuration with a one call segment. Figure 4-3 shows a diagram of a call configuration with two call segments.

Figure 4-2
Call configuration diagram with one call segment

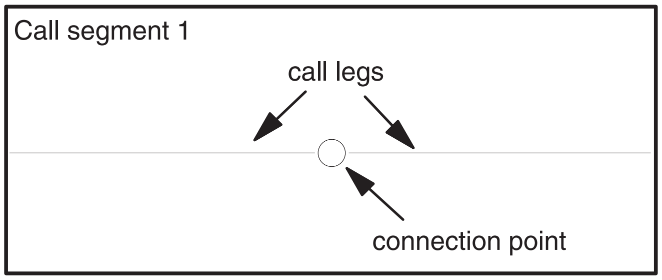
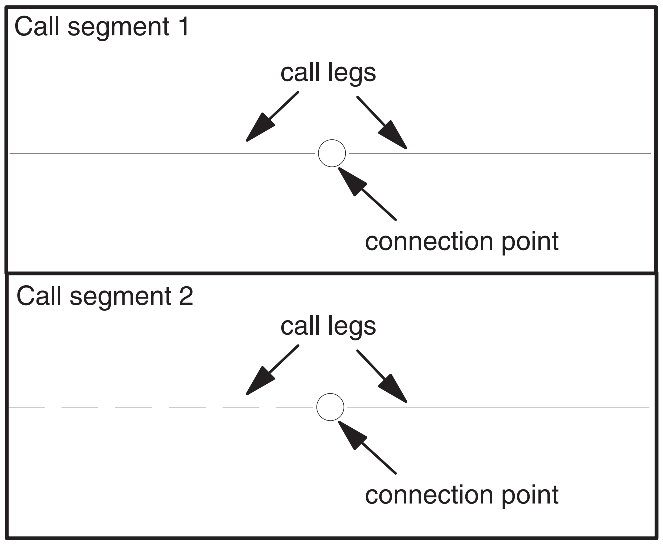


Figure 4-3
Call configuration diagram with two call segments



When a call configuration has two call segments, the call segment appearing at the diagram's top is the only segment where the switch can process events. Therefore, when call segment one appears above call segment two, the switch can process events in call segment one. When call segment two appears above call segment one, the switch can process events in call segment two.

Call segment identification parameter

The switch assigns identification numbers to the call segments. Call segment 1 is the call segment where the call originated. For example, a two-party call takes place within one call segment. This is the call segment where the call originated. A three-party call has two call segments:

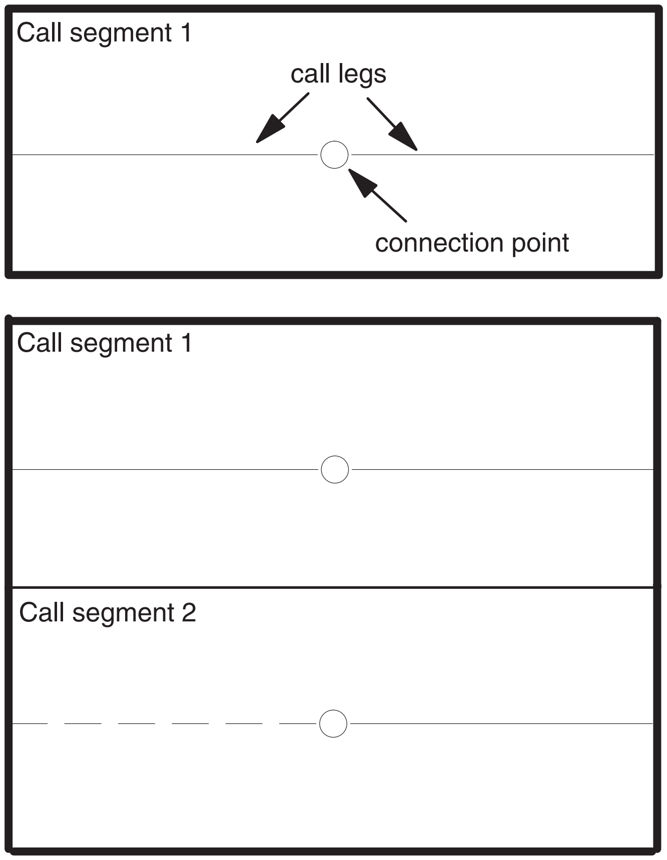
- the call segment where the two-party call originated (call segment 1)
- the call segment where the call to the third party originated (call segment 2)

Through the ***`CsID`*** parameter, the switch uses the call segment identification numbers to send and receive information on specific call segments.

Call segment association

The call segment association is the relationship between the call segments within a call. The call configuration diagram is the call segment association. In Figure 4-4, the heavy line around the call segments shows the call segment associations in call configurations with one and two call segments.

Figure 4-4
Call segment association



Supported call configurations

NetworkBuilder supports eleven call configurations:

- CC0, (Null)
- CC1 (Originating setup)
- CC2 (Stable two-party call)
- CC4 (Three-party setup)
- CC5 (Three-party setup complement)
- CC6 (Party on hold)
- CC6 subset (Subset of party on hold)
- CC7 (Party on hold complement)
- CC10 (Stable multi-party call)
- CC10 subset (Subset of stable multi-party call)
- CC11 (Transfer)

During CC4-CC11, inclusive, the switch must maintain an open transaction with the SCP. If the switch does not maintain an open transaction, the switch detects an error and provides final treatment.

Types of call legs

Two types of call legs exist:

- controlling leg
- passive leg

Each call configuration has one controlling leg and one or two passive legs. (Because call processing is inactive during call configuration 0 [null], CC0 is the only call configuration without call legs present.)

Whether a call leg is a controlling or passive leg depends on the call legs role during the call. A controlling leg is the call leg that either originates the call or initiates a transition from call configuration 2 (stable two-party call) to call configuration 4 (three-party setup).

A passive leg is a call leg that remains mainly inactive during the call. The switch connects a passive leg to the connection point mainly as a result of the controlling leg’s actions. A passive leg can only cause a transition from one call configuration to another call configuration by disconnecting.

When the call legs’ roles change during the call, the switch reassigns the call legs’ types. A controlling leg can become a passive leg and a passive leg can become a controlling leg.

When a caller originates a two-party call, the leg connecting this caller to the connection point is the controlling leg. The leg connecting the terminating party to the connection point is the passive leg. When a terminating party in a two-party call places the originating caller on hold and initiates a query to the SCP, the terminating party's leg becomes the controlling leg. The originating caller's leg becomes the passive leg.

Despite the reassignment of call legs, the call configurations diagrams always show the controlling leg to the left of the connection point and the passive legs to the right of the connection point. For more information on how the switch reassigns call legs' types, refer to "Call configuration 4" in this chapter.

Leg identification parameter

The switch assigns leg identification numbers to the call legs. The switch uses the call leg type to assign the identification numbers. The controlling leg's identification number is 0. The passive legs' identification numbers are 1 and 2.

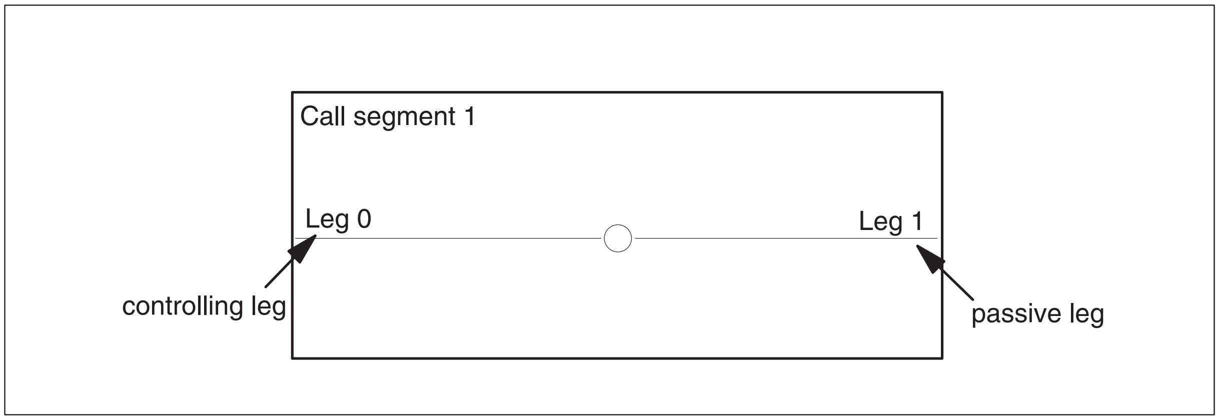
The call leg's identification number corresponds to the call leg's type. When the switch changes a call leg's type, the switch also changes the call leg's identification number. For example, during a call's progression, the switch changes a call leg's type from a controlling leg to a passive leg. When the switch makes this change, the switch also changes the call leg's identification number from 0 to 1.

Note: The switch never changes a call leg's identification number from 0 to 2.

Through the ***`LegID`*** parameter, the switch uses the leg identification numbers to send and receive information on specific call legs.

The call configurations diagrams label the controlling leg "Leg 0" and the passive legs "Leg 1" and "Leg 2."

Figure 4-5
Call legs in the call configuration diagram



Status of call legs

Table 4-1 describes the different statuses a call leg can have, describes which call leg (controlling or passive) can have the status, and describes which call segment the leg status occurs.

Table 4-1
Statuses of call legs

Status	Description of status	Applies to	Occurs in
joined	The legs are connected and are in a stable phase.	both controlling and passive legs	any call segment except CC0 (null)
pending	The controlling leg is attempting to connect with a passive leg, but the switch has not yet selected a route.	passive legs	call segment where the controlling leg is attempting to connect to the passive leg

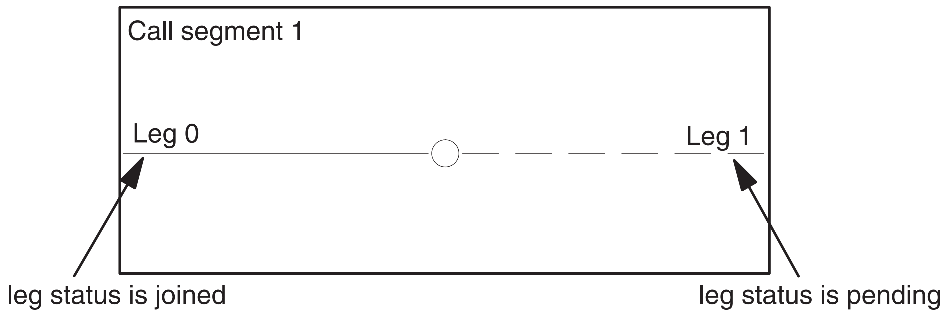
—continued—

Table 4-1
Statuses of call legs (continued)

Status	Description of status	Applies to	Occurs in
shared	The controlling leg is present in two call segments because the controlling leg has placed a passive leg on hold and has initiated a second call.	controlling leg	call segment where a passive leg is on hold
surrogate	The controlling leg has disconnected from the call leaving the passive legs connected. The controlling leg still has an active billing relationship with the remaining passive legs.	controlling leg	call segment where only the passive legs remain in the call
	—end—		

When the call leg has a status of joined, the call configuration diagrams represent this status with a solid line. When the call leg has a status of pending, shared, or surrogate, the call configuration diagrams represent this status with a broken line. Refer to Figure 4-6.

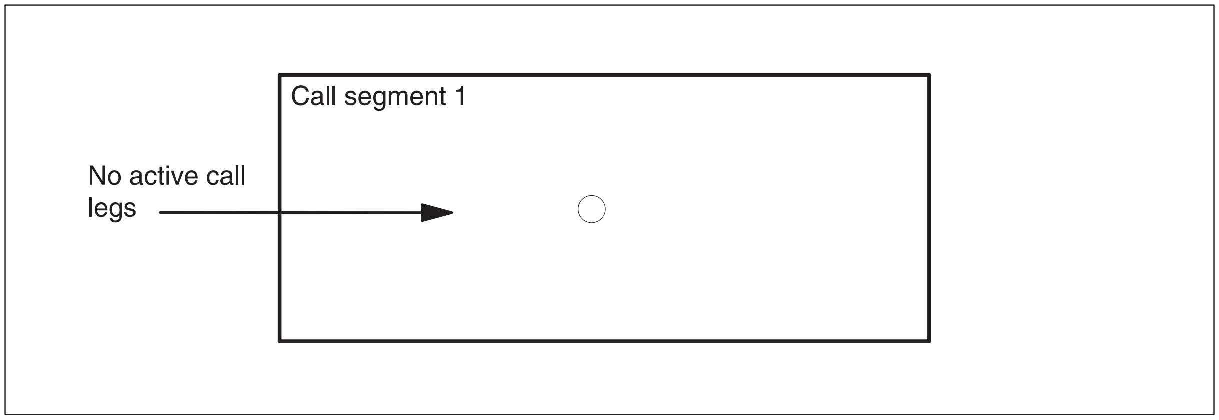
Figure 4-6
Call legs in the call configuration diagram



Call configuration 0
Null (end)

Call configuration 0 (CC0), null, represents a state where call processing is inactive. The call legs are absent from the configuration. No connections exist during CC0. Refer to Figure 4-7.

Figure 4-7
CC0 (null)



Transitions

The call transitions from CC0 to CC1 (Originating two-party setup).

Transitioning to CC1

Table 4-2 describes the user's and the switch's actions that cause the call to transition from CC0 to CC1.

Table 4-2
Transitioning to CC1

The user's actions	The switch	Then the switch receives
takes the phone off the hook (goes "off-hook").	detects the off-hook state (encounters the ***Origination_Attempt*** TDP).	not applicable

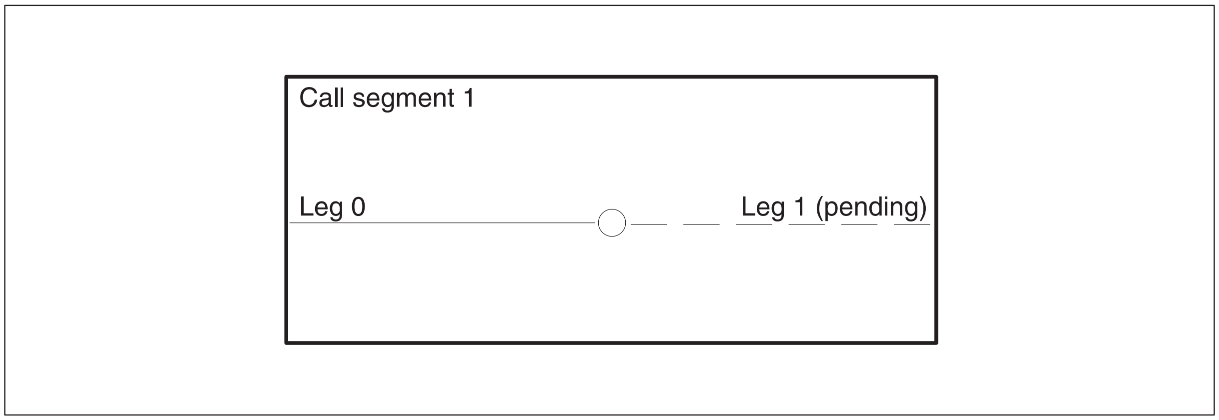
Call configuration 1 Originating setup (continued)

Call configuration 1 (CC1), originating setup, represents a state where an originating two-party call is in the setup phase. The switch is establishing a connection between the originating and terminating parties (legs 0 and 1). The connection to the terminating party (leg 1) is pending.

In CC1, the originating party is the controlling leg (leg 0). The terminating party is the passive leg (leg 1).

Refer to Figure 4-8.

Figure 4-8
CC1 (originating setup)



Call configuration 1
Originating setup (continued)

Transitions

CC1 can transition to two call configurations:

- CC0 (null)
- CC2 (stable two-party call)

Transitioning to CC0

Table 4-3 describes the user's and the switch's actions that cause the call to transition from CC1 to CC0.

Table 4-3
Transitioning to CC0

The user's action	The switch	Then the switch receives
The originating party (leg 0) disconnects.	detects events related to exception processing or events related to an incomplete call.	not applicable

Transitioning to CC2

Table 4-4 describes the user's and the switch's actions that cause the call to transition from CC1 to CC2.

Table 4-4
Transitioning to CC2

the user's actions	the switch	then the switch receives
not applicable	selects a route and is about to seize the terminating trunk.	not applicable

Remaining in CC1

Table 4-5 shows actions the users and switch take that cause the call to remain in CC1 and not transition to another call configuration.

Table 4-5
Remaining in CC1

The user's action	The switch	Then the switch receives
not applicable	encounters the ***Info_Collected*** TDP	not applicable
not applicable	encounters the ***Info_Analyzed*** TDP	not applicable

Call configuration 2
Stable two-party call (continued)

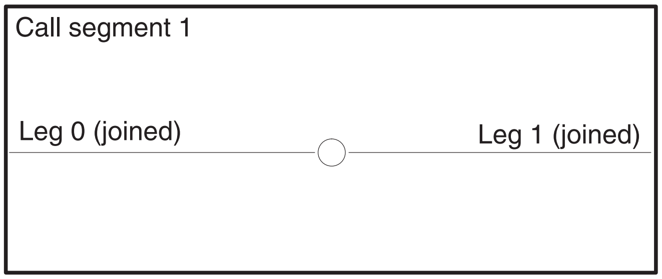
Call configuration 2 (CC2), stable two-party call, represents a key state where a two-party call has a stable connection. The switch has established a stable connection between the originating and terminating parties (legs 0 and 1).

Note: A stable connection occurs when the switch selects a route and is about to seize the terminating trunk.

In CC2 the originating party is the controlling leg (leg 0) and the terminating party is the passive leg (leg 1).

Refer to Figure 4-9.

Figure 4-9
CC2 (stable two-party call)



Reentering CC2

The call reenters CC2 when leg 1 or 2 disconnects. If leg 1 disconnects, the switch sends a close message and the SCP closes the transaction. If leg 2 disconnects, the switch reassigns the originating party as the controlling leg and the terminating party as the passive leg. When a call reenters CC2, NetworkBuilder can arm only the *Switch_Hook_Flash* event. All other events are invalid.

Note: For information on arming EDPs upon reentering CC2, see Chapter 3, “Event Processing,” in this volume. For more information on the reassigning of call legs, refer to “Call configuration 4” within this chapter.

Call configuration 2
Stable two-party call (continued)

Transitions

CC2 can transition to two call configurations:

- CC0 (null)
- CC4 (three-party setup)

Transitioning to CC0 (null)

Table 4-6 shows the user's and switch's actions that cause the call to transition from CC2 to CC0.

Table 4-6
Transitioning to CC0

The user's actions	The switch	Then the switch receives
A party allows the timeout timer to expire.	sends a **`Timeout`** EDP-request message.	the **`Connect_To_Resource`** SCP response message in a response package or the **`Disconnect`** SCP response message.

Transitioning to CC4 (three-party setup)

Table 4-7 shows the user's and switch's actions that cause the call to transition from CC2 to CC4.

Table 4-7
Transitioning to CC4

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	an **`Originate_Call`** SCP response message.
The terminating party (leg 0) enters digits as requested by a resource.	sends a **`CTR_Clear`** message.	an **`Originate_Call`** SCP response message.
Note: When the call transitions from CC2 to CC4, the switch reassigns leg identification numbers. See "Reassignment of leg identification numbers" in the "CC4 (three-party setup)" section for more information.		

Call configuration 2 Stable two-party call (end)

Remaining in CC2

Table 4-8 shows actions the users and switch take that cause the call to remain in CC2 and not transition to another call configuration.

Table 4-8
Remaining in CC2

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Connect_To_Resource`** SCP response message in a response package.
The terminating party (leg 0) enters digits as requested by a resource.	sends a **`CTR_Clear`** message.	a **`Connect_To_Resource`** SCP response message.
A party allows the timeout timer to expire.	sends a **`Timeout`** EDP-request message	a **`Connect_To_Resource`** SCP response message in a conversation package.

Call configuration 4
Three-party setup (continued)

Call configuration 4 (CC4), three-party setup, represents a state where the originating party (leg 1) is on hold and the terminating party (leg 0) has originated a call to a third party (leg 2).

This call configuration has two call segments:

- In call segment 1, leg 1 is on hold and leg 0 remains in call segment 1 with a status of shared.
- In call segment 2, leg 0 is in a setup phase with leg 2. Leg 2 has a status of pending.

Call segment 2 is above call segment 1 in this call configuration diagram. This means the switch can only process events in call segment 2.

In CC4, the originating party is a passive leg (leg 1), the terminating party is the controlling leg (leg 0), and the third party is a passive leg (leg 2).

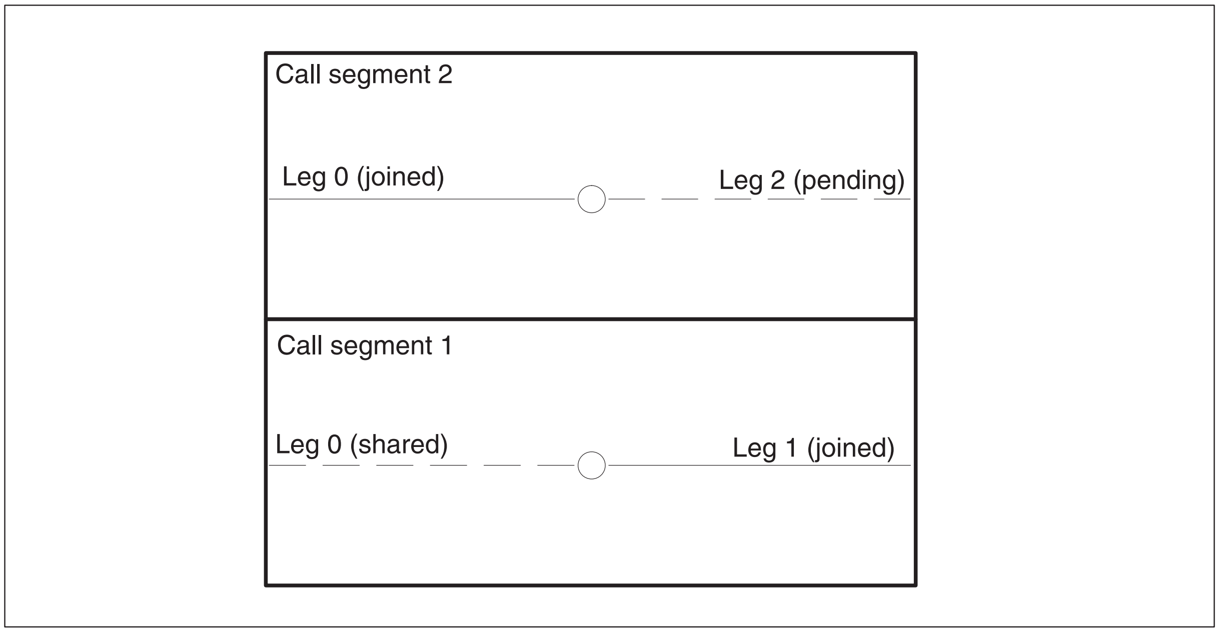
Refer to Figure 4-10.

Technically, leg 0 is an originating party since it is originating a call to the third party. And leg 2 is a terminating party since the second call is terminating to this leg. But, to simplify the discussion of call configurations, this chapter always uses the following terminology:

- the party who originated the first two-party call (in CC1) is the “originating party”
- the party to which the first two-party call terminated (in CC1) is the “terminating party”
- the party to which the second call terminates (in CC4) is the “third party”

Call configuration 4
Three-party setup (continued)

Figure 4-10
CC4 (three-party setup)



Reassigning leg identification numbers

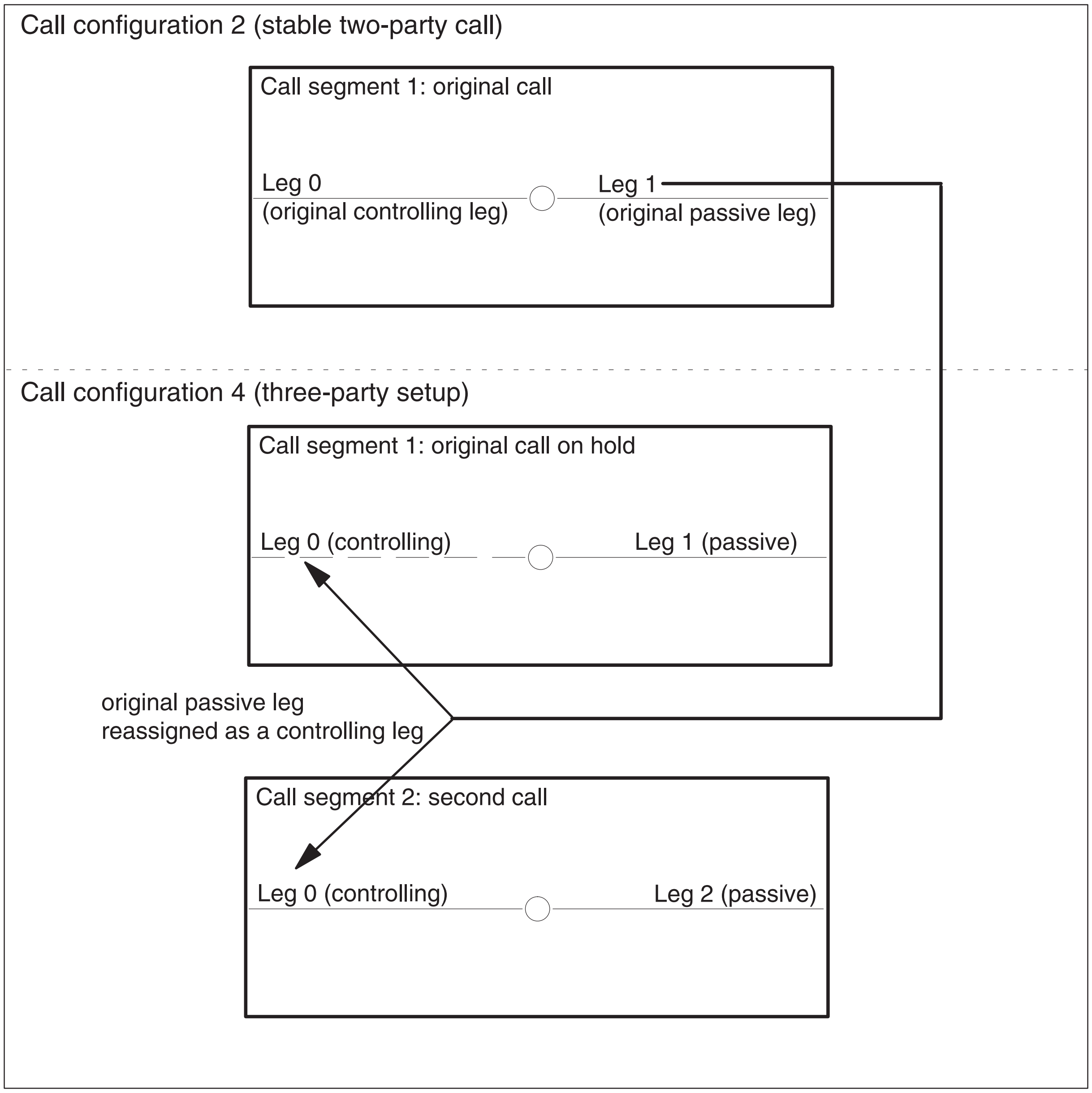
When the call transitions from CC2 to CC4, the switch reassigns leg identification numbers. During CC2, the terminating party is a passive leg (leg 1). When the terminating party (leg 1) initiates a second call, the switch reassigns the terminating party from a passive leg (leg 1) to a controlling leg (leg 0). Refer to Figure 4-11.

During CC2, the originating party is the controlling leg (leg 0). When the terminating party initiates a second call, the switch reassigns the originating party from the controlling leg (leg 0) to a passive leg (leg 1). Refer to Figure 4-12.

When the terminating party initiates the second call, the switch creates a new passive leg. The new passive leg is the third party's call leg. The switch gives the new passive leg a leg identification number of 2. Refer to Figure 4-13.

Call configuration 4
Three-party setup (continued)

Figure 4-11
Passive leg reassigned as a controlling leg


Call configuration 2 (stable two-party call)
Call segment 1: original call
Leg 0
(original controlling leg)
Leg 1
(original passive leg)
Call configuration 4 (three-party setup)
Call segment 1: original call on hold
Leg 0 (controlling)
Leg 1 (passive)
original passive leg
reassigned as a controlling leg
Call segment 2: second call
Leg 0 (controlling)
Leg 2 (passive)

Call configuration 4
Three-party setup (continued)

Figure 4-12
Controlling leg reassigned as a passive leg

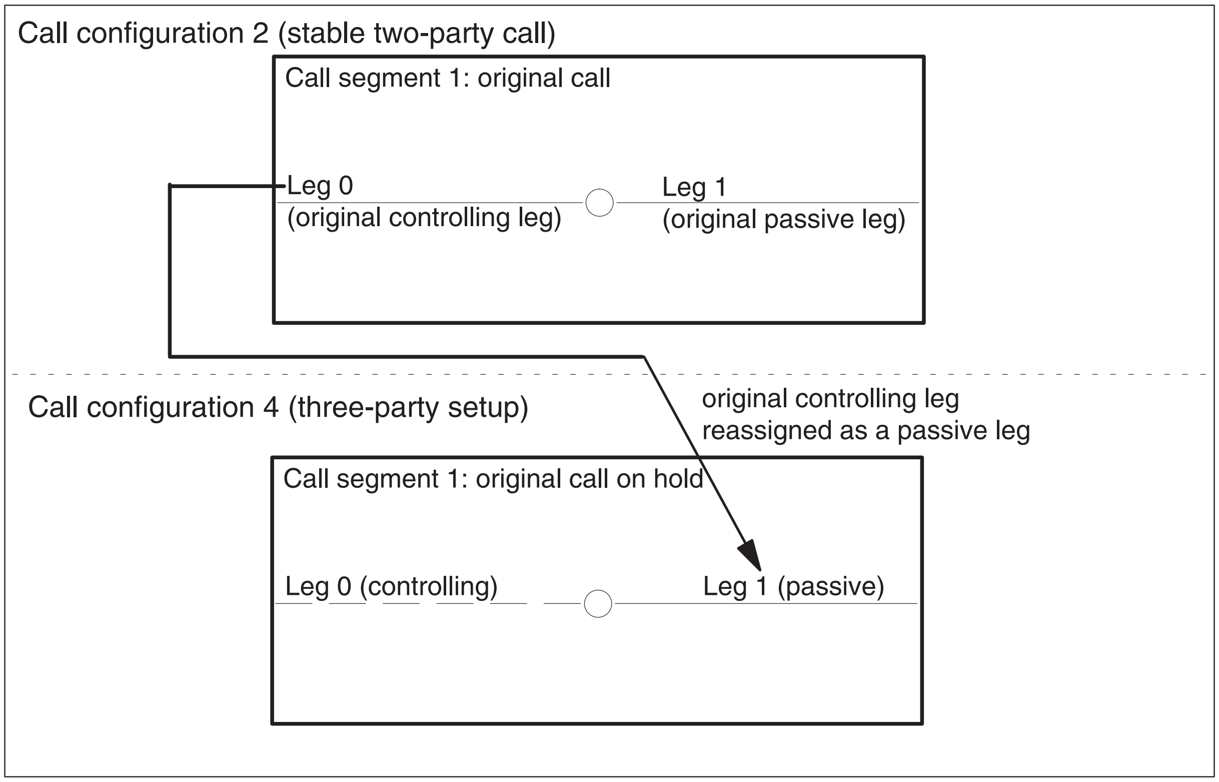
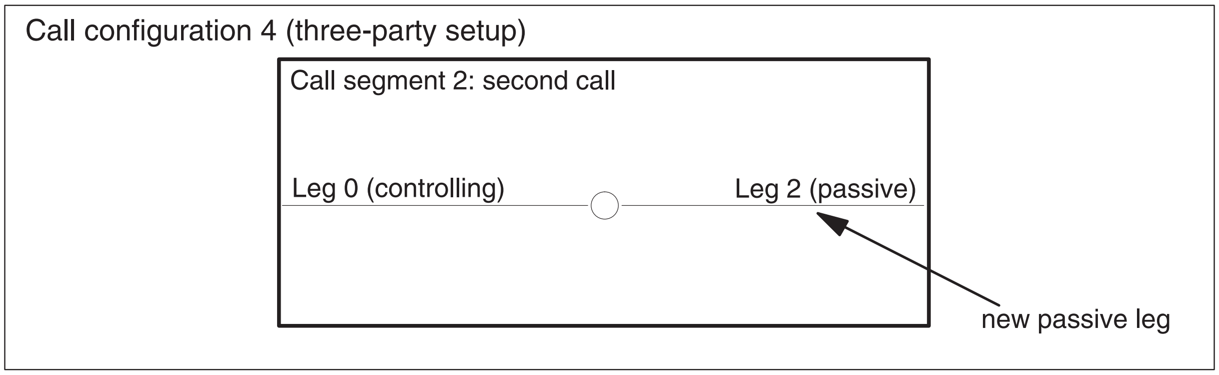

Call configuration 2 (stable two-party call)
Call segment 1: original call
Leg 0
(original controlling leg)
Leg 1
(original passive leg)
Call configuration 4 (three-party setup)
original controlling leg
reassigned as a passive leg
Call segment 1: original call on hold
Leg 0 (controlling)
Leg 1 (passive)

Figure 4-13
New passive leg


Call configuration 4 (three-party setup)
Call segment 2: second call
Leg 0 (controlling)
Leg 2 (passive)
new passive leg

Call configuration 4
Three-party setup (continued)

Transitions

The call can transition from CC4 to four call configurations:

- CC0 (null)
- CC2 (stable two-party call)
- CC5 (three-party setup complement)
- CC6 (party on hold)

Transitioning to CC0 (null)

Table 4-9 describes the user's and the switch's actions that cause the call to transition from CC4 to CC0.

Table 4-9
Transitioning to CC0 (null)

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **O_Abandon** EDP-request message.	a **Disconnect** SCP response message.
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **O_Abandon** EDP-request message.	a **Connect_To_Resource** in a response package for the originating party (leg 1).
not applicable	sends a **Failure_Outcome** message in response to an **Originate_Call** SCP response message. (See note.)	a **Disconnect** SCP response message.
The terminating party (leg 0) enters digits as instructed by a resource.	sends a **CTR_Clear** message.	a **Disconnect** SCP response message.

Note: The switch sends the **Failure_Outcome** message when all routes are busy or Leg 2 (the third party) does not answer.

Call configuration 4
Three-party setup (continued)

Transitioning to CC2 (stable two-party call)

Table 4-10 describes the user's and the switch's actions that cause the call to transition from CC4 to CC2.

Table 4-10
Transitioning to CC2

The user's actions	The switch	Then the switch receives
not applicable	sends a `Failure_Outcome` message in response to an `Originate_Call` SCP response message.	a `Merge_Call` SCP response message.
Note: The switch sends the `Failure_Outcome` message when all routes are busy or Leg 2 (the third party) does not answer.		

Transitioning to CC5 (three-party setup complement)

Table 4-11 describes the user's and the switch's actions that cause the call to transition from CC4 to CC5.

Table 4-11
Transitioning to CC5

The user's actions	The switch	Then the switch receives
The originating party (leg 1) disconnects.	detects the *O_Disconnect* event for the originating party (leg 1).	not applicable

Call configuration 4
Three-party setup (end)

Transitioning to CC6 (party on hold)

Table 4-12 describes the user's and the switch's actions that cause the call to transition from CC4 to CC6.

Table 4-12
Transitioning to CC6

The user's actions	The switch	Then the switch receives
not applicable	selects a route and is about to seize a terminating trunk.	not applicable

Call configuration 5
Three-party setup complement (continued)

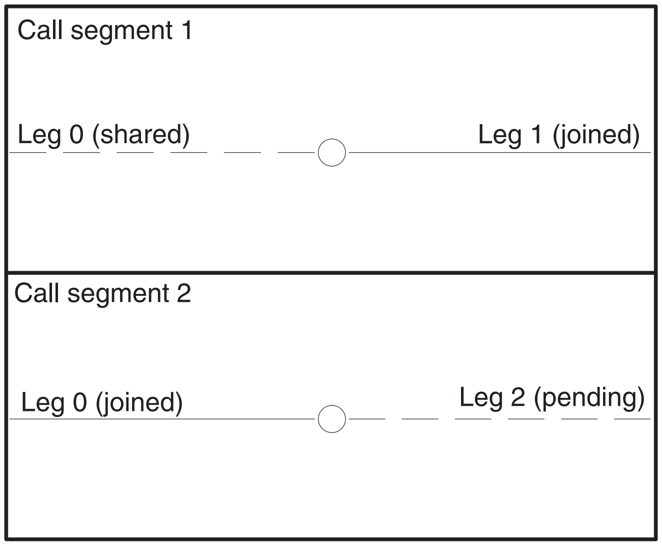
CC5 (three-party setup complement) represents a state where the originating party (leg 1) is on hold while the terminating party (leg 0) is attempting to connect to a third party (leg 2). During CC4, the originating party (leg 1) caused an event to occur. This event caused the call to transition into CC5. The switch detected the event and, during CC5, is about to act on the event.

Call segment 1 is above call segment 2 in this call configuration diagram. This means the switch can only process events in call segment 1.

In CC5, the originating party is a passive leg (leg 1), the terminating party is the controlling leg (leg 0), and the third party is a passive leg (leg 2).

Refer to Figure 4-14.

Figure 4-14
CC5 (three-party setup complement)



Call configuration 5
Three-party setup complement (end)

Transitions

The call can transition from CC5 to two call configurations:

- CC0 (null)
- CC1 (originating two-party setup)

Transitioning to CC0 (null)

Table 4-13 shows the user's and switch's actions that cause the call to transition from CC5 to CC0.

Table 4-13
Transitioning to CC0

The user's action	The switch	Then the switch receives
The originating party (leg 1) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1.	a **`Disconnect`** SCP response message.

Transitioning to CC1 (originating two-party setup)

Table 4-14 shows the user's and switch's actions that cause the call to transition from CC5 to CC1.

Table 4-14
Transitioning to CC1

The user's actions	The switch	Then the switch receives
The originating party (leg 1) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1.	a **`Disconnect_Leg`** SCP response for the originating party (leg 1).

Call configuration 6
Party on hold (continued)

Call configuration 6 (CC6), party on hold, represents a state where the terminating party (leg 0) has the originating party (leg 1) on hold and has established a stable connection with the third party (leg 2).

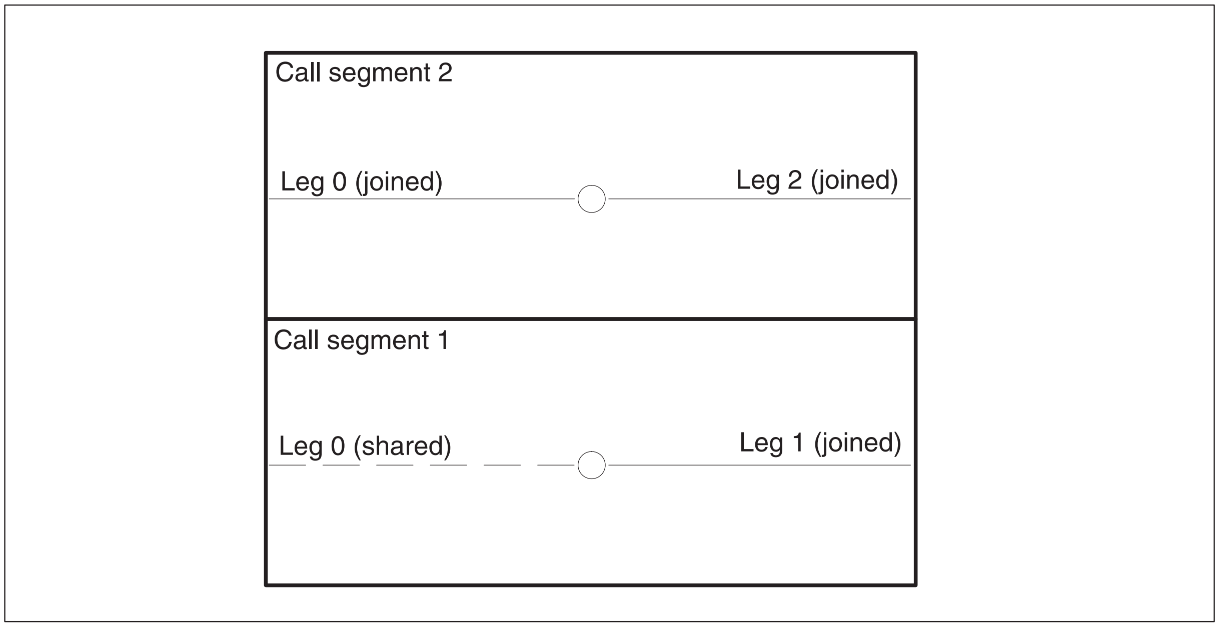
Note: A stable connection occurs when the switch selects a route and is about to seize the terminating trunk.

Call segment 2 is above call segment 1 in this call configuration diagram. This means the switch can only process events in call segment 2.

In CC6, the terminating party is the controlling leg (leg 0), the originating party is a passive leg (leg 1), and the third party is a passive leg (leg 2).

Refer to Figure 4-15.

Figure 4-15
CC6 (party on hold)



Transitions

The call can transition from CC6 to six call configurations:

- CC0 (null)
- CC2 (originating two-party setup)
- CC6 subset

- CC7 (party on hold complement)
- CC10 (stable multi-party call)
- CC11 (transfer)

Transitioning to CC0

Table 4-15 shows the user's and switch's actions that cause the call to transition from CC6 to CC0.

Table 4-15
Transitioning to CC0

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Disconnect`** SCP response message.
During a **`Connect_To_Resource`** interaction, the originating or third party (leg 0 or 2) disconnects.	sends a **`CTR_Clear`** message for leg 0 or 2, respectively, with a ***`ClearCause`*** parameter value of `userAbandon`.	a **`Disconnect`** SCP response message.
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **`O_Abandon`** EDP-request message.	a **`Disconnect`** SCP response message.
The terminating party (leg 0) or third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 0 or 2, respectively.	a **`Disconnect`** SCP response message.
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **`O_Abandon`** EDP-request message.	a **`Connect_To_Resource`** SCP response message for leg 1.
The terminating party (leg 0) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 0.	a **`Connect_To_Resource`** SCP response message for leg 1.

Call configuration 6
Party on hold (continued)

Transitioning to CC2

Table 4-16 shows the user's and switch's actions that cause the call to transition from CC6 to CC2.

Table 4-16
Transitioning to CC2

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Disconnect_Leg`** SCP response message for leg 1 or 2.
The terminating party (leg 0) hears a resource playing and follows the resources instructions.	sends a **`CTR_Clear`** for leg 0.	a **`Disconnect_Leg`** SCP response message for leg 1 or 2.
The third party (leg 2) hears a resource playing and follows the resources instructions.	sends a **`CTR_Clear`** for leg 2.	a **`Disconnect_Leg`** SCP response message for leg 1 or 2.
The third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 2.	a **`Merge_Call`** SCP response message.
The third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 2.	a **`Disconnect_Leg`** SCP response message for leg 2.

Call configuration 6
Party on hold (continued)

Transitioning to CC6 subset

Table 4-17 shows the user's and switch's actions that cause the call to transition from CC6 to CC6 subset.

Table 4-17
Transitioning to CC6 subset

The user's action	The switch	Then the switch receives
The third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 2.	a **`Connect_To_Resource`** SCP response message for leg 0.

Transitioning to CC7

Table 4-18 shows the user's and switch's actions that cause the call to transition from CC6 to CC7.

Table 4-18
Transitioning to CC7

The user's action	The switch	Then the switch receives
The originating party (leg 1) disconnects.	detects the *O_Disconnect* event.	not applicable

Call configuration 6
Party on hold (continued)

Transitioning to CC10

Table 4-19 shows the user's and switch's actions that cause the call to transition from CC6 to CC10.

Table 4-19
Transitioning to CC10

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Merge_Call`** SCP response message.
The terminating party (leg 0) or the third party (leg 2) hears a resource playing and follows the resource's instructions.	sends a **`CTR_Clear`** message for leg 0 or 2.	a **`Merge_Call`** SCP response message.

Transitioning to CC11

Table 4-20 shows the user's and switch's actions that cause the call to transition from CC6 to CC11.

Table 4-20
Transitioning to CC11

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **`O_Abandon`** EDP-request message.	a **`Merge_Call`** SCP response message.
The terminating party (leg 0) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 0.	a **`Merge_Call`** SCP response message.
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Disconnect_Leg`** SCP response message is received for leg 0 .

—continued—

Table 4-20
Transitioning to CC11 (continued)

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) or third party (leg 2) hears a resource playing and follows the resource's instructions.	sends a **CTR_Clear** message for leg 0 or 2.	a **Disconnect_Leg** SCP response message is received for leg 0.
The terminating party (leg 0) disconnects.	sends an **O_Disconnect** EDP-request message for leg 0.	a **Disconnect_Leg** SCP response message is received for leg 0.
The terminating party (leg 0) disconnects before the third party (leg 2) answers.	sends an **O_Abandon** EDP-request message.	a **Disconnect_Leg** SCP response message is received for leg 0.

—end—

Remaining in CC6

Table 4-21 shows the user's and switch's actions that cause the call to remain in CC6 and not transition to another call configuration.

Table 4-21
Remaining in CC6

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **O_Mid_Call** EDP-request message for the *Switch_Hook_Flash* event	a **Connect_To_Resource** SCP response message for leg 0 or 2
The terminating party (leg 0) or third party (leg 2) hears a resource playing and follows the resource's instructions.	sends a **CTR_Clear** message for leg 0 or 2.	a **Connect_To_Resource** SCP response message for leg 0 or 2

—continued—

Call configuration 6
Party on hold (end)

Table 4-21
Remaining in CC6 (continued)

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event	an **`Acknowledge`** SCP response message
The terminating party (leg 0) or third party (leg 2) hears a resource playing and follows the resource's instructions.	sends a **`CTR_Clear`** message for leg 0 or 2.	an **`Acknowledge`** SCP

—end—

Call configuration 6 subset
CC6 subset (continued)

Call configuration 6 (CC6) subset represents a state where the switch has connected the terminating party (leg 0) to a resource after third party (leg 2) has disconnected.

Transitions to other call configurations

CC6 subset can transition to two call configurations:

- CC0 (null)
- CC2 (stable two-party call)

Transitioning to CC0

Table 4-22 shows the user's and switch's actions that cause the call to transition from CC6 subset to CC0.

Table 4-22
Transitioning to CC0

The user's action	The switch	Then the switch receives
The terminating party (leg 0) hears a resource playing.	sends a **`CTR_Clear`** message for leg 0.	a **`Disconnect`** SCP response message.

Transitioning to CC2

Table 4-23 shows the user's and switch's actions that cause the call to transition from CC6 subset to CC2.

Table 4-23
Transitioning to CC2

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) hears a resource playing.	sends a **`CTR_Clear`** message for leg 0.	a **`Merge_Call`** SCP response message.
The terminating party (leg 0) hears a resource playing.	sends a **`CTR_Clear`** message for leg 0.	a **`Disconnect_Leg`** SCP response message is received for leg 2.

Call configuration 6 subset
CC6 subset (end)

Remaining in CC6 subset

Table 4-24 shows the user's and switch's actions that cause the call to remain in CC6 subset and not transition to another call configuration.

Table 4-24
Remaining in CC6 subset

The user's action	The switch	Then the switch receives
A party hears a resource playing.	sends a **`CTR_Clear`** message.	a **`Connect_To_Resource`** SCP response message.

Call configuration 7
Party on hold complement (continued)

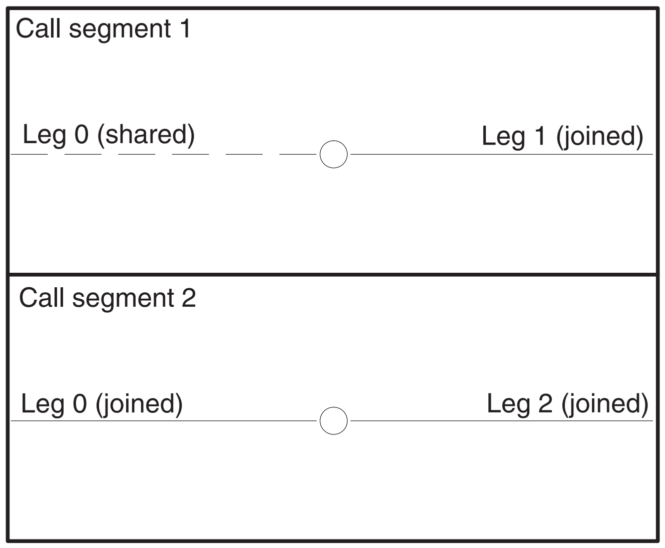
CC7 (party on hold complement) represents a state where the originating party (leg 1) is on hold while the terminating party (leg 0) has a stable connection with the third party (leg 2). During CC6, the switch detects an event for the party on hold (leg 1) and , during CC7, is about to act on this event.

Call segment 1 is above call segment 2 in this call configuration diagram. This means the switch can only process events in call segment 1.

In CC7, the terminating party is the controlling leg (leg 0), the originating party is a passive leg (leg 1), and the third party is a passive leg (leg 2).

Refer to Figure 4-16.

Figure 4-16
CC7 (party on hold complement)



Call configuration 7
Party on hold complement (end)

Transitions

CC7 can transition to two call configurations:

- CC0 (null)
- CC2 (stable two-party call)

Transitioning to CC0

Table 4-25 shows the user's and switch's actions that cause the call to transition from CC7 to CC0.

Table 4-25
Transitioning to CC0

The user's action	The switch	Then the switch receives
The originating party (leg 1) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1.	a **`Disconnect`** SCP response message.

Transitioning to CC2

Table 4-26 shows the user's and switch's actions that cause the call to transition from CC7 to CC2.

Table 4-26
Transitioning to CC2

The user's action	The switch	Then the switch receives
The originating party (leg 1) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1.	a **`Disconnect_Leg`** SCP response message for leg 1.

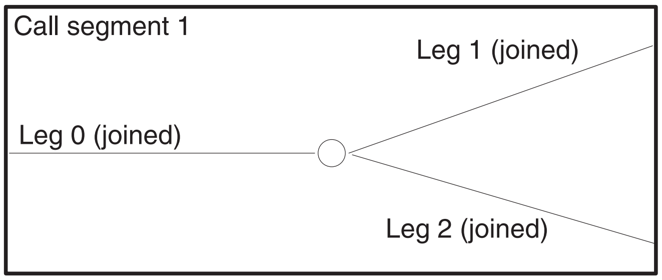
Call configuration 10
Stable multi-party call (continued)

Call configurations 10 (CC10), stable multi-party call, represents a state where the switch has connected the three parties in a multi-party call.

In CC10, the terminating party is the controlling leg (leg 0), the originating party is a passive leg (leg 1), and the third party is a passive leg (leg 2).

Refer to Figure 4-17.

Figure 4-17
CC10 (stable multi-party call)



Transitions to other call configurations

CC10 can transition to four call configurations:

- CC0 (null)
- CC2 (stable two-party call)
- CC10 subset
- CC11 (transfer)

Call configuration 10
Stable multi-party call (continued)

Transitioning to CC0

Table 4-27 shows the user's and switch's actions that cause the call to transition from CC10 to CC0.

Table 4-27
Transitioning to CC0

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) disconnects.	sends an `O_Disconnect` EDP-request message for leg 0.	a `Disconnect` SCP response message.
The terminating party (leg 0) disconnects.	sends an `O_Disconnect` EDP-request message for leg 0.	a `Connect_To_Resource` SCP response message.
The terminating party (leg 0) presses the star (*) key.	sends an `O_Mid_Call` EDP-request message for the *Switch_Hook_Flash* event.	a `Disconnect` SCP response message.

Transitioning to CC2

Table 4-28 shows the user's and switch's actions that cause the call to transition from CC10 to CC2.

Table 4-28
Transitioning to CC2

The user's actions	The switch	Then the switch
The third party (leg 2) disconnects.	sends an `O_Disconnect` EDP-request message for leg 2.	a `Disconnect_Leg` SCP response message for leg 2.
The originating party (leg 1) disconnects.	sends an `O_Disconnect` EDP-request message for leg 1.	a `Disconnect_Leg` SCP response message for leg 1.

—continued—

Call configuration 10
Stable multi-party call (continued)

Table 4-28
Transitioning to CC2 (continued)

The user's actions	The switch	Then the switch
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	**`Disconnect_Leg`** SCP response message for leg 1.
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Disconnect_Leg`** SCP response message for leg 2.
—end—		

Transitioning to CC10 subset

Table 4-29 shows the user's and switch's actions that cause the call to transition from CC10 to CC10 subset.

Table 4-29
Transitioning to CC10 subset

The user's action	The switch	Then the switch receives
The originating party (leg 1) or the third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1 or 2.	a **`Connect_To_Resource`** SCP response message.

Call configuration 10
Stable multi-party call (end)

Transitioning to CC11

Table 4-30 shows the user's and switch's actions that cause the call to transition from CC10 to CC11.

Table 4-30
Transitioning to CC11

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 0.	a **`Disconnect_Leg`** SCP response message for leg 0.
The terminating party (leg 0) presses the star (*) key.	sends an **`O_Mid_Call`** EDP-request message for the *Switch_Hook_Flash* event.	a **`Disconnect_Leg`** SCP response message for leg 0.

Call configuration 10 subset
CC10 subset (continued)

CC10 subset represents a state where the switch has connected the terminating party (leg 0) and the remaining party (leg 1 or 2) to a resource after either the originating party (leg 1) or the third party (leg 2) disconnected.

Transitions

CC10 subset can transition to two call configurations:

- CC0 (null)
- CC2 (stable two-party call)

Transitioning to CC0

Table 4-31 shows the user's and switch's actions that cause the call to transition from CC10 subset to CC0.

Table 4-31
Transitioning to CC0

The user's action	The switch	Then the switch receives
One of the two remaining parties hears a resource playing.	sends a **`CTR_Clear`** message for the party who hears the resource (leg 0, 1, or 2).	a **`Disconnect`** SCP response message.

Call configuration 10 subset
CC10 subset (end)

Transitioning to CC2

Table 4-32 shows the user's and switch's actions that cause the call to transition from CC10 subset to CC2.

Table 4-32
Transitioning to CC2

The user's actions	The switch	Then the switch receives
The terminating party (leg 0) or third party (leg 2) hears a resource playing and follows the resource's instructions.	sends a **`CTR_Clear`** message for leg 0 or 2.	a **`Disconnect_Leg`** SCP response message for leg 1.
The terminating party (leg 0) or originating party (leg 1) hears a resource playing and follows the resource's instructions.	sends a **`CTR_Clear`** message for leg 0 or 1.	a **`Disconnect_Leg`** SCP response message for leg 2.

Remaining in CC10 subset

Table 4-33 shows the switch's actions that cause the call to remain in CC10 subset and not transition to another call configuration.

Table 4-33
Remaining in CC10 subset

The user's action	The switch	Then the switch receives
One of the two remaining parties hears a resource playing.	sends a **`CTR_Clear`** for the party who hears the resource (leg 0, 1, or 2).	a **`Connect_To_Resource`** SCP response message.

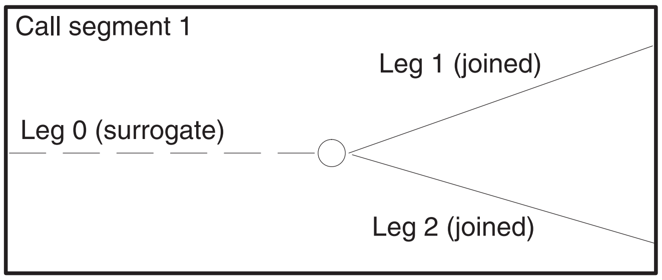
Call configuration 11
Transfer (continued)

CC11 (transfer) represents a key state where the originating party (leg 1) and third party (leg 2) remain in a stable connection after the terminating party (leg 0) disconnected from the call. The terminating party's leg is a surrogate leg, which means the terminating party still has a billing relationship between the remaining parties.

In CC11, the terminating party is the controlling leg (leg 0), the originating party is a passive leg (leg 1), and the third party is a passive leg (leg 2).

Refer to Figure 4-18.

Figure 4-18
CC11 (transfer)



Call configuration 11
Transfer (end)

Transitions

CC11 can transition to CC0 (null).

Transitioning to CC0

Table 4-34 describes the user's and the switch's actions that cause the call to transition from CC11 to CC0.

Table 4-34
Transitioning to CC0

The user's actions	The switch	Then the switch receives
The originating party (leg 1) or third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1 or 2.	a **`Disconnect`** SCP response message.
The originating party (leg 1) or third party (leg 2) disconnects.	sends an **`O_Disconnect`** EDP-request message for leg 1 or 2.	a **`Connect_To_Resource`** SCP response message in a response package.

Call configurations
Example of a transfer call (continued)

The example in this section shows:

- how a terminating subscriber can use the N00 take-back and transfer service.
- how a transfer call transitions through the call configurations.

The example describes what the terminating subscriber and the switch do to cause the call to transition through the call configurations. Each step number and letter identifies a user's action or the switch's action that causes the call to transition.

The following example consists of seven steps and describes a transfer call originating from the null state, transitioning through the call configurations to the transfer state, and returning to the null state after the call's parties disconnect.

1 A caller picks up the phone and dials 1-800-123–4567.

   Then, the switch

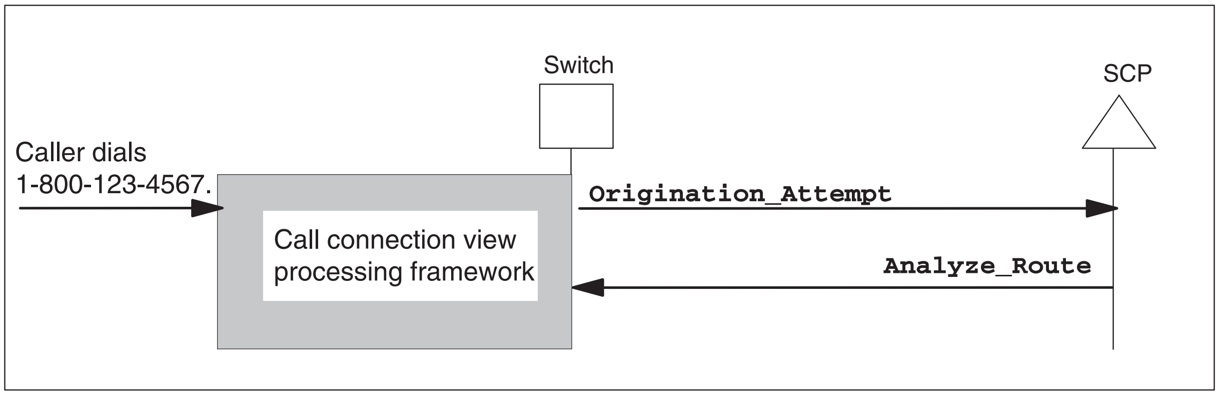
   a. detects an off-hook state (encounters the *Offhook_Delay* trigger).

   b. sends an **`Origination_Attempt`** TDP-Request message to the SCP.

   Refer to Figure 4-19.

   The call transitions from CC0 (null) to CC1 (originating two-party setup).

Figure 4-19
Message flow due to Step 1


Switch
SCP
Caller dials 1-800-123-4567.
Origination_Attempt
Call connection view processing framework
Analyze_Route

Call configurations
Example of a transfer call (continued)

2 The switch selects a route and is about to seize a terminating trunk.

The call transitions to CC2 (stable two-party call).

A receptionist answers the call. The caller asks the receptionist to transfer the call to a third party.

3 The receptionist presses the star (*) key. The receptionist hears a resource playing. The receptionist enters the third party's phone number.

Then, the switch

a. sends an **`O_Mid_Call`** EDP-Request message for the SwitchHookFlash event

b. receives a **`Connect_To_Resource`** message and plays a resource to the subscriber

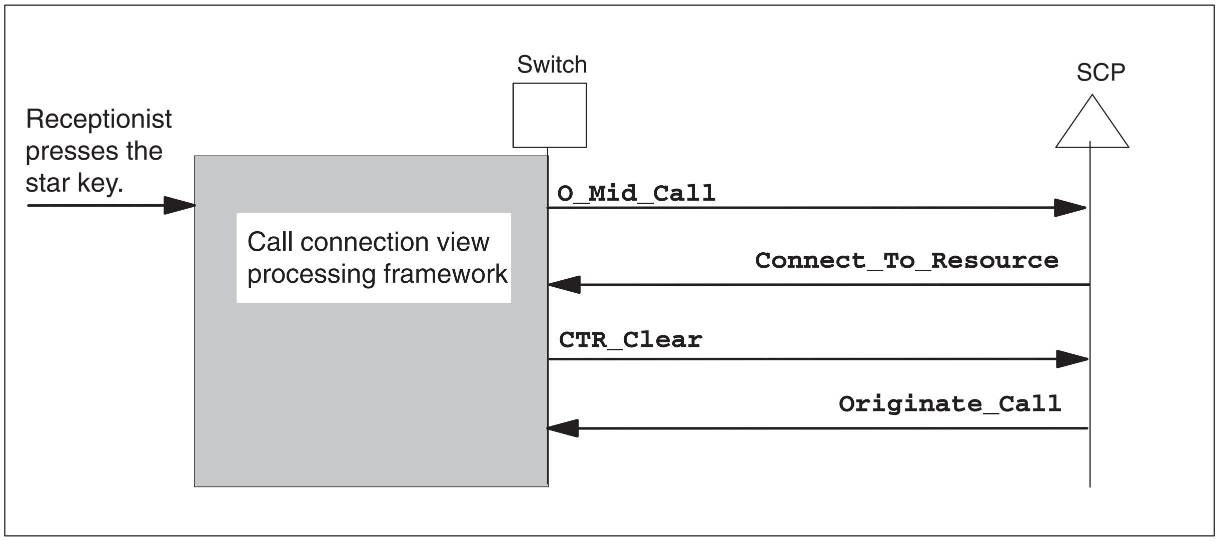
c. sends a **`CTR_Clear`** message to the SCP. The **`CTR_Clear`** message contains the digits the subscriber entered

d. receives an **`Originate_Call`** message

Refer to Figure 4-20.

The call transitions to CC4 (three-party setup).

Figure 4-20
Message flow due to step 3


Switch
SCP
Receptionist presses the star key.
Call connection view processing framework
O_Mid_Call
Connect_To_Resource
CTR_Clear
Originate_Call

Call configurations
Example of a transfer call (continued)

4 The switch selects a route and is about to seize a terminating trunk for the third call.

The call transitions to CC6 (Party on hold). (The party on hold is the originating caller from step 1.)

The third party answers the call.

5 The receptionist presses the star (*) key again.

Then, the switch

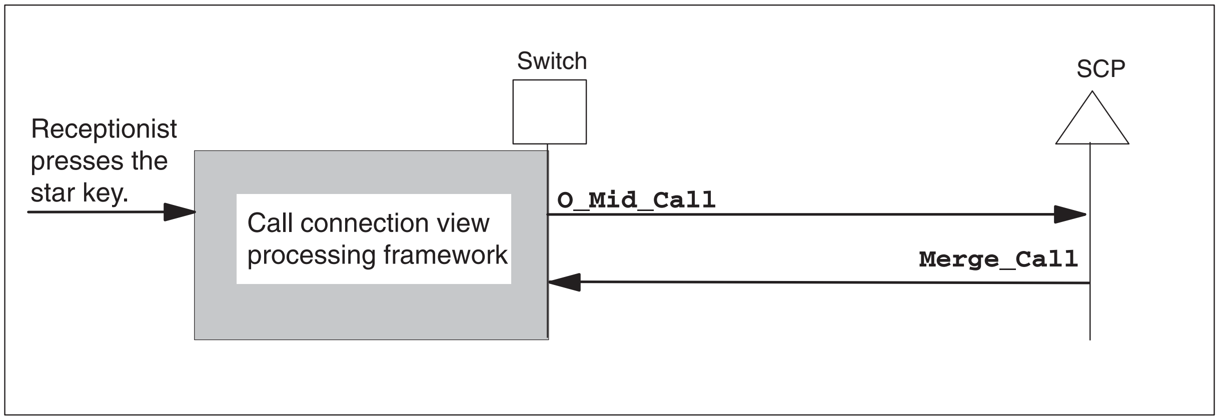
a. sends an **`O_Mid_Call`** EDP-Request message for the SwitchHookFlash event

b. receives a **`Merge_Call`** message

Refer to Figure 4-21.

The call transitions to CC10 (stable multi-party call).

Figure 4-21
Message flow in step 5


Switch
SCP
Receptionist presses the star key.
Call connection view processing framework
O_Mid_Call
Merge_Call

Call configurations
Example of a transfer call (continued)

6 The receptionist disconnects. The original caller and the third party remain on the line.

Then, the switch

a. detects the *O_Disconnect* event

b. sends the **`O_Disconnect`** EDP-Request message

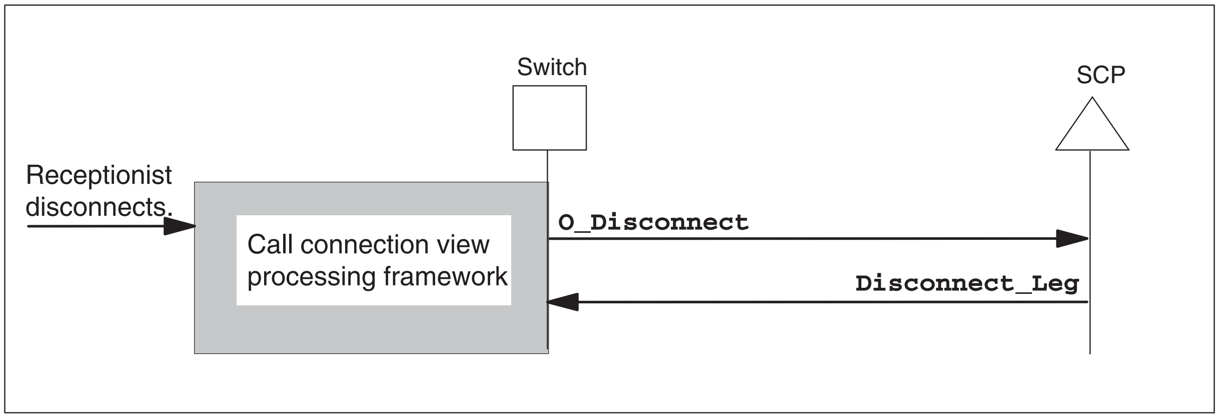
c. receives the **`Disconnect_Leg`** message for leg 0

d. disconnects the controlling leg

Refer to Figure 4-22.

The call transitions to CC11 (transfer).

Figure 4-22
Message flow in step 6


Switch
SCP
Receptionist disconnects.
Call connection view processing framework
O_Disconnect
Disconnect_Leg

Call configurations
Example of a transfer call (end)

7 The original caller and the third party finish their conversation and disconnect.

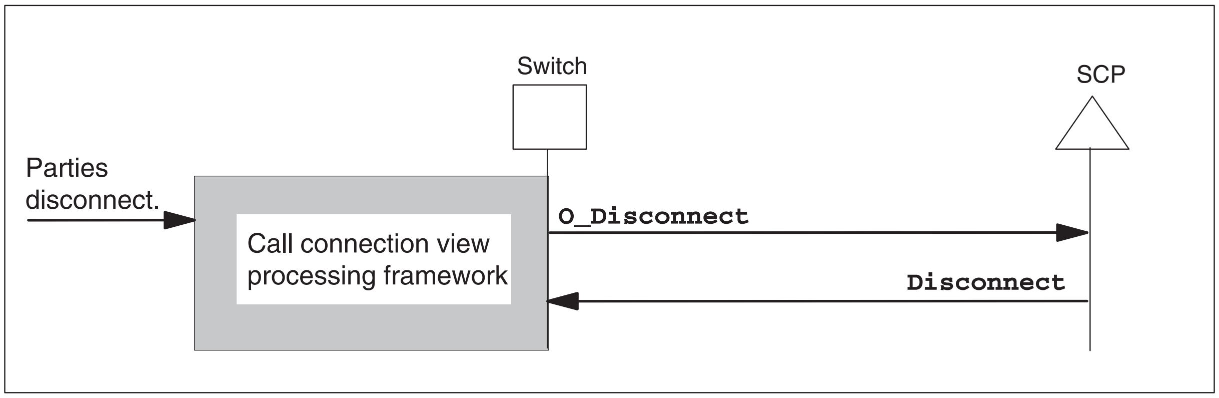
The, the switch

a. detects the *O_Disconnect* event
b. sends the **`O_Disconnect`** EDP-Request message
c. receives the **`Disconnect`** message
d. disconnects legs 1 and 2

Refer to Figure 4-23.

The call returns to the null call configuration.

Figure 4-23
Message flow due to connection view processing



Call configurations
Error messages (end)

Fatal application errors

Table 4-35 provides the fatal application errors that the switch may detect during CC3–11 processing.

Table 4-35
CC3–11 fatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Call processing attempts to close transaction during CC 3-11 (Note 1)	N/A	CAIN200	No	The switch disconnects all parties and sends the orignator to AINF treatment.
Transaction closed in inappropriate CC (Note 2)	N/A	CAIN200	Yes	**`Close`** sent with a ***`CloseCause`*** of `unexpectedCommunication`

Note 1: This fatal error occurs when another error is detected during CC3–11. If NetworkBuilder call processing directs the switch to send a **`Close`** message while the call is in CC3–11, this fatal application error is detected. The switch detects this error only during Takeback and Transfer processing.

Note 2: Whenever the call is in CC3–11, inclusive, the SCP must maintain a conversation transaction with the switch by requesting one or more detectable events. This error occurs when the SCP does not request events to allow the conversation transaction to be maintained in one of these CCs.

Terminology

To simplify the discussion of call configurations, the quick-reference uses the following terminology:

- the party who originated the first two-party call (in CC1) is the "originating party"
- the party to which the first two-party call terminated (in CC1) is the "terminating party"
- the party to which the second call terminates (in CC4) is the "third party"

Quick-reference to the call configuration model

Table 4-36 provides the following information:

- the supported call configurations
- a brief description of the call configuration
- an explanation at what point a call enters the call configuration

Table 4-36
Call configurations quick-reference

Call configurations	Means	Begins when
CC0 (Null)	Call processing is inactive.	the switch releases a call.
CC1 (Originating Setup)	A caller is originating a two-party call.	the switch detects an off-hook state (encounters the ***Origination_Attempt*** TDP).
CC2 (Stable two-party)	A two-party call is in a stable phase.	the switch selects a route and is about to to seize the terminating trunk.
CC4 (Three-party setup)	The terminating party (leg 0) placed the originating party (leg 1) on hold and is originating a call to a third party (leg 2).	the switch receives an **`Originate_Call`** message.

Note 1: RLT operations are not supported when a call is in CC4, CC5, CC6, CC7, CC10, or CC11.
Note 2: NetworkBuilder does not support CC3, CC8, CC9, and CC12.

—continued—

Call configuration model
Quick-references (continued)

Table 4-36
Call configurations quick-reference (continued)

Call configurations	Means	Begins when
CC5 (Three-party setup complement)	While in CC4, the originating party (leg 1) performed an action (for example, disconnected). The action caused the switch to detect an event for the originating party (leg 1).	the switch detects an event for the originating party (leg 1) while the call is in CC4.
CC6 (Party on hold)	The terminating party (leg 0) has the originating party (leg 1) on hold and is in a stable phase with the third party (leg 2).	the switch selects a route and is about to to seize the terminating trunk for the third party (leg 2).
CC6 subset (Subset of party on hold)	The third party (leg 2) disconnected. The switch connects the terminating party (leg 0) to a resource.	the switch receives a **`Connect_To_Resource`** message for the terminating party (leg 0).
CC7 (Party on hold complement)	While in CC6, the originating party (leg 1) performed an action (for example, disconnected). The action caused the switch to detect an event for the party on hold.	the switch detects an event for the originating party (leg 1) while the call is in CC6.
CC10 (Stable multi-party call)	A multi-party call is in a stable phase. The three parties (legs 0, 1, and 2) are connected in a conference call.	the switch receives a **`Merge_Call`** message while three legs are present in the call.

Note 1: RLT operations are not supported when a call is in CC4, CC5, CC6, CC7, CC10, or CC11.
Note 2: NetworkBuilder does not support CC3, CC8, CC9, and CC12.

—continued—

Call configuration model
Quick-references (continued)

Table 4-36
Call configurations quick-reference (continued)

Call configurations	Means	Begins when
CC10 subset (Subset of stable multi-party call)	The originating party (leg 1) or third party (leg 2) disconnected. The switch connects the terminating party (leg 0) and remaining party (leg 1 or 2) to a resource.	the switch receives a **`Connect_To_Resource`** message for the terminating party (leg 0) and remaining party (leg 1 or 2).
CC11 (Transfer)	The originating party (leg 1) and third party (leg 2) are in a stable phase. The terminating party (leg 0) disconnected, but still has a billing relationship between the two remaining parties (legs 1 and 2).	the switch receives a **`Disconnect_Leg`** message for the terminating party (leg 0).
Note 1: RLT operations are not supported when a call is in CC4, CC5, CC6, CC7, CC10, or CC11. ***Note 2:*** NetworkBuilder does not support CC3, CC8, CC9, and CC12.		
—end—		

Call configuration model
Quick-references (continued)

Quick-reference to handling of unarmed events

Table 4-37 provides the following information:

- the call configuration where the switch detected the unarmed event
- the unarmed event
- the call leg on which the switch detected the unarmed event
- the action the switch takes
- the call configuration to which the call transitions

Table 4-37
Switch's action when it detects an unarmed event

CC	Unarmed event	Leg	Switch's action	CC call transitions to
CC4	*O_Abandon*	0	Switch takes the call down.	CC0
CC5	*O_Disconnect*	1	Switch clears the call toward Leg 1.	CC1
CC6	*O_Abandon*	0	Switch takes the call down.	CC0
	O_Disconnect	0	Switch takes the call down.	CC0
	O_Disconnect	1	Switch clears the call toward Leg 1.	CC2
	O_Disconnect	2	Switch clears the call toward Leg 2.	CC2
CC7	*O_Disconnect*	1	Switch clears the call toward Leg 1.	CC2
CC10	*O_Disconnect*	0	Switch takes the call down.	CC0

Note: When a party disconnects from the call and the switch has not armed the event, the switch generates a CAIN907 log.

—continued—

Call configuration model Quick-references (end)

Table 4-37
Switch’s action when it detects an unarmed event (continued)

CC	Unarmed event	Leg	Switch’s action	CC call transitions to
CC10 (continued)	*O_Disconnect*	1	Switch clears the call toward Leg 1.	CC2
	O_Disconnect	2	Switch clears the call toward Leg 2.	CC2
Note: When a party disconnects from the call and the switch has not armed the event, the switch generates a CAIN907 log.				
—end—				

Termination_Notification processing

When the SCP wants to be notified when the call has been released, it sends a response or conversation package to the switch. Once the switch receives this package, the switch examines the package for multiple components. The first component must contain an SCP call-related message (such as **`Analyze_Route`** or **`Continue`**). This message requests that the switch execute a function or task which may modify the state of the call. The second component, if it exists, must contain an SCP non-call related message (such as **`Send_Notification`**). This message requests that the switch execute some functions or tasks, but it cannot modify the state of the call.

After the multiple components (including the **`Send_Notification`** message) are successfully examined, the switch waits until it releases the call. Once the call is released, the switch sends a switch non-call related message (**`Termination_Notification`**) to the SCP. This message corresponds to the SCP non-call related message (**`Send_Notification`**). Refer to Chapter 10, "Incoming CAIN messages," for further information on the **`Send_Notification`** message.

Note: A multiple component package may contain more than two components. This package must contain one SCP call-related component and at least one SCP non-call related message. In the preceding description, to complete the **`Termination_Notification`** functionality, the **`Send_Notification`** message could be contained in any component except the first one.

If the switch receives multiple **`Send_Notification`** requests for the same call, and each request contains an unique TCAP transaction id, the switch sends one **`Termination_Notification`** message for each request with an unique transaction id when an event occurs that releases the call.

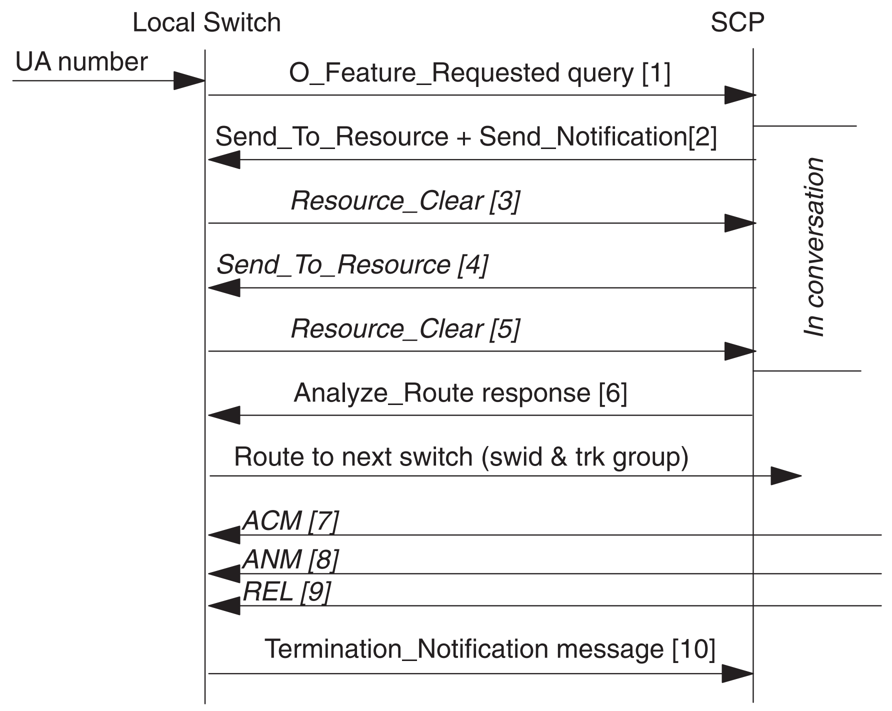
If the switch receives multiple **`Send_Notification`** requests within the same TCAP transaction, the switch only replies to the last **`Send_Notification`** received in the transaction.

Termination_Notification messaging scenarios

Figures 5-1 through 5-4 provide example scenarios which clarify the switch-to-SCP interactions that may occur in calls involving `Termination_Notification` messages. Many other scenarios are possible.

Scenario 1

Figure 5-1
Termination_Notification for scenario 1


Local Switch
SCP
UA number
O_Feature_Requested query [1]
Send_To_Resource + Send_Notification[2]
Resource_Clear [3]
Send_To_Resource [4]
Resource_Clear [5]
In conversation
Analyze_Route response [6]
Route to next switch (swid & trk group)
ACM [7]
ANM [8]
REL [9]
Termination_Notification message [10]

1. The local switch sends an O_Feature_Requested TDP–Request to the SCP.
2. The SCP sends the local switch a conversation package containing an STR component requesting to collect digits followed by a Send_Notification component.
3. The switch sends a Resource_Clear message with the user entered digit to the SCP for review.
4. The SCP sends the local switch another conversation package containing an STR component requesting to collect digits.
5. The local switch sends a Resource_Clear message with the user entered digits to the SCP for review.
6. The SCP provides the following information in a (non–conversational) response package containing an Analyze_Route component only.
7. ACM is received which indicates the called party has been notified. This is sent from the terminating trunk not the SCP.
8. ANM is received which indicates the call has been answered. This is sent from the terminating trunk, not the SCP.
9. REL is received which indicates the call has been released.
10. The local switch sends a Termination_Notification message to the SCP.

The call duration 20 min 10.0 sec is relfected in the ConnectTime parameter

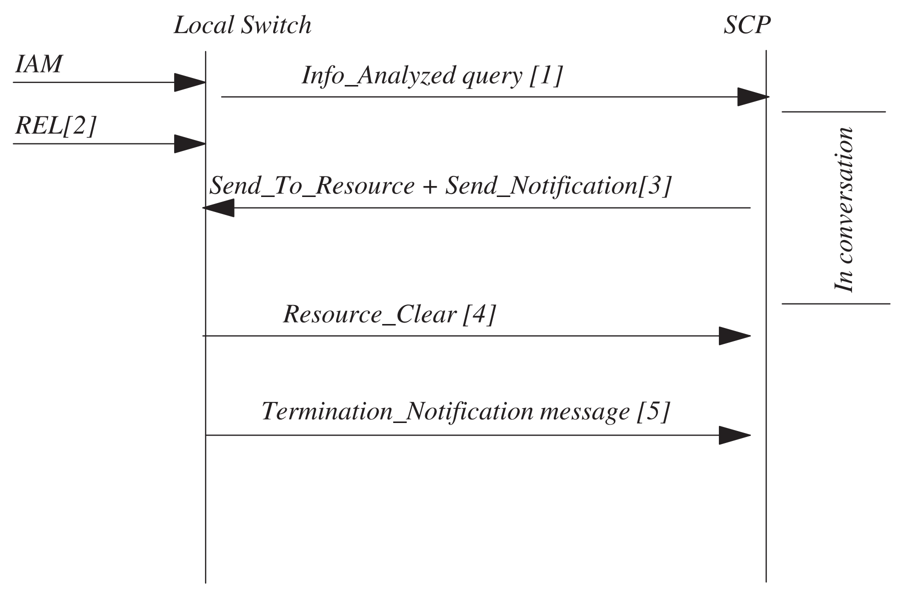
In this example the customer is using a debit card. This customer may dial a Universal Access (UA) number which triggers at *O_Feature_Requested*. The SCP decides it needs additional information to complete the call (the users access code) and sends a **`Send_To_Resource`**. The user complies by entering in their personal access code at which the switch sends a **`Resource_Clear`** message containing the collected digits. The SCP decides it needs additional information once again and sends another **`Send_To_Resource`** with instructions to collect additional digits (the destination address). The user once again enters the needed information and the switch sends a **`Resource_Clear`** message containing the collected digits. The call connects, and the originating trunk waits for the call to be released. Once the call has been released the switch sends the **`Termination_Notification`** message to the SCP.

The **`Termination_Notification`** message would contain:

- the ***`EchoData`*** parameter (containing the same information as the ***`EchoData`*** in the corresponding **`Send_Notification`** message).
- the ***`TerminationIndicator`*** parameter (with the Answer Indication set to YES).
- the ***`ConnectTime`*** parameter (set to 20 min, 10.0 sec).

Scenario 2

Figure 5-2
Termination_Notification for scenario 2



1. The local switch sends an Info_Analyzed TDP–Request to the SCP.
2. The local switch receives a REL message from the originator.
3. The SCP sends the local switch a conversation package containing a Send_To_Resource component requesting to collect digits followed by a Send_Notification component
4. The local switch sends a Resource_Clear message to SCP.
5. The local switch sends a Termination_Notification message to the SCP.

The originating switch sends an **`Info_Analyzed`** query message to the SCP. While the SCP is decoding the query message and encoding a response message, the originating switch receives a REL message (probably because the calling party went on-hook). Meanwhile, the SCP still believes the switch is waiting for a response and sends a **`Send_To_Resource`** message in conversation with an extra **`Send_Notification`** component message. The switch sends a **`Resource_Clear`** message, which alerts the SCP that the call is finished and to close the **`Send_To_Resource`** transaction. The switch sends a **`Termination_Notification`** message.

The **`Termination_Notification`** message would contain:

- the ***`EchoData`*** parameter (containing the same information as the ***`EchoData`*** in the corresponding **`Send_Notification`** message).
- the ***`TerminationIndicator`*** parameter (with the Busy Cause Indication set to YES).

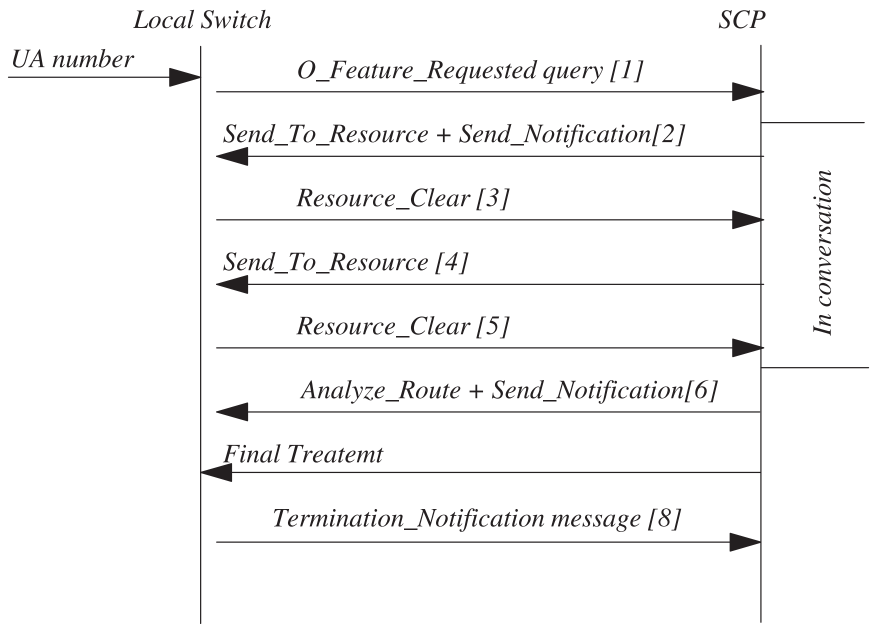
  Note: The Busy Cause Indication is set to YES because the calling party abandons the call before the switch receives the **`Send_Notification`**, yet after the switch sent the switch call-related message (**`Info_Analyzed`**).

- the ***`BusyCause`*** parameter (set to `CallerAbandon`).

  Note: Since the call did not connect, there would not be a ***`ConnectTime`***.

Scenario 3

Figure 5-3
Termination_Notification for scenario 3


Local Switch
SCP
UA number
O_Feature_Requested query [1]
Send_To_Resource + Send_Notification[2]
Resource_Clear [3]
Send_To_Resource [4]
Resource_Clear [5]
In conversation
Analyze_Route + Send_Notification[6]
Final Treatemt
Termination_Notification message [8]

1. The local switch sends an O_Feature_Requested TDP–Request to the SCP.
2. The SCP sends the local switch a conversation package containing an STR component requesting to collect digits followed by a Send_Notification component.
3. The local switch sends a Resource_Clear message with the user entered digits to the SCP for review.
4. The SCP sends the local switch another conversation package containing a Send_To_Resource component requesting to collect digits.
5. The local switch sends a Resource_Clear message with the user entered digits to the SCP for review
6. The SCP provides the following information in a response package containing an Analyze_Route component and a Send_NotifIcation component.
7. The local switch detects a need to perform the AIN Final treatment action.
8. The local switch sends a Termination_Notification message to the SCP.

In this example the customer is using a debit card. This customer may dial a Universal Access (UA) number which triggers at *O_Feature_Requested*. The SCP decides it needs additional information to complete the call (the users access code) and sends a **`Send_To_Resource`**. The user complies by entering in their personal access code at which the switch sends a **`Resource_Clear`** message containing the collected digits. The SCP decide it needs additional information once again and sends another **`Send_To_Resource`** with instructions to collect additional digits (the destination address). The user once again enters the needed information and the switch sends a **`Resource_Clear`** message containing the collected digits. The switch receives the **`Analyze_Route`** message with another **`Send_Notification`**. It determines that the **`Analyze_Route`** message is incomplete (missing a mandatory parameter). The switch invokes AIN Final Treatment but does not discard the rest of the message. The switch sends the **`Termination_Notification`** message to the SCP.

The **`Termination_Notification`** message would contain:

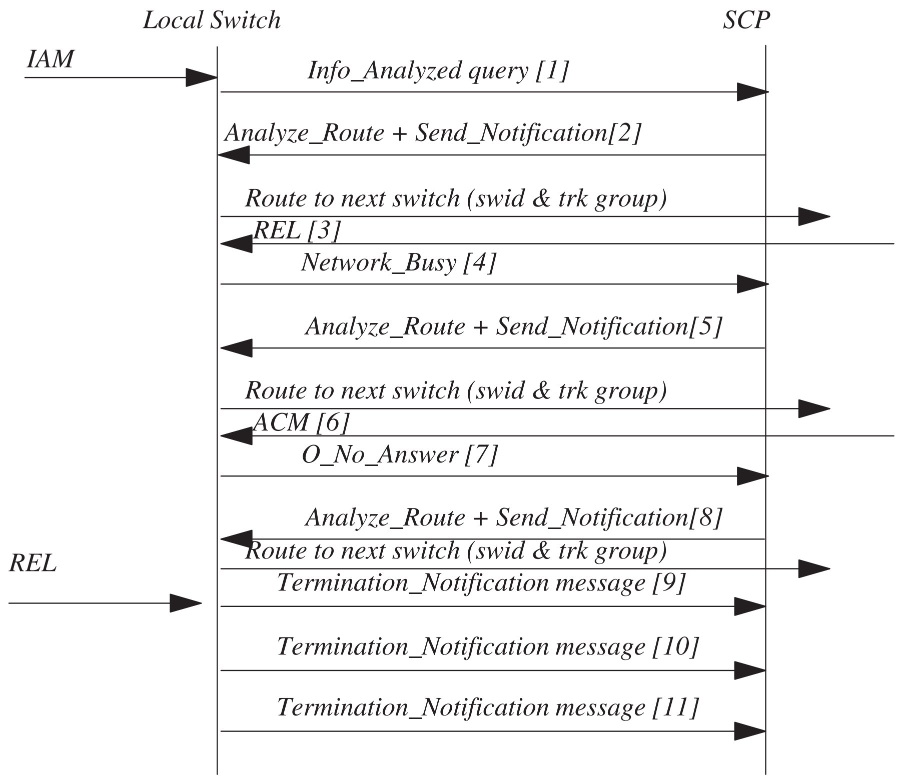
- the ***`EchoData`*** parameter (containing the same information as the ***`EchoData`*** in the last **`Send_Notification`** message).
- the ***`TerminationIndicator`*** parameter (with the Unrelated Error Condition Indication set to YES).

Note 1: The switch responds to the last **`Send_Notification`** message.

Note 2: Since the call did not connect, there would not be a ***`ConnectTime`***.

Scenario 4

Figure 5-4
Termination_Notification for Scenario 4


Local Switch
SCP
IAM
Info_Analyzed query [1]
Analyze_Route + Send_Notification[2]
Route to next switch (swid & trk group)
REL [3]
Network_Busy [4]
Analyze_Route + Send_Notification[5]
Route to next switch (swid & trk group)
ACM [6]
O_No_Answer [7]
Analyze_Route + Send_Notification[8]
Route to next switch (swid & trk group)
REL
Termination_Notification message [9]
Termination_Notification message [10]
Termination_Notification message [11]

1. The local switch sends an Info_Analyzed TDP–Request to the SCP.
2. The SCP sends the local switch an Analyze_Route component followed by a Send_Notification component
3. The local switch receives a REL message (network_busy).
4. The local switch sends a Network_Busy TDP–Request to the SCP.
5. The SCP sends the local switch an Analyze_Route component followed by a Send_Notification component.
6. The local switch receives an ACM message signaling the called party has been notified.
7. The local switch O_No_Answer timer has expired. It sends a O_No_Answer TDP–Request to the SCP.
8. The SCP sends the local switch an Analyze_Route component followed by a Send_Notification component.
9. The local switch sends a Termination_Notification message to the SCP.
10. The local switch sends a Termination_Notification message to the SCP.
11. The local switch sends a Termination_Notification message to the SCP.

The originating switch receives an IAM message. The called number triggers at ***Info_Analyzed***. The SCP sends an **Analyze_Route** message with detailed instructions on routing the call plus a second component containing a **Send_Notification** message. The destination switch is busy and sends back a REL message with a cause value of no circuit available. The originating switch triggers at ***Network_Busy*** and receives another **Analyze_Route** with another **Send_Notification**. This time the destination switch is not busy and sends an ACM message back to the originating switch. The called party has not answered and the ***ONoAnswerTimer*** timer has expired. The originating switch triggers again at ***ONoAnswerTimer***. It does not release the resources until it receives another route. After receiving the final **Analyze_Route** with another **Send_Notification** component the calling party goes on-hook. The switch sends three **Termination_Notification** messages. Each one containing a different ***EchoData*** and ***TerminationIndicator*** value.

The first **Termination_Notification** message would contain:

- the ***EchoData*** parameter (containing the same information as the ***EchoData*** in the first **Send_Notification** message).
- the ***TerminationIndicator*** parameter (with the Exception Indication and Reroute Indication set to YES).

The second **Termination_Notification** message would contain:

- the ***EchoData*** parameter (containing the same information as the ***EchoData*** in the second **Send_Notification** message).
- the ***TerminationIndicator*** parameter (with the Exception Indication and Reroute Indication set to YES).

The third **Termination_Notification** message would contain:

- the ***EchoData*** parameter (containing the same information as the ***EchoData*** in the third **Send_Notification** message).
- the ***TerminationIndicator*** parameter (with the Exception Indication and Reroute Indication set to YES).

Note: Although each **Send_Notification** message contains the same information except for the ***EchoData*** parameter, each **Send_Notification** message could go to a different SCP.

Termination_Notification call processing

Under normal circumstances, **`Termination_Notification`** is enabled when a **`Send_Notification`** component is received with a call-related component in a multiple component conversation package or a response package in response to a TDP– or EDP–Request message. The following conditions would prevent the **`Termination_Notification`** functionality from being successful when a **`Send_Notification`** component is received:

- The CAIN **`Termination_Notification`** Software Optionality Control (SOC) option, CAIN0609, is idle. This is treated as a non-fatal Unexpected Communication application error. A CAIN102 log is generated. The call-related component is processed normally. Refer to the *UCS DMS-250 Commands Reference Manual* for more information on SOC option CAIN0609.
- A **`Send_Notification`** component message is received in response to an LNP query
- The **`Send_Notification`** component is received as the first component in the package, or the first component is not call-related. This is a fatal application error.
- In the case of SS7 RLT (Release Link Trunk) **`Termination_Notification`** functionality can only occur on the first call leg. All information pertinent to completing **`Termination_Notification`** functionality will be kept intact.
- The **`Send_Notification`** component is received and there are no extension blocks present. This is treated as a nonfatal application error. A CAIN102 log is generated and this component is ignored.

If none of the above conditions apply, the switch processes the call-related component and the non-call related **`Send_Notification`** component.

Restrictions and limitations

`Termination_Notification` functionality must be activated by its SOC option, CAIN0609. If a **`Send_Notification`** component is received while this option is off, this is treated as a nonfatal Unexpected Communication application error. A CAIN102 log is generated and a **`Close`** message is sent to the SCP with ***`CloseCause`*** parameter value `unexpectedCommunication`. The call-related component is processed normally.

Termination_Notification message parameter ***TerminationIndicator*** can have one and only one of the following fields set to YES:

- AnswerIndication
- BusyCauseIndication
  - — If the BusyCauseIndication is set to YES then the ***BusyCause*** parameter must be present and indicate the reason.
- UnrelatedErrorConditionIndication
- ExceptionIndication

If the switch receives multiple **Send_Notification** request several times for the same call within the same TCAP transaction, the switch only replies to the last **Send_Notification** received in the transaction.

If the switch receives multiple **Send_Notification** requests for the same call and each request contains an unique TCAP transaction id, the switch sends one **Termination_Notification** message for each request with an unique transaction id when an event occurs that causes the call to exit to the **O_Null** PIC.

The **Send_Notification** message must be sent in a multiple component package and cannot be the first component in the package. The first component is reserved for a call-related messages.

Termination_Notification functionality is not supported in a response to an LNP query. If this occurs, then the **Send_Notification** component message received by the switch is ignored and no other action besides producing a nonfatal application error is taken.

The number of **Send_Notification** messages the switch can receive is controlled by the NUM_SN_EXT_BLOCKS in table CAINPARM.

Outgoing CAIN messages

ATTENTION
Outgoing CAIN messages and extension parameters require various SOC options in order to function. Refer to Volume 5, “NetworkBuilder tools,” for more information.

Note: Outgoing messages are handled differently for LNP and AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide and UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Outgoing CAIN messages

Once trigger criteria is met or an active EDP is encountered, and extension blocks are allocated, the appropriate TCAP outgoing message is built and sent to the SCP. Tables 6-1 and 6-2 list the supported outgoing messages. These messages can be a response to a SCP message (as in the case of Resource_Clear).

Table 6-1
Outgoing call-related CAIN messages

Message	Description
`Call_Info_From_Resource`	The switch sends this message to the SCP during an active IP connection. The message contains intermediate data received from the IP. Refer to Volume 4, “Conversational processing,” for more information.
`CTR_Clear`	This message is sent in response to conversational digit collection initiated by the SCP. This message also contains any digits collected as requested by the SCP. This message is used in response to a `Connect_To_Resource` message. Refer to Volume 4, “Conversational processing,” for more information.
—continued—	

Table 6-1
Outgoing call-related CAIN messages (continued)

Message	Description
`Failure_Outcome`	This message informs the SCP that a failure event was detected in processing a **`Disconnect_Leg`**, **`Merge_Call`**, or **`Originate_Call`** message. Refer to Volume 3, Chapter 3 “Event processing.” for more information.
`Info_Analyzed`	This message requests data from the SCP to direct call processing. This message is sent when a CAIN call meets the trigger criteria for a *Specific_Feature_Code*, *Customized_Dialing_Plan*, *Specific_Digit_String*, or *Office_Code* triggers. Refer to Volume 1, Chapter 5, “Analyze_Information PIC,” for details.
`Info_Collected`	This message requests data from the SCP to direct call processing. This message is sent when a CAIN call meets the trigger criteria for an *Offhook_Delay*, *Shared_Interoffice_Trunk*, or *PRI_B-Channel* trigger. Refer to Volume 1, Chapter 4, “Collect_Information PIC,” for details.
`Network_Busy`	This message requests data from the SCP to direct call processing. This message is sent as a TDP request message when a CAIN call meets the trigger criteria for a *Network_Busy* trigger or as an EDP-Request message when a CAIN call encounters an active ***Network_Busy*** EDP. Refer to Volume 1, Chapter 6, Select_Route PIC,” for details.
`O_Answer`	This message notifies the SCP when the armed ***O_Answer*** EDP is reached. Refer to Volume 1, Chapter 8, “O_Alerting PIC,” for details.
`O_Called_Party_Busy`	This message requests data from the SCP to direct call processing. This message is sent as a TDP request message when a CAIN call meets the trigger criteria for an *O_Called_Party_Busy* trigger or as an EDP-Request message when a CAIN call encounters an active ***O_Called_Party_Busy*** EDP. Refer to Volume 1, Chapter 7, “Send_Call PIC,” for details.
`O_Disconnect`	This message notifies the SCP when the armed ***O_Disconnect*** EDP is reached. Refer to Volume 1, Chapter 9, “O_Active and O_Suspended PICs,” for more information.
`O_Feature_Requested`	This message requests data from the SCP to direct call processing. This message is sent when a CAIN call meets the trigger criteria for an *O_Feature_Requested* trigger. Refer to Volume 1, Chapter 4, “Collect_Information PIC,” for details.

—continued—

Table 6-1
Outgoing call-related CAIN messages (continued)

Message	Description
`O_Mid_Call`	This message requests data from the SCP to direct call processing. The switch may send this message when a CAIN call meets the trigger criteria for an *O_IEC_Reorigination* trigger. Refer to Volume 1, Chapter 7, "Send_Call PIC," Volume 1, Chapter 8, "O_Alerting PIC," and Volume 1, Chapter 9, "O_Active and O_Suspended PICs," for more information. The switch may also send this message as an EDP-Request, to indicate that a *Switch_Hook_Flash* has occurred.
`O_No_Answer`	This message requests data from the SCP to direct call processing. This message is sent as a TDP request message when a CAIN call meets the trigger criteria for a *O_No_Answer* trigger or as an EDP-Request message when a CAIN call encounters an active ***O_No_Answer*** EDP. Refer to Volume 1, Chapter 8, "O_Alerting PIC," for details.
`Origination_Attempt`	This message requests data from the SCP to direct call processing. This message is sent when a CAIN call meets the trigger criteria for an *Off_Hook_Immediate* trigger. Refer to Volume 1, Chapter 3, "O_Null PIC," for details.
`O_Term_Seized`	This message notifies the SCP when the armed ***O_Term_Seized*** EDP is reached. Refer to Volume 1, Chapter 7, "Send_Call PIC," for details.
`Resource_Clear`	This message sent in response to conversational digit collection initiated by the SCP. This message also contains any digits collected as requested by the SCP. This message is used in response to a **`Send_To_Resource`** message when finished. Refer to Volume 4, "Conversational processing," for more information.
`Termination_Attempt`	This message requests data from the SCP to direct call processing. This message is sent when a CAIN call meets the trigger criteria for a *Termination_Attempt* trigger. Refer to Volume 1, Chapter 10, "T_Null PIC," for more information.
`Timeout`	This message requests data from the SCP to direct call processing. This message is sent as an EDP-Request message when a CAIN call encounters an active *Timeout* event. Refer to Volume 1, Chapter 9, "O_Active and O_Suspended PICs," for more information.

—end—

Table 6-2
Outgoing non-call-related CAIN messages

Message	Description
`ACG_Global_Ctrl_Restore_Success`	The switch sends this message to the SCP when an ACG Global Restore Request has been successfully completed. Refer to Volume 3, "Outgoing CAIN messages." for more information.
`ACG_Overflow`	The switch sends this message to the SCP when an ACG control that cannot be added to the control list has been discarded (due to the lack of an available space on the ACG control list). Refer to Volume 3, "Outgoing CAIN messages." for more information.
`Termination_Notification`	The switch sends this message to the SCP when the call is released (sent in response to a **`Send_Notification`** message from the SCP). Refer to Volume 3, "Outgoing CAIN messages." for more information.

Use

The switch sends an **`ACG_Global_Ctrl_Restore_Success`** message to the SCP when the switch has successfully completed an ACG Global Restore Request.

Message parameters

There are no parameters sent by the switch in this message.

Fatal application errors

None

Nonfatal application errors

None

Associated logs

VAMP902

Note: When the switch sends a **`ACG_Global_Ctrl_Restore_Success`** message, a Return Result Component is displayed in the component field of the generated VAMP902 log.

Associated OMs

CAINNCRS

ACG_Overflow (end)

Use

The switch sends an **`ACG_Overflow`** message to the SCP when a new ACG control has been discarded. A new ACG control is discarded when it cannot be added to the ACG control list due to a lack of available spaces.

Message parameters

Table 6-3 provides the **`ACG_Overflow`** message parameters.

Table 6-3
ACG_Overflow parameters

Parameter	Usage	Definition
`ControlCauseIndicator`	Required	This parameter indicates whether the control is an SCP overload control or an SMS initiated control. It also contains the number of digits to which the control is applied.
`TranslationType`	Required	This parameter specifies the translation type of the ACG control.
`GlobalTitleAddress`	Required	This parameter specifies the global title address of the ACG control.

Fatal application errors

None

Nonfatal application errors

None

Associated logs

VAMP902

Associated OMs

CAINNCRS

Use

The switch may send a **`Termination_Notification`** message when the call is released. This is done in response to a **`Send_Notification`** message from the SCP.

Message parameters

Table 6-4 describes the **`Termination_Notification`** message parameters.

Table 6-4
Termination_Notification parameters

Parameter	Usage	Definition
`EchoData`	Required	The switch sends this parameter to correlate the **`Termination_Notification`** with the appropriate **`Send_Notification`** message.
`TerminationIndicator`	Required	The switch sends this parameter to indicate the reason for sending the **`Termination_Notification`** message.
`ConnectTime`	Optional	The switch sends this parameter to specify how much time has elapsed since the call was answered. ***Note:*** The ***`ConnectTime`*** parameter can only be sent if the AnswerIndication field within ***`TerminationIndicator`*** is set to YES.
`BusyCause`	Optional	The switch sends this parameter to indicate the reason the call cannot be completed to the terminating party. ***Note:*** The ***`BusyCause`*** parameter can only be sent if the BusyCauseIndication field within ***`TerminationIndicator`*** is set to YES.

Associated logs

VAMP902

Associated OMs

CAINAGOM, CAINMSGR, CAINTRIG

Outgoing IN/1 messages

ATTENTION
Outgoing messages require the CAIN0100 SOC option in order to function. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information.

Note: Outgoing messages are handled differently for LNP and AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide and UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Outgoing IN/1 messages

Once trigger criteria is met and extension blocks are allocated, the appropriate TCAP outgoing message is built and sent to the SCP. Table 7-1 lists the supported outgoing call-related messages.

Table 7-1
Outgoing call-related IN/1 messages

Message	Description
`Start`	The switch sends this message when the call encounters a toll-free service trigger interaction and the trigger action is QUERY.

Table 7-2 lists the supported IN/1 non-call related incoming messages.

Table 7-2
Outgoing non-call related IN/1 messages

Message	Description
`Termination_Information`	The switch may send this message to the SCP when the call is released (sent in response to a **`Termination`** message from the SCP). This message does not contain an operation code, which normally describes the the message type. The message type is specified by the ComponentTypeIdentifier.

Table 7-3 lists the supported IN/1 outgoing error messages.

Table 7-3
Outgoing IN/1 error messages

Message	Description
`Reject`	The switch sends this message when a protocol error is encountered in a message sent by the SCP. This error message does not contain an operation code, which normally describes the message type. The message type is specified by the ComponentTypeIdentifier.
`Report_Error`	The switch sends this message if a received message indicates the call should be completed through an IXC that does not serve the originating LATA, or if it detects an error in an **`ACG`** component of the response message.

Fatal application errors

For more information on fatal application errors, refer to Chapter 1, “TCAP messaging.”

Nonfatal application errors

For more information on nonfatal application errors, refer to Chapter 1, “TCAP messaging.”

Associated logs

VAMP902

Associated OMs

CAINMSGS

Reject (end)

Use

The switch sends a **`Reject`** message when protocol errors are encountered in messages received from the SCP. The **`Reject`** message does not contain an operation code, which normally describes the message type. The message type is actually contained in the ComponentTypeIdentifier.

Message parameters

The **`Reject`** message is sent in a unidirectional package, with a component type of Reject.

Table 7-4 describes the **`Reject`** message parameters.

Table 7-4
Reject message parameters

Parameter	Usage	Definition
`ProblemCode`	Required	The ***`ProblemCode`*** parameter identifies the reason for sending the **`Reject`** message.
`Digits` (called party number)	Optional	The ***`Digits`*** (calledpartynumber) parameter contains the dialed tollfree number to further assist the SCP in fault isolation.

Associated logs

VAMP902

Associated OMs

CAINMSGS

Use

The switch sends a **`Report_Error`** message for one of the following reasons:

- a received response message indicates that the call should be completed through an IXC that does not serve the originating LATA (the call is also routed to an announcement)
- the switch can complete a call, but detects an error in an **`ACG`** component of the response message

Message parameters

The **`Report_Error`** message is sent in a unidirectional package, with a component type of Invoke (Last).

Table 7-5 describes the **`Report_Error`** message parameters.

Table 7-5
Report_Error parameters

Parameter	Usage	Definition
`ProblemData`	Required	The ***`ProblemData`*** parameter contains the Identifier, Length of Contents, and Contents of an erroneous data element.
`Digits` (called party number)	Optional	The ***`Digits`*** (called party number) parameter contains the dialed tollfree number to assist the SCP in fault isolation.
`Digits` (calling party number)	Optional	The ***`Digits`*** (calling party number) parameter the 3, 6, or 10-digit ANI associated with the call.

Associated logs

VAMP902

Associated OMs

CAINMSGS, TFREE533

Start (end)

Use

The switch sends a **`Start`** message when the call encounters a toll-free service trigger interaction and the trigger action is QUERY.

Message parameters

The **`Start`** message is sent in a query package, with a component type of Invoke (Last).

Table 7-6 describes the **`Start`** message parameters.

Table 7-6
Start message parameters

Parameter	Usage	Definition
`ServiceKey`	Required	The ***`ServiceKey`*** parameter contains the called party number for the call.
`Digits` (calling party number)	Required	The ***`Digits`*** (calling party number) parameter contains the 3, 6, or 10-digit ANI associated with the call. For PRI originations, this parameter contains the datafill from the LANI option DIGITS field in table CALLATTR.
`Digits` (LATA)	Required	The ***`Digits`*** (LATA) parameter contains the LATA code associated with the call. For PRI originations, this parameter is populated with the LATANUM in table LATAID from the LATA option in table CALLATTR.
`OriginatingStationType`	Required	The ***`OriginatingStationType`*** parameter contains the II info digits associated with the ANI for the call, or any incoming Originating Line Information (OLI) digits.

Associated logs

VAMP902

Associated OMs

CAINMSGS, TFREE533

Use

The switch sends a **Termination_Information** message when the call is released. This is done in response to a **Termination** message from the SCP.

Message parameters

The **Termination_Information** message is sent in a unidirectional package, with a component type of Return Result (Last).

Table 7-7 describes the **Termination_Information** message parameters.

Table 7-7
Termination_Information message parameters

Parameter	Usage	Definition
TerminationIndicators	Required	The ***TerminationIndicators*** parameter specifies which indicators are used for the call.
EchoData	Required	The ***EchoData*** parameter correlates the termination information with the received request for information.
ConnectTime	Required	The ***ConnectTime*** parameter indicates the duration of the call between answer and disconnect.
ErrorCode	Optional	The ***ErrorCode*** parameter indicates if an error condition has resulted from an unexpected data value, unexpected component sequence, unavailable network resource, or unavailable data.
StandardUserErrorCode	Optional	The ***StandardUserErrorCode*** parameter replaces the ***ErrorCode*** parameter if termination information cannot be returned because of a customer action (that is, user abandon).

Associated logs

VAMP902

Associated OMs

None

Outgoing CAIN message parameters

ATTENTION

Outgoing messages and extension parameters require certain SOC options in order to function. Refer to Volume 5, Chapter 5, "NetworkBuilder SOC functionality," for more information.

Table 8-1 lists the parameters that the switch may send to the SCP within a TCAP outgoing message.

Note: Outgoing message parameters and extension parameters are handled differently for LNP and AXXESS agents. Refer to the *UCS DMS-250 Local Number Portability Application Guide* or *UCS DMS-250 CAIN/FlexDial Interactions*. Refer to Chapter 3, "Event processing," for more information on outgoing EDP message parameters and extension parameters.

Table 8-1
Outgoing CAIN message parameters

Parameter	Usage
ACGEncountered	Optional
AccessCode (Note)	Optional
BearerCapability	Optional for the **Close** message; required for all others
BusyCause	Optional
CalledPartyID (Note)	Optional
Note: Based on protocol control, these parameters use the AIN Digits format. Refer to the population rules for each parameter for more information.	
—continued—	

Table 8-1
Outgoing CAIN message parameters (continued)

Parameter	Usage
CallingPartyID (Note)	Optional
Carrier	Optional
CcID	Optional
ChargeNumber (Note)	Optional
ChargePartyStationType	Optional
ClearCause	Required
ClearCauseData	Optional
CloseCause	Required
CollectedAddressInfo (Note)	Optional
CollectedDigits (Note)	Optional
ConnectTime	Optional
ControlCauseIndicator	Required
EchoData	Required
ExtensionParameter	Optional
accountCode	Optional
acgRequery	Optional
adin	Optional
billingNumber	Optional
busyRoute	Optional
cainGroup	Optional
cainPRT	Optional
collectedAddress (Note)	Optional
connectTime	Optional
jurisdictionInfo	Optional
lnpReceived	Optional

Note: Based on protocol control, these parameters use the AIN Digits format. Refer to the population rules for each parameter for more information.

—continued—

Table 8-1
Outgoing CAIN message parameters (continued)

Parameter	Usage
`netinfo`	Optional
`numReorig`	Optional
`origTrunkInfo`	Optional
`pinDigits`	Optional
`reorigCall`	Optional
`subscriptionInfo`	Optional
`switchID`	Optional
`termTrunkInfo`	Optional
`treatment`	Optional
`t1Overflow`	Optional
`universalAccess`	Optional
`univIdx`	Optional
`FailureCause`	Optional
`FeatureActivatorID`	Required
`GlobalTitleAddress`	Required
`IPReturnBlock`	Optional
`JurisdictionInfo`	Optional
`Lata`	Optional
`LegID`	Optional
`NotificationIndicator`	Optional
`PointInCall`	Required for **`O_Feature_Requested`** Optional for **`O_Mid_Call`** and **`O_Disconnect`**
`TerminationIndicator`	Required
`TranslationType`	Required

Note: Based on protocol control, these parameters use the AIN Digits format. Refer to the population rules for each parameter for more information.

—continued—

Table 8-1
Outgoing CAIN message parameters (continued)

Parameter	Usage
`TriggerCriteriaType`	Optional
`UserID`	Optional for the **`Close`** message; required for all others
`VerticalServiceCode` (Note)	Optional
Note: Based on protocol control, these parameters use the AIN Digits format. Refer to the population rules for each parameter for more information.	
—end—	

When an outgoing message is built, the switch formats the message and sends it to the message encoder. Once encoded, the message is sent to the SCP.

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 8-2 shows fatal application errors that may occur before an SCP response is returned.

Note: Fatal application errors are handled differently for LNP and AXXESS agents. Refer to the *UCS DMS-250 Local Number Portability Application Guide* and *UCS DMS-250 CAIN/FlexDial Interactions*.

Table 8-2
Outgoing message parameter fatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
Message driver error (Note 1)	CAIN200	No	ERRACT in trigger table (Note 2)
CAIN_T1_TIMEOUT (Note 3)	CAIN200	Yes	ERRACT in trigger table (Note 2)
Unable to allocate extension block	CAIN200	No	ERRACT in trigger table (Note 2)
Unable to allocate VAMP resources	CAIN200 VAMP202	No	ERRACT in trigger table (Note 2)
Unable to allocate message against SOC option CAIN0100	CAIN102 CAIN200	No	ERRACT in trigger table (Note 2)

Note 1: A problem was encountered while sending the query message to the SCP.
Note 2: ERRACT is provisioned in trigger tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT. Table OFFCCODE defaults to the ROUTE error action, no error action is datafilled. Otherwise, the switch applies AINF to the call.
Note 3: The SCP did not respond within the provisioned amount of time. Refer to the CAINPARM CAIN_T1_TIMEOUT section of the *UCS DMS-250 Data Schema Reference Manual* for more information.

Nonfatal application errors

Table 8-3 shows nonfatal application errors that can occur before an SCP response is returned.

Table 8-3
Outgoing message parameter nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle	CAIN101 CAIN102	No	Extension parameters are ignored
Failure to encode an optional parameter	CAIN101	No	Optional parameters are ignored

Associated logs

CAIN101, CAIN102, CAIN200, CAIN201, CAIN903, CAIN905, VAMP201, VAMP202, VAMP203, VAMP601, VAMP602, VAMP603

Associated OMs

CAINMSGS, CAINMSGR, CAINAGOM, CAINOM, CAINTRIG, VTCAPERR, VTCAPSNT, VTCAPRCV

Parameter definition

This parameter contains the account code collected from in-switch translations or collected by the *O_Feature_Requested* trigger. This number may or may not be validated in-switch.

Message types

The following TDP-Request messages support the ***`AccessCode`*** parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`O_Mid_Call`**

Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

The range of values is 2–12 digits when collected in-switch. The range of values is up to 24 digits when collected using *O_Feature_Requested* feature processor.

AccessCode (end)

Population rules

Table 8-4 provides the population rules for the `AccessCode` parameter.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

ATTENTION
Population of the `AccessCode` parameter with the UCS DMS-250 collected account code is controlled by the CAIN_PROTOCOL_VERSION parameter in table CAINPARM. If the CAIN_PROTOCOL_VERSION parameter is set to V3 the `AccessCode` parameter is not populated and therefore not sent.

Table 8-4
AccessCode parameter population rules

	CAIN_PROTOCOL_ VERSION parameter set to V2 or lower		**CAIN_PROTOCOL_ VERSION parameter set to V3 or higher**	
Rule	**NOA**	**Numbering Plan**	**NOA**	**Numbering Plan**
When available	ACCT	PRVT	N/A	N/A

Restrictions

None

Parameter definition

This parameter contains the control indicator (SCP Overload or SOCC), the control type, and the number of digits in the control.

Message types

The following TDP-Request messages support the ***ACGEncountered*** parameter:

- **Origination_Attempt**
- **Info_Collected**
- **O_Feature_Requested**
- **Info_Analyzed**
- **O_Mid_Call**
- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **Termination_Attempt**

Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

ACGEncountered (end)

Range of values

Table 8-5 shows the range of values for each field that may be present in the ***ACGEncountered*** parameter.

Table 8-5
ACGEncountered field values

Field name	Range of values
SCP Overload Cntrl Indicator	No SCP control encountered
	SCP control encountered
SOCC Init Cntrl Indicator	No SOCC control encountered
	SOCC control encountered
Control Type	Normal Control
	Manual Individual Control
	Manual Global Control
	Both Manual Global and Manual Individual control encountered
ACG Encountered	1 – 10 (number of digits in the control encountered)

Population rules

None

Restrictions

None

Parameter definition

The ***`Amp1`*** parameter is supported in an outgoing **`Info_Collected`** message and an incoming **`Analyze_Route`** response messages as a test parameter and does not affect call processing. If the ***`Amp1`*** parameter is received in a response message other than **`Analyze_Route`**, the rest of the message is processed as usual, and a CAIN101 log is generated to indicate an unexpected parameter.

Message types

The following outgoing message supports the ***`Amp1`*** parameter:

- **`Info_Collected`**

Usage

Optional

Population rules

- Only the AMPTime field of the ***`Amp1`*** parameter is used by the SSP and the SCP, the others fields are populated with 0s. The AMPTime field specifies the time and date that the SCP should use when processing a message and is used to test services that make time-sensitive call processing decisions.

Associated logs

CAIN101, VAMP901, VAMP902

Restrictions

None

Parameter definition

The ***BearerCapability*** parameter contains the bearer capability of the call at the time the query is sent.

Message types

The following TDP-Request messages support the ***BearerCapability*** parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**

  Note: This parameter is handled differently in the **`Info_Analyzed`** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

- **`O_Mid_Call`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Termination_Attempt`**

The following EDP-Request messages support the ***BearerCapability*** parameter:

- **`Network_Busy`**
- **`O_Abandon`**
- **`O_Called_Party_Busy`**
- **`O_Disconnect`**
- **`O_Mid_Call`**
- **`O_No_Answer`**
- **`Timeout`**

The following EDP-Notification messages support the ***BearerCapability*** parameter:

- **O_Term_Seized**
- **O_Answer**
- **O_Disconnect**

The following messages support the ***BearerCapability*** parameter:

- **Close** (transaction control message)
- **Failure_Outcome**

Usage

Required

Optional in **Close** message.

Range of values

speech
f31kHzAudio
f7kHzAudio
b56kbps
b64kbps
multirate

Population rules

If the CAIN_PROTOCOL_VERSION parameter is set to V3 or higher, and either the f7kHzAudio or the multirate is received, the value is changed to speech.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Parameter definition

This parameter indicates the reason the call cannot be completed to the terminating party.

Message types

The following TDP-Request messages support the ***BusyCause*** parameter:

- **Network_Busy**
- **O_Called_Party_Busy**

The following EDP-Request messages support the ***BusyCause*** parameter:

- **Network_Busy**
- **O_Called_Party_Busy**

The following non-call related message also supports the ***BusyCause*** parameter:

- **Termination_Notification**

Usage

Optional

Range of values

The following data is transmitted in the ***BusyCause*** parameter:

- coding standard – contains ccitt, international, national, or network
- location – contains local_network or transit_network
- cause value – contains the incoming release cause value

  Note: The translated description for the cause value may not match the description defined by Bellcore and may instead reflect the description of another standard.

Population rules

The SS7 or ISUP release cause value received is used as follows:

- SS7 value is sent as received from the incoming network message
- ISDN cause values are mapped to ISUP values

Network busy due to exhausted route list contains: ccitt, transit_network, and ci_no_circuit_available.

Called party busy due to ONNET trunk busy contains: ccitt, local_network, and ci_user_busy.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The CAIN office parameter, RESTRICT_NETBUSY_BUSYCAUSE, in table CAINPARM controls the sending of this parameter by the switch. When this CAIN office parameter is set to N, the ***`BusyCause`*** parameter is included in the **`Network_Busy`** EDP-Request message. When the CAIN office parameter is set to Y, this parameter is not included in the message.

Parameter definition

The ***CalledPartyID*** is the known address or directory number of the called party. This number is used to route the call. When a call is initiated, ***CalledPartyID*** holds the dialed number. However, this number may change during standard pretranslation or N00 translation (in-switch or off-board).

Message types

The following TDP-Request messages support the ***CalledPartyID*** parameter:

- **Info_Analyzed**

  Note: This parameter is handled differently in the **Info_Analyzed** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **O_Mid_Call**
- **Termination_Attempt**

  Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Maximum of 18 digits

Population rules

Table 8-6 shows the population precedence rules for the ***CalledPartyID*** parameter. Refer to the *UCS DMS-250 NetworkBuilder Carrier AIN TCAP Protocol Definition* and the *UCS DMS-250 NetworkBuilder Compliancy Matrix* for more information on abbreviated terms.

Note: The CAIN_PROTOCOL_VERSION parameter controls the population of the ***CalledPartyID*** parameter.

Table 8-6
CalledPartyID parameter population precedence rules

		CAIN_PROTOCOL_ VERSION parameter set to V2 or lower		**CAIN_PROTOCOL_ VERSION parameter set to V3 or higher**	
	Rule	**NOA**	**Numbering Plan**	**NOA**	**Numbering Plan**
1	International or international partitioned numbers	INTL	ISDN	INTL	ISDN
2	Hotline calls	HOTL	PRVT	Not applicable	Not applicable
3	OFFNET	NATL	ISDN	NATL	ISDN
4	N00 (Note 1)	N00	PRVT	NATL	ISDN
5	ONNET	VPN	PRVT	SUBR	ISDN
6	Number does not fully pretranslate	VPN	PRVT	UNK	UNK

Note 1: This number looks like a national number due to pretranslations.
Note 2: For Nature of Address and Numbering Plan values, refer to the *UCS DMS-250 NetworkBuilder Carrier AIN TCAP Protocol Definition* .

Restrictions

The SCP should supply an address in the ***CalledPartyID*** parameter in all cases where the switch's query lacked the ***CalledPartyID*** or ***CollectedAddressDigits*** parameters.

CallingPartyID (continued)

Parameter definition

The ***CallingPartyID*** parameter contains the address used for caller identification purposes and is always transmitted.

Message types

The following TDP-Request messages support the ***CallingPartyID*** parameter:

- **Origination_Attempt**
- **Info_Collected**
- **O_Feature_Requested**
- **Info_Analyzed**
- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **O_Mid_Call**
- **Termination_Attempt**

Note 1: This parameter is not sent in EDP-Request messages.

Note 2: This parameter is handled differently for AXXESS agents. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Note 3: Refer to *UCS DMS-250 Local Number Portability Application Guide* for more information on ***CallingPartyID*** parameter for LNP queries.

Usage

Optional

Range of values

Maximum of 24 digits

Population rules

Table 8-7 shows the precedence rules for the information populated into the ***CallingPartyID*** parameter.

Table 8-7
CallingPartyID parameter population rules

	Rule	NOA	Numbering Plan
1	For CAIN_PROTOCOL_VERSION V5 or higher, the ***CallingPartyID*** is populated with the Calling Party Number received from the SCP on a previous interaction with the SCP for the same call.		
2	Calling Party Number transmitted in the SS7 IAM message. (Note 2)		
3	Valid FGD ANI (Note 3)	NATL	ISDN
4	PRI CallingLineID (CLID). For CAIN_PROTOCOL_VERSION V5 or higher, the following extra mapping from the Calling Party Number information element in the SETUP message to the ***CallingPartyID*** parameter is added: • The nature of address field is mapped directly from the type of number field in the calling party number information element. • The presentation restriction indicator field is mapped directly from the presentation status field in the calling party number information element.	NATL	ISDN
5	Valid Originating agency SNPA	NATL	ISDN
6	Default SNPA value in table CAINPARM	UNK	UNK

Note 1: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.
Note 2: If the switch receives a Calling Party Number in the SS7 IAM message, the switch populates the ***CallingPartyID*** parameter with the digits, NOA, and numbering plan values received. Also, for CAIN_PROTOCOL_VERSION V5 or higher, the presentation restriction indicator field is mapped directly from the address presentation restriction indicator in the Calling Party Number parameter of the received SS7 IAM message.
Note 3: DAL PANIs and FGD PANIs never populate the ***CallingPartyID*** parameter.

—end—

CallingPartyID (end)

Restrictions

The following restrictions apply:

- The content of this parameter is used for global title with CAIN_CLID_GT global title translations.

  Note: If the NOA of the ***`CallingPartyID`*** is not NATL, the SNPA is used for global title translations.

- PANIs are not supported.
- This parameter is always sent with presentation allowed, except for SS7 and PRI trunks when the CAIN_PROTOCOL_VERSION is set to V5 or higher.

Parameter definition

This parameter transmits the carrier identification code (CIC) received from an access tandem switch, when available, or when parameter SEND_CARRIER_FROM_TRKGRP in table TRKGRP is set to Y.

With UCS17, for PRI originations, this parameter transmits the CIC value from the DEFCIC option in table CALLATTR.

Message types

The following TDP-Request messages support the ***`Carrier`*** parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**

Note 1: This parameter is not sent in EDP-Request messages.

Note 2: This parameter is handled differently for AXXESS agents. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Usage

Optional

Range of values

Carrier Selection Field

- NO_INDICATION
- PRESUBSCRIBED_AND_NOT_INPUT
- PRESUBSCRIBED_AND_INPUT
- PRESUBSCRIBED_AND_NO_INDICATION
- NOT_PRESUBSCRIBED_AND_INPUT

Carrier (end)

Carrier ID field

- 0000–9999

Note: This parameter is always four digits in length. If the value is a three-digit number, the parameter is prefixed with a zero.

Population rules

Table 8-8 shows the population rules for the **`Carrier`** parameter. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Table 8-8
Carrier parameter population rules

Originating agent	Call processing populates Carrier with	Carrier selection field value
SS7	CIC digits from the carrier identification parameter (CIP) or transit network selector (TNS)	Incoming carrier selection indicator (CSI) value or No_indication
PTS FGD with 0ZZ + CIC	CIC from dialing	No_indication
Non-AXXESS	carrier from table TRKGRP when parameter SEND_CARRIER_FROM_TRKGRP is set to Y	Presubscribed_and_not_input
PRI	CIC value from DEFCIC option in table CALLATTR.	

Parameter definition

The ***CcID*** parameter specifies the current call configuration. Refer to Chaper 4, “Call configuration model,” for more information.

Message types

The following messages support the ***CcID*** parameter:

- **`O_Abandon`**
- **`Failure_Outcome`**
- **`O_Mid_Call`** TDP-Request and EDP-Request
- **`Timeout`**
- **`CTR_Clear`**

Usage

Optional

Range of values

The following values are supported:

- originatingSetup
- stable2party
- threePartySetup
- threePartySetupComplement
- partyOnHold
- partyOnHoldComplement
- stableMParty
- Transfer

Population rules

None

Restrictions

None

Parameter definition

This parameter holds the billing number that is used to populate the CDR at this point in the call. If the switch has not determined a charge number, this parameter is not transmitted to the SCP.

Message types

The following TDP-Request messages support the ***ChargeNumber*** parameter:

- **Origination_Attempt**
- **Info_Collected**
- **O_Feature_Requested**
- **Info_Analyzed**
- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **O_Mid_Call**
- **Termination_Attempt**

Note 1: This parameter is handled differently in the **Info_Analyzed** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

Note 2: This parameter is handled differently for AXXESS agents. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Usage

Optional

Range of values

Maximum of 24 digits

Population rules

In messages other than **Origination_Attempt**, the value sent is determined by the **first applicable item** from the following list:

Table 8-9
ChargeNumber parameter population rules for messages other than Origination_Attempt

Rule	NOA	Numbering Plan
ChargeNumber received in an IAM for originating SS7 EANT or IMT, if CAIN_PROTOCOL_VERSION is V4 or higher.	Same as received	Same as received
ANI of the calling party, if an ANI can be determined for the calling party and the CAIN_PROTOCOL_VERSION is V4 or higher.	Unique subscriber number	ISDN
No ***ChargeNumber*** sent, if the CAIN_PROTOCOL_VERSION is V4 or higher (and the above two rules do not apply).	None	None
Authorization code, if an IAM received on the second leg of an SS7 Inter-IMT RLT call contains a ***GenericDigits*** parameter of type BILLNUM (CAIN_PROTOCOL_VERSION V3 or lower).	Authcode	Private
Billing Number determined previously through standard call processing or *O_Feature_Requested* processing (CAIN_PROTOCOL_VERSION V3 or lower).	Determined previously	Private
ANI of the calling party, if an ANI can be determined for the calling party and the CAIN_PROTOCOL_VERSION is V2 or V3	Unique National (significant) number	ISDN

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

—continued—

ChargeNumber (continued)

Table 8-9
ChargeNumber parameter population rules for messages other than Origination_Attempt

Rule	NOA	Numbering Plan
ANI of the calling party prefixed with information digits, if an ANI can be determined for the calling party (CAIN_PROTOCOL_VERSION V1 or lower)	I2ANI	Private
No ***ChargeNumber*** sent (if none of the other rules apply)	None	None
Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.		
—end—		

In the **Origination_Attempt** message, the value sent is determined by the **first applicable item** from the following list:

Table 8-10
ChargeNumber parameter population rules for Origination_Attempt message

Rule	NOA	Numbering Plan
Filed authorization code digits, if they are present on the originating trunk group and the CAIN_PROTOCOL_VERSION is V3 or lower	Authcode	Private
No ***ChargeNumber*** sent (in all other cases)	None	None
Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.		

In some cases, the UCS DMS-250 SSP sends billing information in the `billingNumber` extension parameter. Refer to Volume 3, Chapter 8 `billingNumber` extension parameter section for more information.

Restrictions

The SCP should provide a billing number in the ***`ChargeNumber`*** parameter in all cases where the query did not include the ***`ChargeNumber`*** parameter.

ATTENTION

Failure to provide the ***`ChargeNumber`*** parameter in this case results in failure to bill.

ChargePartyStationType (end)

Parameter definition

This parameter contains the information digits for the call.

Message types

The following TDP-Request messages support the ***ChargePartyStationType*** parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

0–99

Population rules

None

Restrictions

This parameter requires the office parameter CAIN_PROTOCOL_STREAM in table CAINPARM to be set to UCS07 or higher. Refer to *UCS DMS-250 Local Number Portability Application Guide* for more details on the ***ChargePartyStationType*** parameter for LNP queries.

Parameter definition

This parameter indicates the reason a connection between a caller and a resource was terminated.

Message types

The following messages support the ***ClearCause*** parameter:

- **`Resource_Clear`**
- **`CTR_Clear`**

Usage

Required

Range of values

Table 8-11 list the supported values.

Table 8-11
ClearCause values

Cause	Hex value	Meaning
normal	00	The value indicates the expected result for end of dialing or announcement complete.
timeout	02	A timeout occurred while attempting to access the resource.
resourceCancelled	03	The switch cancels the resource interaction in response to a **`Cancel_Resource_Event`** received from the SCP.
userAbandon	06	The caller went onhook during the resource interaction.
invalidCode	07	Reception of invalid digits or digit timeout.
failure	08	The received operation could not be performed due to the unavailability of a hardware or software resource.
channelsBusy	09	The local switch was unable to identify an idle trunk for termination to the IP.
calledPartyAnswered	0A	Answer was received after **`O_No_Answer`** was sent to SCP.

—continued—

ClearCause (end)

Table 8-11
ClearCause values (continued)

Cause	Hex value	Meaning
resourceNotAvailable	0B	The switch was unable to terminate to the requested resource.
resourceTypeNotSupported	0D	The specified resource is not supported.
taskRefused	0E	The switch has determined that the request is not allowed due to SOC or CAIN_CONVERSATION_LIMIT exceeded.
protocolError	11	SCP request encoding error and the switch can not determine a suitable action.
abort	12	The local switch encountered problems during an STR-Connection.
suppServiceInvoked	13	The IP invoked a supplemental service on the local switch (for example, RLT).
strCancelled	14	Indicates the long call duration timer expired during an STR-Connection.
—end—		

Population rules

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Parameter definition

The ***ClearCauseData*** parameter is used to send additional error information to the SCP when the IP responds with a Return Error component.

Message types

The following messages support the ***ClearCauseData*** parameter:

- **`Resource_Clear`**
- **`CTR_Clear`**

Usage

Optional

Range of values

1–20 bytes

Population rules

None

Restrictions

None

CloseCause (end)

Parameter definition

This parameter indicates the reason a **`Close`** message is sent to end a TCAP transaction between the switch and the SCP.

Message types

The following message supports the ***`CloseCause`*** parameter:

- **`Close`**

Usage

Required

Range of values

The following values are supported:

- unexpectedCommunication
- callTerminated
- eDPsCompleted
- calledPartyAnswered

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Parameter definition

This parameter contains the address collected from the incoming agent message, through subscriber dialing, through datafilled hotline digits, or through the *O_Feature_Requested* trigger.

With UCS17, functionality is provided for PRI originations. If the called party number information element (CdPN IE) contains a 7–digit number and table CALLATTR has the TRGPREFX option, the NPA from the SNPA field in table BCHNLMEM is prepended to the called number.

Message types

The following messages support the ***CollectedAddressInfo*** parameter:

- **CTR_Clear**
- **Info_Analyzed**
- **Info_Collected**
- **O_Feature_Requested**
- **O_Mid_Call** TDP-Request
- **Resource_Clear**

Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Maximum of 24 digits

Population rules

Table 8-12 shows the population rules for the ***CollectedAddressInfo*** parameter.

CollectedAddressInfo (end)

Table 8-12
CollectedAddressInfo parameter population rules

Rule	NOA	Numbering Plan
Populated according to GR-1298-CORE (Note 2)		
Originally collected address If option REORIGADDR is present on CAINGRP, the ***CollectedAddressInfo*** parameter is populated with the originally dialed address for the call after reorigination (Note 3)	UNK	UNK

Note 1: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.
Note 2: Used when the CAIN_PROTOCOL_VERSION parameter in table CAINPARM is set to V2 or higher.
Note 3: If option reorigaddr is not present, the ***CollectedAddressInfo*** parameter is populated with the newly collected address after reorigination.

Restrictions

Data in the ***CollectedAddressInfo*** parameter is not the universal access number.

Note: Although the ***CollectedAddressInfo*** parameter normally contains the initial collected address, universal access numbers require special handling. The address collected after the UA code is stored in the ***CollectedAddressInfo*** parameter and UA codes are stored in the ***ExtensionParameter***'s `universalAccess` parameter. Therefore, the ***CollectedAddressInfo*** parameter never contains a UA code.

Parameter definition

This parameter holds the collected PIN code, when available, or a PIN collected at *O_Feature_Requested*. It also contains digits collected during conversational digit collection. Refer to “Population rules” for more specific population information.

Message types

The following TDP-Request messages support the ***`CollectedDigits`*** parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**

The following messages also support the ***`CollectedDigits`*** parameter:

- **`Resource_Clear`** (see Restrictions for population)
- **`CTR_Clear`** (see Restrictions for population)

Usage

Optional

Range of values

Maximum of 24 digits

Population rules

Populating the ***`CollectedDigits`*** parameter with collected PIN digits in outgoing query messages is controlled by the CAIN_PROTOCOL_VERSION parameter in table CAINPARM. Table 8-13 shows the population rules for the ***`CollectedDigits`*** parameter in outgoing query messages.

CollectedDigits (end)

Table 8-13
CollectedDigits parameter population rules

Rule	**CAIN_PROTOCOL_ VERSION parameter set to V2 or lower**		**CAIN_PROTOCOL_ VERSION parameter set to V3 or higher**	
	NOA	**Numbering Plan**	**NOA**	**Numbering Plan**
When PIN digits are collected	PIN	PRVT	Not populated	Not populated
Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.				

The CAIN_PROTOCOL_VERSION does not affect parameter population for the ***CollectedDigits*** parameter in the **Resource_Clear** or **CTR_Clear** messages. The ***CollectedDigits*** parameter is populated with the digits collected for the **Send_To_Resource** or **Connect_To_Resource** conversational interactions and sent to the SCP in the **Resource_Clear** or **CTR_Clear** messages regardless of version. Refer to “Restrictions” in this chapter for more information.

Restrictions

A **Resource_Clear** or **CTR_Clear** message contains any buffered or collected digits associated with the call while in the UIF. The NOA and numbering plans are UNK.

Extension parameter definition

The ***ConnectTime*** parameter specifies how much time has elapsed since the call was answered (in minutes, seconds, and tenths of seconds).

Message types

The following message supports the ***ConnectTime*** parameter:

- **Termination_Notification**

Usage

Optional

Range of values

The range of values for minutes is 0–99999.
The range of values for seconds is 0–59.
The range of values for tenths of seconds is 0–9.

Population rules

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

ControlCauseIndicator (continued)

Parameter definition

This parameter specifies whether a control is an SCP overload control, or an SMS initiated control. ***`ControlCauseIndicator`*** also contains the number of digits to which the control is applied.

Message types

The following message types support the ***`ControlCauseIndicator`*** parameter:

- **`ACG_Overflow`**

Usage

Required

Range of values

Table 8-14 shows the range of values for each field that may be present in the `ControlCauseIndicator` parameter.

Table 8-14
ControlCauseIndicator field values

Field name	Range of values	Explanation
SCP Overload Controls Indicator	0	No SCP overload controls
	1	SCP overload controls
SMS Initiated Controls Indicator	0	No SMS initiated controls
	1	SMS initiated controls
Number of Digits to which Control is applied	000001	1 digit control
	000010	2 digit control
	000011	3 digit control
	000100	4 digit control
	000101	5 digit control
	000110	6 digit control
	000111	7 digit control
	001000	8 digit control
	001001	9 digit control
	001010	10 digit control

Population rules

This parameter contains the `ControlCauseIndicator` of the control that overflowed.

Application errors

None

ControlCauseIndicator (end)

Restrictions

None

Parameter definition

This parameter is used to match the SCP **`Send_Notification`** message with the switch **`Termination_Notification`** message.

Message types

The following message supports the ***`EchoData`*** parameter:

- **`Termination_Notification`**

Usage

Required

Range of values

A string of 12 hexadecimal digits

Population rules

This parameter contains a unique TCAP transaction identifier.

Restrictions

None

ExtensionParameter (continued)

ATTENTION
Extension parameters require the CAIN0200 SOC option. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information. A CAIN102 log is generated when the switch attempts to use extension parameters without the CAIN0200 SOC option.

Note: Extension parameters are handled differently for LNP and AXXESS agents. Refer to *UCS DMS-250 Local Number Portability Application Guide and UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Parameter definition

The ***ExtensionParameter*** set is provided to send optional feature-specific information to the SCP. Refer to Table 8-1 for a complete list of outgoing message extension parameters supported by CAIN call processing. Detailed descriptions of individual extension parameters follow in this chapter.

With UCS17, the ***ExtensionParameter*** parameter may be included in the **Info_Analyzed** message as well as in the **Info_Collected** message.

Message types

The following TDP-Request messages support the ***ExtensionParameter***:

- **Origination_Attempt**
- **Info_Collected**
- **O_Feature_Requested**
- **Info_Analyzed**
- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **O_Mid_Call**
- **Termination_Attempt**

The following EDP-Request messages support the ***ExtensionParameter***:

- **Network_Busy**
- **O_Called_Party_Busy**
- **O_Disconnect**
- **O_Mid_Call**
- **O_No_Answer**
- **Timeout**

The following EDP-Notification messages support the ***ExtensionParameter***:

- **O_Term_Seized**
- **O_Answer**
- **O_Disconnect**

Usage

Optional

Range of values

Refer to the appropriate extension parameter for more information.

Population rules

Refer to the appropriate extension parameter for more information.

Restrictions

None

FailureCause (end)

Parameter definition

This parameter is populated when a ***`ClearCause`*** parameter is encoded as "failure."

Message types

The following messages support the ***`FailureCause`*** parameter:

- **`Failure_Outcome`**
- **`Resource_Clear`**
- **`CTR_Clear`**

Usage

Optional

Range of values

The following values are supported

- appl_err – applicationError
- improperCoding
- inappropriateCondition
- inappropriateLegManipulation
- timerExpired
- unavailableResources

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Parameter definition

This parameter contains the identification number of the feature specified by table OFTRREQ's FEAT selector.

Message types

The following TDP-Request message supports the ***`FeatureActivatorID`*** parameter:

- **`O_Feature_Requested`**

Usage

Required

Range of values

1 = CARD feature
2 = AUTH feature
3 = ADDR feature

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

GlobalTitleAddress (end)

Parameter definition

This parameter specifies the global title address of an ACG control.

Message types

The following message type supports the ***`GlobalTitleAddress`*** parameter:

- **`ACG_Overflow`**

Usage

Required

Range of values

10 digits (encoded in Binary Coded Decimal format)

Population rules

This parameter contains the ***`GlobalTitleAddress`*** of the control that overflowed.

Application errors

None

Restrictions

None

Parameter definition

This parameter is used to send the result of any user-interaction with the IP to the SCP. When the ***IPReturnBlock*** parameter is received by the switch, it is passed directly to the SCP without verification of the parameter contents.

Message types

The following messages support the ***IPReturnBlock*** parameter:

- **Resource_Clear**
- **CTR_Clear**
- **Call_Info_From_Resource**

Usage

Optional

Range of values

The ***IPReturnBlock*** parameter contains the following:

- For VIP interactions the value is a string of bytes identified by collectible type (AUTH, CARD, ADDR, PIN, ACCT, and UNKNOWN) encoded as AINDigits
- For STR- and CTR-Connections the value is a string of bytes passed directly to the SCP without verification of the parameter contents

Population rules

None for STR- and CTR-Connections

For Virtual IP interactions the ***IPReturnBlock*** parameter contains the collected digits requested by the ***FlexParameterBlock*** of the **Send_To_Resource** message. Digits are returned by collectible type (AUTH, CARD, ADDR, PIN, ACCT, UNKNOWN). Digits are encoded as AINDigits. Table 8-15 lists the population rules for the ***IPReturnBlock*** parameter for in-switch Virtual IP interactions.

IPReturnBlock (continued)

Table 8-15
IPReturnBlock parameter population for Virtual IP interactions

Collectible	NOA	Numbering Plan
ADDR	UNK (Note)	UNK
AUTH	AUTH	PRVT
CARD	MCCS	PRVT
ACCT	ACCT	PRVT
PIN	PIN	PRVT
UNKNOWN	UNK	UNK

Note: If the actual NOA can be determined, it will be, otherwise, UNK UNK will be returned.

—end—

Restrictions

ATTENTION

Due to UCS DMS-250 size limitations for internal messaging, the UCS DMS-250 does not support the maximum size ***`IPReturnBlock`*** as specified in Bellcore's *GR-1129-CORE* and *GR-1299-CORE* documents. If the internal limit is exceeded, the switch sends a **`Resource_Clear`** or **`CTR_Clear`** message to the SCP in a conversation package with a ***`ClearCause`*** parameter value of `Abort`.

Table 8-16 identifies size limits by message and trunk type.

Table 8-16
IPReturnBlock size limits

Message	**Parameter breakdown**	**IP trunk type**	
		PRI (Note 1)	**ISUP**
`Resource_Clear`	ParmID and length byte overhead	3	3
	Payload	92	115
	Total	95	118
`Call_Info_From_Resource`	ParmID and length byte overhead	3	3
	Payload	36	115
	Total	39 (Note 2)	118

Note 1: Requires XPM load ELI81AZ or later.
Note 2: Due to CM-XPM interactions, the `Resource_Clear` message are not sent when this limit is exceeded.

JurisdictionInformation (end)

Parameter definition

This parameter contains the originating switch's LRN.

The feature, Office Wide Default JIP, provides an office parameter, DEFAULT_JIP, in table OFCVAR that allows a default JIP to be sent in CAIN queries to the SCP if the JIP option is not datafilled in table TRKGRP. This feature applies to DAL or PRI originations that terminate to SS7 trunks. Only six–digits of the DEFAULT_JIP are sent in CAIN queries to the SCP.

The UCS16 feature, Jurisdiction Option Enhancement provides four profiles that are available for look–up within the four pre–defined jurisdictions, through datafill in table MULTPROF.

Message types

The following TDP-Request message supports the ***JurisdictionInformation*** parameter:

- **Info_Analyzed**

Usage

Optional

Range of values

6 binary coded decimal (BCD) digits

Population rules

Note: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on ***JurisdictionInfo*** parameter population for LNP.

The office wide default JIP is captured in the ORIGLRN CDR field for DAL and PRI originations.

Restrictions

The switch only includes this parameter in an **Info_Analyzed** message when the CAIN_PROTOCOL_STREAM is set to UCS08 or higher.

Parameter definition

The ***`Lata`*** parameter identifies the Local Access and Transport Area (LATA) applicable to the call. If this parameter is included in a TDP-request message from the switch, the value of the ***`Lata`*** parameter is captured in the LATA field of the CDR.

Note: The ***`Lata`*** parameter is only supported when the CAIN_PROTOCOL_STREAM office parameter is set to UCS11 or higher.

With UCS17, for PRI originations, the ***`Lata`*** parameter is populated with the LATANUM entry in table LATAID from the LATA option in table CALLATTR.

Message types

The following TDP-Request messages support the ***`Lata`*** parameter:

- **`O_Feature_Requested`**
- **`Origination_Attempt`**
- **`Info_Analyzed`**
- **`Info_Collected`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Termination_Attempt`**

Note: This parameter is not supported by EDP-Request messages

Usage

Optional

Range of values

3 digit string in the AINDigits format

Population rules

None

Restrictions

The ***`Lata`*** parameter is supported for the following trunk group types:

- AXXESS

Lata (end)

- DAL
- EANT
- PRI

Parameter definition

The ***LegID*** parameter identifies a leg in a call segment.

Message types

The following messages support the ***LegID*** parameter:

- **`CTR_Clear`**
- **`Failure_Outcome`**
- **`O_Disconnect`**

Usage

Optional

Range of values

The possible values for the ***LegID*** parameter are as follows:

- 0 - controlling leg
- 1 - passive leg
- 2 - additional passive leg

Population rules

For **`O_Disconnect`**, a value of 0 indicates that the controlling leg disconnected, a value of 1 indicates that the passive leg disconnected, and a value of 2 indicates that the additional passive leg disconnected.

For **`CTR_Clear`**, the value is normally 0 indicating the event was caused by the controlling leg of the call. In CC6, the value identifies the leg that disconnected.

For **`Failure_Outcome`**, the ***LegID*** parameter is populated with the value of leg which caused the failure.

Restrictions

None

NotificationIndicator (continued)

Parameter definition

This parameter identifies the message type as a request or notification for EDP messages.

Message types

The following EDP-Request messages support the ***`NotificationIndicator`*** parameter:

- **`O_Abandon`**
- **`O_Disconnect`**
- **`O_Mid_Call`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Timeout`**

Note: This parameter is not sent in TDP-Request messages.

The following EDP-Notification messages support the ***`NotificationIndicator`*** parameter:

- **`O_Term_Seized`**
- **`O_Answer`**
- **`O_Disconnect`**

Usage

Optional

Range of values

True or False

Population rules

Contains the value TRUE in EDP-Notifications and FALSE in EDP-Requests.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

PointInCall (continued)

Parameter definition

The ***PointInCall*** parameter specifies the current point in call (PIC).

Message types

The following TDP-Request messages support the ***PointInCall*** parameter:

- **O_Feature_Requested**
- **O_Mid_Call**

Note: The ***PointInCall*** parameter is not sent in **O_Mid_Call** EDP-Request message.

The following EDP-Notification message supports the ***PointInCall*** parameter:

- **O_Disconnect**

The following EDP-Request messages support the ***PointInCall*** parameter:

- **O_Abandon**
- **O_Disconnect**

The following message supports the ***PointInCall*** parameter:

- **Failure_Outcome**

Usage

Required in the **O_Feature_Requested** message

Optional in the **Failure_Outcome**, **O_Abandon**, **O_Mid_Call**, and **O_Disconnect** messages

Range of values

collectInformation
sendCall
oAlerting
oActive
oSuspended

Population rules

This parameter contains the point in call (PIC).

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

TerminationIndicator (continued)

Parameter definition

This parameter indicates the reason for sending the **`Termination_Notification`** message.

Message types

The following message supports the ***`TerminationIndicator`*** parameter:

- **`Termination_Notification`**

Usage

Required

Range of values

Table 8-17 shows the range of values for each field that may be present in the ***`TerminationIndicator`*** parameter.

Table 8-17
TerminationIndicator field values

Field name	Range of values	Explanation
AnswerIndication	YES, NO (see Note)	If an event occurs that releases the call after the call is answered, this field is populated to YES.
BusyCauseIndication	YES, NO (see Note)	This field is populated to YES under the following circumstances: • the SSP detects that the call was not completed because of a Network Busy, Busy Report, or Busy event, after the SSP has sent a switch call related message • the caller abandons before the SSP receives the **`Send_Notification`** message, and the T1 timer has not expired

Note: Only one of the following fields can be set to YES for a call: AnswerIndication, BusyCauseIndication, UnrelatedErrorConditionIndication, and ExceptionIndication.

—continued—

Table 8-17
TerminationIndicator field values (continued)

Field name	Range of values	Explanation
UnrelatedErrorCondition Indication	YES, NO (see Note)	This field is populated to YES under the following circumstances: • the SSP detects an error in a message that results in final treatment (except the **`Send_Notification`** message) • the call is cleared because it exceeded the serial trigger limit
ExceptionIndication	YES, NO (see Note)	This field is set to YES if the SSP detects that the call was not completed, and an exit event related to incomplete calls other than Network_Busy, Busy Reported, or Busy is detected.
ReroutIndication	YES, NO	This field is set to YES if the **`Send_Notification`** message was sent with **`Collect_Information`**, **`Analyze_Route`**, or **`Forward_Call`**, and the SSP successfully rerouted the call based on the contents of the SCP response message.
DisplayTextProvidedIndi cation	YES, NO	This field is set to YES if the SCP message sent with the **`Send_Notification`** message contained a ***`DisplayText`*** parameter, and the SSP sent the contents of the ***`DisplayText`*** parameter to the access interface identified in the SCP response message.
NMControlListOverflowI ndication	YES, NO	This field is set to YES if the Network Management control list overflows.

Note: Only one of the following fields can be set to YES for a call: AnswerIndication, BusyCauseIndication, UnrelatedErrorConditionIndication, and ExceptionIndication.

—end—

TerminationIndicator (end)

Population rules

Refer to the Explanation column of Table 8-17, “TerminationIndicator field values.”

Restrictions

None

Parameter definition

This parameter specifies the translation type of an ACG control.

Message types

The following message type supports the ***TranslationType*** parameter:

- **ACG_Overflow**

Usage

Required

Range of values

an integer from 0–255

Population rules

This parameter contains the ***TranslationType*** of the control that overflowed.

Application errors

None

Restrictions

None

TriggerCriteriaType (continued)

Parameter definition

This parameter holds the nature of the criteria used to trigger, and also allows the trigger to be identified.

Message types

The following TDP-Request messages support the ***TriggerCriteriaType*** parameter:

- **Info_Analyzed**

  Note: This parameter is handled differently in the **Info_Analyzed** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

- **Info_Collected**
- **Network_Busy**
- **O_Called_Party_Busy**
- **O_Feature_Requested**
- **O_Mid_Call**
- **O_No_Answer**
- **Origination_Attempt**
- **Termination_Attempt**

Note: This parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Refer to population rules.

Population rules

Table 8-18 shows the population rules for the ***TriggerCriteriaType*** parameter.

TriggerCriteriaType (continued)

Table 8-18
TriggerCriteriaType parameter population rules

Rule	TriggerCriteriaType value
Off_Hook_Immediate trigger	offHookImmediate
PRI_B-Channel trigger with CLID	csp_clid
PRI_B-Channel trigger with national or VPN address	csp_addr
PRI_B-Channel trigger with an authcode database index value	csp_adin
PRI_B-Channel trigger with N00 address	csp_n00
PRI_B-Channel trigger with international address	csp_intl
PRI_B-Channel trigger with standard value (channelSetupPRI is sent when STD is datafilled in table PRIBCHNL)	channelSetupPRI
Shared_Interoffice_Trunk trigger with ANI	sio_ani
Shared_Interoffice_Trunk trigger with national or VPN address	sio_addr
Shared_Interoffice_Trunk trigger with an authcode database index value	sio_adin
Shared_Interoffice_Trunk trigger with information digits	sio_info
Shared_Interoffice_Trunk trigger with CIC	sio_cic
Shared_Interoffice_Trunk trigger with N00 address	sio_n00
Shared_Interoffice_Trunk trigger with standard value (sharedIOTrunk is sent when STD is datafilled in table SIOTRK)	sharedIOTrunk
Shared_Interoffice_Trunk trigger with international address	sio_intl

Note 1: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on the ***`TriggerCriteriaType`*** parameter for office code triggering.

Note 2: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

—continued—

TriggerCriteriaType (continued)

Table 8-18
TriggerCriteriaType parameter population rules (continued)

Rule	TriggerCriteriaType value
O_Feature_Requested trigger	oFeatureActivator
Customized_Dialing_Plan trigger	cust_int
Specific_Digit_String trigger with national or VPN address (datafilled as SDS_ADDR in table SPECDIG)	npaNXXXXXX
Specific_Digit_String trigger with an authcode database index value	sds_adin
Specific_Digit_String trigger with information digits	sds_info
Specific_Digit_String trigger ANI	sds_ani
Specific_Digit_String trigger CIC	sds_cic
Specific_Digit_String trigger with N00 address	sds_n00
Specific_Digit_String trigger with international address (countryCodenpaNXXXXXX is sent when SDS_INTL is datafilled in table SPECDIG)	countryCodenpaNXXXXXX
Specific_Digit_String trigger on 3 or fewer address digits, or INFO, or ANI, or CIC, or *Office_Code* trigger with NPA address	NPA
Specific_Digit_String trigger on 4 address digits	NPA_N
Specific_Digit_String trigger on 5 address digits	NPA_NX
Specific_Digit_String trigger on 6 address digits or *Office_Code* trigger with NPA_NXX address	NPA_NXX
Specific_Digit_String trigger on 7 address digits or *Office_Code* trigger with NPA_NXXX address	NPA_NXXX

Note 1: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on the **`TriggerCriteriaType`** parameter for office code triggering.

Note 2: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

—continued—

Table 8-18
TriggerCriteriaType parameter population rules (continued)

Rule	TriggerCriteriaType value
Specific_Digit_String trigger on 8 address digits or *Office_Code* trigger with NPA_NXXXX address	NPA_NXXXX
Specific_Digit_String trigger on 9 address digits or *Office_Code* trigger with NPA_NXXXXX address (datafilled as SDS_ADDR in table SPECDIG and table OFFCCODE)	NPA_NXXXXX
Specific_Digit_String trigger on10 address digits or *Office_Code* trigger with NPA_NXXXXXX address	NPA_NXXXXXX
Office_Code trigger with datafilled Bellcore defined value LNP_OFCD	LNP_OFCD
Network_Busy trigger	networkBusy
O_Called_Party_Busy trigger	oCalledPartyBusy
O_No_Answer trigger	oNoAnswer
OffHook_Delay trigger	offHookDelay
O_IEC_Reorigination trigger	oIECReorigination
Specific_Feature_Code trigger	specificFeatureCode
Termination_Attempt trigger	terminationAttempt

Note 1: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on the ***TriggerCriteriaType*** parameter for office code triggering.

Note 2: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

—end—

Restrictions

None

Parameter definition

The ***UserID*** parameter informs the SCP of the network identity of the originating agent. The value of this parameter is the switch identifier plus the trunk group number (defined in table CLLI) for the DN format, or trunk group number for the TrunkGroupID format, or the value in the ADNUM field (defined in table CLLI) of the originating DAL trunk group for the PrivateFacilityGID format.

Message types

The following TDP-Request messages support the ***UserID*** parameter:

- **Origination_Attempt**
- **Info_Collected**
- **O_Feature_Requested**
- **Info_Analyzed**

  Note: This parameter is handled differently in the **Info_Analyzed** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**
- **O_Mid_Call**
- **Termination_Attempt**

The following EDP-Request messages support the ***UserID*** parameter:

- **Network_Busy**
- **O_Abandon**
- **O_Called_Party_Busy**
- **O_Disconnect**
- **O_Mid_Call**
- **O_No_Answer**
- **Timeout**

The following EDP-Notification messages support the ***UserID*** parameter:

- **O_Term_Seized**
- **O_Answer**

- **`O_Disconnect`**

The following messages support the ***`UserID`*** parameter:

- **`Close`** (transaction control message)
- **`Failure_Outcome`**

Usage

Required

Optional in **`Close`** message.

Range of values

Table 8-19 shows the range of values available for the ***`UserID`*** parameter.

Table 8-19
UserID range of values

Parameter	Range of values	Format	CAIN_ PROTOCOL_ VERSION
Prefix (reserved)	00	DN	V2 or lower
SWID	000–999	DN	V2 or lower
CLLI	0000–9999	DN	V2 or lower
TrunkGoupID	0000–8191	TrunkGroupID	V3 or higher
PrivateFacilityGID	0000–8191	PRIVFAC	V3 or higher

Population rules

If the CAIN_PROTOCOL_VERSION parameter in table CAINPARM is set to V2 or lower the ***`UserID`*** parameter uses the DN format. When using the DN format, the ***`UserID`*** parameter is always a 10-digit number consisting of the Prefix, SWID, and CLLI values.

If the CAIN_PROTOCOL_VERSION parameter in table CAINPARM is set to V3 or higher the ***`UserID`*** parameter uses the TrunkGroupID or PRIVFAC format.

For V3 or higer, the PrivFac_UserID parameter in table CAINPARM controls which encoding format is used for the ***`UserID`*** parameter. The PrivFac_UserID parameter has two values, Y and N. When set to N, the

UserID (end)

`UserID` parameter uses the TrunkGroupID encoding format. When set to Y, the **`UserID`** parameter uses the PrivateFacilityGID encoding format and contains the incoming DAL PrivateFacilityGID from the ADNUM field in table CLLI.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Parameter definition

This parameter contains feature codes dialed by the subscriber.

Message types

The following TDP-Request messages support the ***`VerticalServiceCode`*** parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`O_Mid_Call`**

Usage

Optional

Range of values

Up to 10 digits with an NOA of HOTL and a numbering plan of PRVT

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

Extension parameter definition

This extension parameter contains the account code collected from in-switch translations or collected by the *O_Feature_Requested* trigger. This number may or may not be validated in-switch.

Note: The digits used to populate this extension parameter may have been collected by the Virtual IP (VIP).

Message types

The following TDP-Request messages support the `accountCode` extension parameter:

- **`Info_Analyzed`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`O_Mid_Call`**

Note: This parameter is not sent in **`O_Mid_Call`** EDP-Request message.

Usage

Optional

Range of values

2–12 digits when collected in-switch
Up to 24 digits when collected using *O_Feature_Requested* feature processor

CDR population

The digits contained in this extension parameter populates the ACCTCD field of the CDR.

Note: If any ***`AMADigitsDialedWC`*** parameters are received later in the call that are to populate the same CDR field (ACCTCD), the ***`AMADigitsDialedWC`*** parameters have precedence over the digits sent in this extension parameter.

Population rules

The existence and population of the `accountCode` extension parameter is controlled by the CAIN_PROTOCOL_STREAM parameter in table CAINPARM being set to UCS09.

ExtensionParameter accountCode (end)

The `accountCode` extension parameter is encoded as AINDigits, with an NOA of UNKNOWN and a Numbering plan (NumPlan) of ISDN.

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They will not be used to check for subscription.

ExtensionParameter
acgRequery (continued)

Parameter definition

A query message includes the `acgRequery` extension parameter when the message encounters an ACG control and call processing performs a requery using a new global title. This parameter, when present, indicates to the SCP that the current query is a requery message resulting from the original query being blocked by an ACG control.

Message types

The following TDP-Request messages support the `acgRequery` extension parameter:

- **`Info_Analyzed`**
- **`Info_Collected`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`Origination_Attempt`**
- **`O_No_Answer`**
- **`Termination_Attempt`**
- **`O_Mid_Call`**
- **`O_Feature_Requested`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Presence or Absence

Population rules

None

Restrictions

ACG Overflow global title routing is not allowed for *Office_Code* and *O_IEC_Reorigination*; therefore the `acgRequery` extension parameter is not support for either trigger.

Associated OMs

CAINOM

ExtensionParameter
adin (continued)

Extension parameter definition

This extension parameter contains the authcode database index (ADIN) used to select the appropriate table (AUTHCODU, AUTHCDU2, AUTHCDU3, AUTHCDU4, or AUTHCDU5) for authcode validation.

Message types

The following TDP-Request messages support the `adin` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**

Usage

Optional

Range of values

0 through 99

Population rules

Table 8-20 shows the population rules for the `adin` extension parameter.

Table 8-20
adin ExtensionParameter population rules

Rule
Delivered if an authcode is present but not used as part of a pseudo-ANI. Field EXTPARM (table CAINGRP) must contain adin.
Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

This parameter is not supported for AXXESS originations. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

ExtensionParameter
billingNumber (continued)

Extension parameter definition

The switch populates this extension parameter with a non-standard charge number (for example, CARD, AUTH, ACCT, PIN, and N00).

Note: The digits used to populate this extension parameter may have been collected by the Virtual IP (VIP).

Message types

The following TDP-Request messages support the `billingNumber` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**

  Note: This extension parameter is handled differently in the **`Info_Analyzed`** message for LNP queries. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note 1: This extension parameter is not sent in EDP-Request messages.

Note 2: This extension parameter is handled differently for AXXESS agents. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Usage

Optional

Range of values

Maximum of 24 digits

CDR population

The digits contained in this extension parameter will populate the BILLNUM field of the CDR.

Note: If any ***`AMADigitsDialedWC`*** parameters are received later in the call that are to populate the same CDR field (BILLNUM), the ***`AMADigitsDialedWC`*** parameters have precedence over the digits sent in this extension parameter.

Population rules

The existence and population of the `billingNumber` extension parameter is controlled by the CAIN_PROTOCOL_STREAM parameter in table CAINPARM being set to UCS09.

The `billingNumber` extension parameter is encoded as AINDigits. The NOA and Numbering Plan (NumPlan) varies based on the type of the billing number, as shown in Table 8-21.

Table 8-21
Billing numbers and NOAs for the billingNumber extension parameter

Type of Billing Number	NOA	NumPlan
Calling Card, MCCS number	CARD	PRVT
Authcode	AUTH	PRVT
Account code	ACCT	PRVT
Personal Identification digits	PIN	PRVT
800/888 services	N00	PRVT

The `billingNumber` extension parameter is controlled by the CAIN_PROTOCOL_VERSION and CAIN_PROTOCOL_STREAM parameters in table CAINPARM. To determine whether the `billingNumber` extension parameter is sent and for population information, refer to Table 8-22.

ExtensionParameter
billingNumber (continued)

Table 8-22
ChargeNumber parameter population based on protocol version and stream control

Protocol Version	**CAIN_PROTOCOL_STREAM set to UCS09**
CAIN_PROTOCOL_VERSION set to V3 or lower	NOAs supported by `billingNumber`: SUBR ANI I2ANI AUTH MCCS N00 NATL INTL UNKNOWN `billingNumber` can contain: AUTH N00 MCCS
CAIN_PROTOCOL_VERSION set to V4	Only standard NOAs supported by `billingNumber`: ANI of calling party; SUBR ANI not available or not provided ANI of calling party; NATL ANI of called party; SUBR ANI of called party; Not included ANI of called party; NATL `billingNumber` can only contain: AUTH N00 MCCS

Feature interactions

At the ***Info_Analyzed*** TDP, when an IAM on the second call leg of an SS7 Inter-IMT RLT call contains a Generic Digits Parameter (GDP) of type

ExtensionParameter billingNumber (end)

BILLNUM, the contents of the GDP are sent to the SCP in the `billingNumber` extension parameter with a NOA value of AUTH. If the BILLNUM is not present in the GDP or if the GDP contains no digits, then the existing population method for the `billingNumber` extension parameter applies.

The `billingNumber` extension parameter is controlled by the CAIN_PROTOCOL_VERSION and CAIN_PROTOCOL_STREAM parameters in table CAINPARM. To determine when the `billingNumber` extension parameter is used, refer to Table 8-23.

Table 8-23
billingNumber extension parameter protocol version and stream control

Protocol Version	CAIN_PROTOCOL_STREAM set to UCS09
CAIN_PROTOCOL_VERSION set to V3 or lower	`billingNumber` extension parameter is sent
CAIN_PROTOCOL_VERSION set to V4	`billingNumber` extension parameter is sent

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They will not be used to check for subscription.

ExtensionParameter
busyRoute (continued)

Extension parameter definition

This extension parameter contains the number of routing attempts for calls at ***Network_Busy***, ***O_Called_Party_Busy***, and ***O_No_Answer*** detection points (DPs). If an outpulse number is available, it is included as the routing number. If the call was routed using a trunk group and switch ID (SWID) from the ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, or ***SecondAlternateTrunkGroup*** parameter, the route index is included.

Message types

The following TDP-Request messages support the `busyRoute` extension parameter:

- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**

The following EDP-Request messages support the `busyRoute` extension parameter:

- **Network_Busy**
- **O_Called_Party_Busy**
- **O_No_Answer**

Usage

Optional

Range of values

The `busyRoute` extension parameter contains the following:

- Number of routing attempts – 0 to 99 (number of attempts to route from *Network_Busy*, *O_Called_Party_Busy*, or *O_No_Answer*.)
- outpulse number – 0 to 24 digits (outpulse number). NOA and numbering plan may be: NATL ISDN, INTL ISDN, or UNK UNK.
- route index:
  - — switch identifier – 0 to 127
  - — trunk group number – 0 to 8191 (from field ADNUM of table CLLI)

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

ExtensionParameter
cainGroup (end)

Extension parameter definition

The `cainGroup` extension parameter identifies the CAIN group whose trigger subscription caused the call to query.

Message types

The following TDP-Request messages support the `cainGroup` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

0 to 4095 (from table CAINGRP's GRPNUM field)

Population rules

Delivered when datafilled in table CAINGRP's EXTPARMS or TEXTPARMS fields.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

Extension parameter definition

The `cainPRT` extension parameter indicates to the SCP that the switch was performing CAIN pretranslations and an error occurred. This parameter contains the digits that were being collected at the time the error occurred.

Message types

The following message supports the `cainPRT` extension parameter:

- **`O_Feature_Requested`**

Usage

Optional

Range of values

0–24 digits

Population rules

This parameter contains the digits that were being collected at the time the error occurred.

Restrictions

The switch only includes this parameter in an **`O_Feature_Requested`** message when the CAIN_PROTOCOL_STREAM is set to UCS08 or higher.

ExtensionParameter collectedAddress (continued)

Extension parameter definition

This extension parameter contains the address collected from the incoming agent, through subscriber dialing, through datafilled hotline digits, or through the *O_Feature_Requested* trigger.

Message types

The following TDP-Request messages support the `collectedAddress` extension parameter:

- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**

Note: This parameter is not sent in EDP-Requests.

Usage

Optional

Range of values

Maximum of 24 digits in AIN Digits format

Population rules

Table 8-24 shows the population rules for the `collectedAddress` extension parameter.

Table 8-24
collectedAddress extension parameter population rules

Rule	NOA	Numbering Plan
Populated according to GR-1298-CORE (Note 2)		
Originally collected address	UNK	UNK
Note 1: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information. ***Note 2:*** Used when the CAIN_PROTOCOL_STREAM parameter in table CAINPARM is set to UCS08 or higher.		

ExtensionParameter collectedAddress (end)

Restrictions

The switch only includes this extension parameter in the **`Network_Busy`**, **`O_Called_Party_Busy`**, and **`O_No_Answer`** message when the CAIN_PROTOCOL_STREAM is set to UCS08 or higher.

ExtensionParameter
connectTime (end)

Extension parameter definition

This extension parameter specifies how much time has elapsed since the call was answered (in minutes, seconds, and tenths of seconds).

Message types

The following messages support the `connectTime` extension parameter:

- **`Timeout`**
- **`O_Disconnect`** EDP-Notification

Usage

Optional

Range of values

The range of values for minutes is 0–99999.
The range of values for seconds is 0–59.
The range of values for tenths of seconds is 0–9.

Population rules

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

ExtensionParameter jurisdictionInformation (end)

Extension parameter definition

This extension parameter contains the originating switch's LRN.

Message types

The following TDP-Request messages support the `jurisdictionInformation` extension parameter:

- **`Origination_Attempt`**
- **`O_Feature_Requested`**
- **`Info_Collected`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

6 binary coded decimal (BCD) digits

Population rules

Note: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on ***`JurisdictionInfo`*** extension parameter population for LNP.

Restrictions

The switch only includes this extension parameter in the **`Origination_Attempt`**, **`O_Feature_Requested`**, **`Info_Collected`**, **`Network_Busy`**, **`O_Called_Party_Busy`**, and **`O_No_Answer`** message when the CAIN_PROTOCOL_STREAM is set to UCS08 or higher.

ExtensionParameter
lnpReceived (end)

Extension parameter definition

This extension parameter indicates that LNP information has been received from a previous switch.

Message types

The following TDP-Request messages support the `lnpReceived` extension parameter:

- **`O_Feature_Requested`**
- **`Info_Collected`**
- **`Info_Analyzed`**

Usage

Optional

Range of values

Presence or absence

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter when the CAIN_PROTOCOL_STREAM is set to UCS07 or higher.

Extension parameter definition

This extension parameter contains the external network ID, network customer group ID, and network class of service to add support for multi-switch business group (MBG) calls.

Message types

The following TDP-Request messages support the `netinfo` extension parameter:

- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Info_Collected`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

PS (Party Selector):
0000–1111
LPII (Line Privileges Information Indicator):
0–1
BGID (Business Group Identifier):
0–1
Attst (Attendant Status):
0–1
EXTNETID (External Network ID):
0000000000000000–1111111111111111
NETCGID (Network Customer Group ID):
0000000000000000–1111111111111111
NCOS (Network Class of Service):
0000000000000000–1111111111111111

ExtensionParameter
netinfo (end)

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS07 or higher.

Extension parameter definition

This extension parameter indicates the number of times a user has reoriginated.

Message types

The following TDP-Request message supports the `numReorig` extension parameter:

- **`O_Mid_Call`**

Note: This parameter is not sent in **`O_Mid_Call`** EDP-Requests.

Usage

Optional

Range of values

1–99

Population rules

None

Restrictions

None

ExtensionParameter origTrunkInfo (continued)

Extension parameter definition

This extension parameter contains the trunk group number, trunk type, and trunk member number for the call's originating agent.

Message types

The following TDP-Request messages support the `origTrunkInfo` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

trunk group number: 0–8191
trunk type: DAL, DAL-TIE, EANT, PRI, IMT, FGA, FGB, FGC, or AXXESS
trunk member number: 0–8191

Population rules

Delivered when datafilled in table CAINGRP's EXTPARMS or TEXTPARMS field.

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

Extension parameter definition

This extension parameter holds the collected PIN code, when available, or a PIN collected at *O_Feature_Requested*. It also contains digits collected during conversational digit collection. Refer to “Population rules” for more specific population information.

Note: The digits used to populate this extension parameter may have been collected by the Virtual IP (VIP).

Message types

The following TDP-Request messages support the `pinDigits` extension parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**

Usage

Optional

Range of values

Maximum of 24 digits

CDR population

The digits contained in this extension parameter populate the PINDIGS field of the CDR.

Note: If any ***`AMADigitsDialedWC`*** parameters are received later in the call that are to populate the same CDR field (PINDIGS), the ***`AMADigitsDialedWC`*** parameters have precedence over the digits sent in this extension parameter.

Population rules

The existence and population of the `pinDigits` extension parameter is controlled by the CAIN_PROTOCOL_STREAM parameter in table CAINPARM being set to UCS09.

The `pinDigits` extension parameter is encoded as AINDigits, with an NOA of UNKNOWN and a Numbering Plan (NumPlan) of ISDN.

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They are not be used to check for subscription.

ExtensionParameter
reorigCall (end)

Extension parameter definition

This extension parameter, when present, indicates that the call in progress is the result of reorigination.

Message types

The following TDP-Request messages support the `reorigCall` extension parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Termination_Attempt`**

Usage

Optional

Range of values

Presence or absence

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

Extension parameter definition

This extension parameter indicates the digit type that the switch triggered on and the subscription method in use when the query occurred.

Message types

The following TDP-Request messages support the `subscriptionInfo` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

The digit type values include: info, adin, ani, xlaaddr, addr, cic

The subscription type values include: scp, addr, auth, ani, agent, office

Population rules

None

Restrictions

The switch only includes this parameter in the messages when the CAIN_PROTOCOL_STREAM is set to UCS08 or higher.

ExtensionParameter
switchID (end)

Extension parameter definition

This extension parameter parameter provides the switch identifier to the SCP.

Message types

The following TDP-Request messages support the `switchID` extension parameter:

- **`Origination_Attempt`**
- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**
- **`Termination_Attempt`**

Note: This parameter is not sent in EDP-Requests.

Usage

Optional

Range of values

The `switchID` extension parameter has a range of 1 to 999 (encoded as a Binary Coded Decimal).

Population rules

The existence and population of the `switchID` extension parameter is controlled by the CAIN_PROTOCOL_STREAM parameter in table CAINPARM being set to UCS09.

Restrictions

None

Extension parameter definition

This extension parameter contains the trunk group number, trunk type, and trunk member number for the call's terminating agent.

Message types

The following TDP-Request messages support the `termTrunkInfo` extension parameter:

- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Termination_Attempt`**

The following EDP-Request messages support the `termTrunkInfo` extension parameter:

- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_Disconnect`**
- **`O_Mid_Call`**
- **`O_No_Answer`**
- **`Timeout`**

The following EDP-Notification messages support the `termTrunkInfo` extension parameter:

- **`O_Term_Seized`**
- **`O_Answer`**
- **`O_Disconnect`**

Usage

Optional

Range of values

trunk group number: 0–8191
trunk type: DAL, DAL-TIE, EANT, PRI, IMT, FGA, FGB, FGC, or AXXESS
trunk member number: 0–8191

ExtensionParameter termTrunkInfo (end)

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Extension parameter definition

This extension parameter, when present, indicates the treatment set at the switch before trigger criteria is met.

Message types

The following TDP-Request messages support the `treatment` extension parameter:

- **`Info_Collected`**
- **`O_Feature_Requested`**
- **`Info_Analyzed`**
- **`O_Mid_Call`**

Note: This parameter is not sent in **`O_Mid_Call`** EDP-Requests.

Usage

Optional

Range of values

Refer to Appendix F, "Treatment codes."

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

The switch only includes this parameter in an **`Info_Analyzed`** message when the CAIN_PROTOCOL_STREAM is set to UCS06 or higher.

ExtensionParameter
t1Overflow (end)

Extension parameter definition

This extension parameter, when present, indicates that an overflow has occurred as a result of a T1 timeout on the query to the initial SCP.

Message types

The following TDP-Request messages support the `t1Overflow` extension parameter:

- **`Origination_Attempt`**
- **`O_Feature_Requested`**
- **`Info_Collected`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Termination_Attempt`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Presence or absence

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

Overflow SCP requerying only applies to initial query messages. Overflow SCP requerying does not apply when a T1 timeout occurs for messages sent in conversation. Also, if a T1 timeout occurs for an initial query message, only one overflow requery is attempted.

Extension parameter definition

This extension parameter allows the switch to identify a universal access number dialed by the customer to access in-switch services. The universal access number is a number (primarily 800 numbers) datafilled with the universal access (UA) selector in subfield PRERTSEL (field PRETRTE) of table STDPRTCT (Standard Pretranslator Control), subtable STDPRT (Standard Pretranslator).

Message types

The following TDP-Request messages support the `universalAccess` extension parameter:

- **`O_Feature_Requested`**
- **`Info_Collected`**
- **`Info_Analyzed`**
- **`Network_Busy`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`O_Mid_Call`**

Note: This extension parameter is not sent in EDP-Request messages.

Usage

Optional

Range of values

Up to 24 digits

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Feature interactions

At the ***Info_Analyzed*** TDP, when an IAM on the second call leg of an SS7 Inter-IMT RLT call contains a Generic Digits Parameter (GDP) of type UNIVACC, the contents of the GDP are sent to the SCP in the `universalAccess` extension parameter with a NOA value of UNKNOWN.

ExtensionParameter universalAccess (end)

Restrictions

The format of `universalAccess` extension parameter is dependent on the CAIN_PROTOCOL_VERSION parameter in table CAINPARM. When the CAIN_PROTOCOL_VERSION is set to V1 or higher, the `universalAccess` extension parameter may contain up to 24 digits encoded in AIN format. If the CAIN_PROTOCOL_VERSION is set to V0, the `universalAccess` extension parameter is limited to 10 digits with no numbering plan or nature of address (NOA) information.

Extension parameter definition

This extension parameter is used for SS7 Global-IMTs to identify the universal translation scheme that is in effect when the call triggers.

Message types

The following TDP-Request messages support the `univIdx` extension parameter:

- **`O_Feature_Requested`**
- **`Info_Collected`**
- **`Info_Analyzed`**

Usage

Optional

Range of values

0–15 (NIL, AC, PX, CT, FA, OFC, DN, AM, FT, CC, NSC, CTY, NN, VPN)

Population rules

None

Note: Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for encoding information.

Restrictions

None

Outgoing IN/1 message parameters

ATTENTION
Outgoing messages require the CAIN0100 SOC option in order to function. Refer to Volume 5, “NetworkBuilder tools,” for more information.

Table 9-1 lists the parameters that the switch may send to the SCP within a TCAP outgoing IN/1 message.

Note: Outgoing message parameters and extension parameters are handled differently for LNP and AXXESS agents. Refer to the *UCS DMS-250 Local Number Portability Application Guide* or *UCS DMS-250 CAIN/FlexDial Interactions* for further information.

Table 9-1
Outgoing IN/1 message parameters

Parameter	Usage
`ConnectTime`	Optional
`Digits` (`CalledPartyNumber`)	Optional
`Digits` (`CallingPartyNumber`)	Required
`Digits` (`LATA`)	Required
`EchoData`	Required
`ErrorCode`	Optional
`Originating StationType`	Required
`PackageTypeIdentifier`	Required
`ProblemCode`	Required
`ProblemData`	Required

—continued—

Table 9-1
Outgoing IN/1 message parameters (continued)

Parameter	Usage
ServiceKey	Required
StandardUserErrorCode	Optional
TerminationIndicators	Required
—end—	

When an outgoing message is built, the switch formats the message and sends it to the message encoder. Once encoded, the message is sent to the SCP.

Fatal application errors

Note: Fatal application errors are handled differently for LNP and AXXESS agents. Refer to the *UCS DMS-250 Local Number Portability Application Guide* and *UCS DMS-250 CAIN/FlexDial Interactions*.

For more information on fatal application errors, refer to Chapter 1, “TCAP messaging,” Table 1-3, or each specific parameter within this chapter.

Nonfatal application errors

For more information on nonfatal application errors, refer to Chapter 1, “TCAP messaging,” Table 1-4, or each specific parameter within this chapter.

Associated logs

CAIN101, CAIN200, CAIN201, CAIN903, VAMP201, VAMP202, VAMP203, VAMP601, VAMP602, VAMP603

Associated OMs

TFREE533, VAMPACG

Parameter definition

The ***ConnectTime*** parameter specifies how much time has elapsed since the call was answered (in minutes, seconds, and tenths of seconds).

Message types

The following message supports the ***ConnectTime*** parameter:

- **`Termination_Information`**

Usage

Optional.

Range of values

The range of values for the ***ConnectTime*** parameter is as follows:

- 0–99999 minutes
- 0–59 seconds
- 0–9 for tenths of seconds
- Sign field is 0 to F (hexadecimal)

Population rules

The Minutes, Seconds, and Tenths fields are populated with the amount of time elapsed since the call was answered. The Sign field is populated with hexadecimal value C (no fill characters).

Restrictions

This parameter is present only if both of the following actions occur:

- the SCP requests termination information for the call (by sending a **`Termination`** message)
- the call is answered

Digits (continued)

Parameter definition

This parameter contains any digits that the switch passes to the SCP.

Message types

The following messages support the ***`Digits`*** parameter:

- **`Start`**
- **`Reject`**
- **`Report_Error`**

Usage

Required for **`Start`**. Optional for **`Reject`** and **`Report_Error`**.

Range of values

Type of Digits field:

- called party number (dialed)
- calling party number (ANI)
- caller interaction
- routing number
- LATA
- carrier

Nature of Number field:

- national (NATL)
- international (INTL)

Numbering Plan field:

- unknown (UNK)
- ISDN
- telephony numbering plan (TELE)

Encoding Scheme field:

- Binary Coded Decimal (BCD)

Number of Digits field:

- a string of digits specifying the number of digits contained in the Digits field

Digit field:

- a string of digits of type 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, #.

Population rules

None

Restrictions

None

EchoData (end)

Parameter definition

This parameter is used to match the SCP **`Termination`** message with the switch **`Termination_Information`** message.

Message types

The following message supports the ***`EchoData`*** parameter:

- **`Termination_Information`**

Usage

Required

Range of values

The range of values for this parameter is a string of 8 hexadecimal characters.

Population rules

This parameter contains a value received in the ***`EchoData`*** parameter of the corresponding Termination message from the SCP.

Restrictions

None

Parameter definition

The ***ErrorCode*** parameter specifies if error condition resulted from an unexpected data value, unexpected component sequence, unavailable network resource, or unavailable data. If termination information cannot be returned because of a customer action (for example, user abandon), the ***StandardUserErrorCode*** parameter replaces the ***ErrorCode*** parameter.

Message types

The following message supports the ***ErrorCode*** parameter:

- **`Termination_Information`**

Usage

Optional

Range of values

Error Type field:

- unexpected component sequence
- unexpected data value
- unavailable resource
- data unavailable

Population rules

None

Restrictions

None

OriginatingStationType (end)

Parameter definition

The ***OriginatingStationType*** parameter contains the ANI II digits for the call.

Message types

The following message supports the ***OriginatingStationType*** parameter:

- **`Start`**

Usage

Required

Range of values

a string of digits representing the ANI II digits for the call

Population rules

None

Restrictions

None

Parameter definition

The ***ProblemCode*** parameter identifies the reason the switch is sending a **`Reject`** message.

Message types

The following message supports the ***ProblemCode*** parameter:

- **`Reject`**

Usage

Required

Range of values

Problem Type field:

- General
- INVOKE
- RETURN RESULT
- RETURN ERROR
- Transaction Portion

Specifier field, for:

General problem type family

— Unrecognized Component

— Incorrect Component Portion

— Badly Structured Component

— Portion

INVOKE problem type family

— Duplicate INVOKE ID

— Unrecognized Operation Code

— Incorrect Parameter

— Unrecognized Correlation ID

RETURN RESULT family

— Unrecognized Correlation ID

— Unexpected RETURN RESULT

ProblemCode (end)

— Incorrect Parameter

RETURN ERROR family

— Unrecognized Correlation ID

— Unexpected RETURN ERROR

— Unrecognized Error

— Unexpected Error

— Incorrect Parameter

Transaction Portion family

— Unrecognized Package Type

— Incorrect Transaction Portion

— Badly Structured Transaction Portion

— Unrecognized Transaction ID

— Permission to Release Problem

— Resource Unavailable

Population rules

None

Restrictions

None

Parameter definition

The switch sends the ***ProblemData*** parameter when there is an error indication caused by erroneous contents within a data element.

Message types

The following message supports the ***ProblemData*** parameter:

- **Report_Error**

Usage

Required

Range of values

This parameter contains the data element identifier, the data element length, and the erroneous data element.

Population rules

None

Restrictions

None

ServiceKey (continued)

Parameter definition

The ***ServiceKey*** parameter contains the called party number for the call. Any prefixed “0” from a “0+” dialing plan are not included.

Message types

The following message supports the ***ServiceKey*** parameter:

- **`Start`**

Usage

Required

Range of values

Type of Digits field:

- called party number (dialed)
- calling party number (ANI)
- caller interaction
- routing number
- LATA
- carrier

Nature of Number field:

- national (NATL)
- international (INTL)

Numbering Plan field:

- unknown (UNK)
- ISDN
- telephony numbering plan (TELE)

Encoding Scheme field:

- Binary Coded Decimal (BCD)

Number of Digits field:

- a string of digits specifying the number of digits contained in the Digits field

Digit field:

- a string of digits of type 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, #.

Population rules

None

Restrictions

None

StandardUserErrorCode (end)

Parameter definition

This ***StandardUserErrorCode*** parameter specifies an error condition that results from a customer action.

Message types

The following message supports the ***StandardUserErrorCode*** parameter:

- **Termination_Information**

Usage

Optional

Range of values

User Error Code field:

- caller abandon
- improper caller response

Population rules

None

Restrictions

This parameter replaces the ***ErrorCode*** parameter inside the **Termination_Information** component message if:

- the SCP requested termination information from the switch by sending a **Termination** message
- the originator goes on-hook before the switch receives the response from the SCP containing the **Termination** message that is requesting terminating information from the switch

Parameter definition

This ***TerminationIndicators*** parameter contains YES or NO values for several indicators.

Message types

The following message supports the ***TerminationIndicators*** parameter:

- **Termination_Information**

Usage

Required

Range of values

NM Control List Overflow Indication (YES or NO)

Answer Indication (YES or NO)

Error Indication (YES or NO)

Population rules

The NM Control List Overflow Indication field is populated to YES only if:

- the switch receives an **ACG** message from the SCP
- the ACG control list overflows

The Answer Indication field is populated to YES only if:

- the call is answered

The Error Indication field is populated to YES only if:

- the call cannot be completed

If the user abandons the call subsequent to the receipt of a response from the SCP, and termination information has been requested, no error is indicated. The call is treated as unanswered (that is, the ***TerminationIndicators*** parameter is populated to Answer Indication=NO, Error Indication=NO).

Restrictions

None

Incoming CAIN messages

ATTENTION
Extension parameters require the CAIN0200 SOC option. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information on SOC.

When the switch receives a message from the SCP, the CAIN framework processes the response. Response processing consists of interpreting, validating, and performing instructions provided in the SCP TCAP message. These instructions direct the switch on how to proceed with the call. CAIN supports the call-related incoming messages listed in Table 10-1.

Note: SCP response messages, parameters, and extension parameters are handled differently for LNP calls; refer to *UCS DMS-250 Local Number Portability Application Guide* for more information. SCP response parameters are also handled differently for AXXESS agents; refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Table 10-1
Call-related incoming CAIN messages

Response	Description
`Acknowledge`	This message enables the SCP to respond to informational messages from the switch without requesting additional Call Party Handling operations.
`Analyze_Route`	This message instructs the switch to resume CAIN call processing, using the address, billing, and routing information provided by the SCP.
`Authorize_Termination`	This message instructs the switch to allow the indicated termination to progress through the terminating call model.
`Call_Info_To_Resource`	This message provides a response to the intermediate information received from the IP through the switch during an active STR-Connection; This message is used only in response to a `Call_Info_From_Resource` message.
`Cancel_Resource_Event`	This message instructs the switch to discontinue caller interaction, which was initiated through a `Send_To_Resource` message from the SCP, and wait for further instructions from the SCP. Refer to Volume 4, “Conversational processing,” for more information.
`Close`	This message informs the switch of a nonfatal unexpected communication error.
`Collect_Information`	In response to an `O_Mid_Call` query, this message instructs the switch to route the call using in-switch routing information and perform as if the call had not triggered a query to the SCP.
`Continue`	This message instructs the switch to route the call using in-switch routing information and perform as if the call had not triggered a query to the SCP.
`CTR_Clear` (Note 2)	This message sent from the switch to the SCP in response to a conversational `Connect_To_Resource` message, indicates the outcome of an SCP request for information. Refer to Volume 4, “Conversational processing,” for more information.
`Disconnect`	This message instructs the switch to disconnect the call and apply treatment.

Note 1: For information on error messages received from the SCP, refer to Chapter 1, “TCAP messaging,” and the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition*.
Note 2: This message is not a response message. However, it is listed in this section because the only time this message is used is when the switch has entered into conversation with the SCP.

—continued—

Table 10-1
Call-related incoming CAIN messages (continued)

Response	Description
`Disconnect_Leg`	This message directs the switch to release a specified leg involved in a call.
`Merge_Call`	This message instructs the switch to allocate a three port conference port during Call Configuration 6.
`Originate_Call`	This message enables the switch to place a stable two-party call on hold and originate an associated call to another party.
`Resource_Clear` (Note 2)	This message sent from the switch to the SCP in response to a conversational `Send_To_Resource` message indicates the outcome of an SCP request for information. Refer to Volume 4, “Conversational processing,” for more information.
`Send_To_Resource` or `Connect_To_Resource` with Play Announcement	This message instructs the switch to play an announcement or tone and disconnect.
`Send_To_Resource` `Connect_To_Resource` with Play Announcement and Collect Digits	This message instructs the switch to play an announcement and collect digits. Refer to Volume 4, “Conversational processing,” for more information.
`Send_To_Resource` or `Connect_To_Resource` with ***`DestinationAddress`*** parameter	This message instructs the switch to route to an IP. Refer to Volume 4, “Conversational processing,” for more information.

Note 1: For information on error messages received from the SCP, refer to Chapter 1, “TCAP messaging,” and the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition*.
Note 2: This message is not a response message. However, it is listed in this section because the only time this message is used is when the switch has entered into conversation with the SCP.

—end—

CAIN supports the non-call related incoming messages listed in Table 10-2.

Table 10-2
Non-call-related incoming CAIN messages

Response	Description
`ACG`	This message specifies an ACG control to be added, updated, or deleted from the ACG control list. Refer to Chapter 2, "Automatic Code Gapping," for further information on ACG.
`ACG_Global_Ctrl_Restore`	The SCP sends an **`ACG_Global_Ctrl_Restore`** message to request the removal of a large selection of controls from the ACG control list.
`Furnish_AMA_Information`	The SCP sends an **`Furnish_AMA_Information`** message to the UCS DMS-250 switch to indicate Bellcore AMA Format (BAF) modules are to be included in the switch CDR. This message is used when service specific modules or standard modules that do not correspond to an existing Automatic Message Accounting (AMA) TCAP parameter must be appended to the CDR.
`Request_Report_BCM_Event`	This message is sent by the SCP to indicate to the switch which event detection points (EDPs) should be armed to send an EDP-Request or EDP-Notification to the SCP. Either the **`Acknowledge`**, **`Analyze_Route`**, **`Collect_Information`**, **`Continue`**, **`Disconnet_Leg`**, **`Merge_Call`**, or **`Originate_Call`** call-related message must be the first component in the conversation package. Refer to Chapter 3, "Event processing," for more information.
`Send_Notification`	This message instructs the switch to send a **`Termination_Notification`** when the call is released. Refer to Chapter 5, "Termination_Notification processing" for further information on the **`Termination_Notification`** message.
Note: For information on error messages received from the SCP, refer to Chapter 1, "TCAP messaging," and the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition*.	

ACG

Use

The SCP sends an **`ACG`** message to specify an ACG control to be added, updated, or deleted from the ACG control list.

Message parameters

Table 10-3 lists the parameters that may be returned by the SCP.

Table 10-3
ACG parameters

Parameter	Usage	Definition
`ControlCauseIndicator`	Required	This parameter indicates whether the control is an SCP overload control or an SMS initiated control. It also contains the number of digits to which the control is applied.
`GapDuration`	Required	This parameter contains the length (in seconds) that an ACG control should be applied before it times out and is removed from the control list by the switch.
`GapInterval`	Required	This parameter contains the minimum length (in seconds) that the switch must wait before sending another query of the type under the control.
`TranslationType`	Required	This parameter specifies the translation type of the ACG control.
`GlobalTitleAddress`	Required	This parameter specifies the global title address of the ACG control.

Fatal application errors

None

Nonfatal application errors

Table 10-4 shows errors that can occur while the **`ACG`** parameters are being processed.

ACG (end)

Table 10-4
ACG nonfatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
an **ACG** message is received in a uni-directional package.	Conversation	CAIN101	No	None
an **ACG** message is received in a conversational package.	Conversation	CAIN101	No	None
an **ACG** message is received in a response package.	Response	CAIN101	No	None
an **ACG** message is received, either the CAIN0900 or CAIN0901 SOCs are not enabled.	Conversation	CAIN102	No	None

Associated logs

VAMP302, VAMP305, VAMP901, CAIN101, CAIN102

Associated OMs

CAINMSGR, VAMPACG

ACG_Global_Ctrl_Restore

Use

The SCP sends an **`ACG_Global_Ctrl_Restore`** message to request the removal of a large selection of controls from the ACG control list.

Message parameters

Table 10-5 lists the parameters that may be returned by the SCP.

Table 10-5
ACG_Global_Ctrl_Restore parameters

Parameter	Usage	Definition
`ACGGlobalOverride`	Required	This parameter reinitializes the ACG control list according to the parameter option provided.

Fatal application errors

None

Nonfatal application errors

Table 10-6 shows errors that may occur while the **`ACG`** parameters are being processed.

Table 10-6
ACG_Global_Ctrl_Restore nonfatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
an **`ACG_Global_Ctrl_Restore`** message is received in a uni-directional package.	Conversation	CAIN101	No	None
an **`ACG_Global_Ctrl_Restore`** message is received in a conversational package.	Conversation	CAIN101	No	None

—continued—

ACG_Global_Ctrl_Restore (end)

Table 10-6
ACG_Global_Ctrl_Restore nonfatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
an **`ACG_Global_Ctrl_Restore`** message is received in a response package.	Response	CAIN101	No	None
an **`ACG_Global_Ctrl_Restore`** message is received which indicates either the CAIN0900 or CAIN0901 SOCs are not enabled.	Conversation	CAIN102	No	None
—end—				

Associated logs

VAMP303, VAMP901, CAIN101, CAIN102

Associated OMs

CAINMSGR, VAMPACG

Use

The **`Acknowledge`** message enables the SCP to respond to informational messages from the switch without requesting additional Call Party Handling operations.

Message parameters

Table 10-7 lists the parameters that may be returned by the SCP.

Table 10-7
Acknowledge parameters

Parameter	Usage	Definition
`CsID`	Optional	This parameter specifies to which call segment the message applies.
`ExtensionParameter`	Optional	Extension parameters are not supported for this message.

Fatal application errors

None

Nonfatal application errors

None

Associated logs

VAMP901

Associated OMs

CAINMSGR

Analyze_Route

Use

The `Analyze_Route` message instructs the switch to resume CAIN call processing at the **Analyze_Information** PIC, using the address, billing, and routing information provided by the SCP. There are two types of routing: direct termination and standard routing.

Note 1: The SCP may include the `Request_Report_BCM_Event` non-call related component with the `Analyze_Route` message in a conversational TCAP package. Refer to chapter 3, "Event processing," for more information.

Note 2: The `Analyze_Route` message is used by the SCP to perform boomerang reorigination or to route the call after digits have been collected through the `Connect_To_Resource` message.

Note 3: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for information on `Analyze_Route` messages for LNP calls.

Terminology

The following terms are used to define CAIN routing:

- **Route indexes** are determined by the data provided in an `Analyze_Route` response or through normal translations. The index points to the appropriate table that identifies a routing list.
- **Route lists** contain the routes to be attempted by the switch when establishing the call. Route lists are provisioned in tables, such as: TANDMRTE, TERMRTE, HNPACONT, FNPACONT, CTRTE, and OFRT. Each route list can contain multiple routes.
- **Routes** typically consist of a route selector, connection type, and common language location identifier (CLLI).
- **CAIN routing parameters** are provided in an `Analyze_Route` response. The parameters are: ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, ***SecondAlternateTrunkGroup***, ***Carrier***, ***AlternateCarrier***, ***SecondAlternateCarrier***, ***CalledPartyID***, and the ***GenericAddressList*** parameter's `OverflowRoutingNo`.

  Note 1: Parameters ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, and ***SecondAlternateTrunkGroup*** are used to identify direct termination route indexes into tables TANDMRTE and TERMRTE.

  Note 2: Standard routing parameters, ***Carrier***, ***AlternateCarrier***, and ***SecondAlternateCarrier*** are used to identify a route by obtaining an STS value through table CICROUTE.

Note 3: Standard routing parameters, ***CalledPartyID*** and/or the `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs), and ***GenericAddressList*** parameter's `OverflowRoutingNo` require in-switch translations to derive the route index. One route index is calculated for each parameter.

- **Serving translation scheme (STS)** extension parameters are provided in an **`Analyze_Route`** response along with CAIN routing parameters. Each CAIN routing parameter has an associated STS extension parameter. Default STS extension parameters are also datafilled in table CAINXDFT to be used when the SCP fails to return the associated STS extension parameter with the CAIN routing parameter. Additionally, the original STS derived prior to the query message may be used when the STS extension parameter is not returned from the SCP, and is not datafilled in table CAINXDFT. Figure 10-1 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter:

Figure 10-1
Precedence order for STS extension parameters

STS Extension Parameters / **Route Parameters Returned by the SCP**	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	**table CAINXDFT** PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	**Pre Query STS**
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
`OverflowRoutingNo`				3	1				4	2	5

LEGEND

`PRISTS (primaryTrunkGroupSTS)`	– STS to be used with the ***PrimaryTrunkGroup*** parameter
`ALTSTS (alternateTrunkGroupSTS)`	– STS to be used with the ***AlternateTrunkGroup*** parameter
`SALTSTS (secondAlternateTrunkGroupSTS)`	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
`STS (servTranslationScheme)`	– STS to be used with the ***CalledPartyID*** parameter
`OVFLSTS (overflowRoutingNo)`	– STS to be used with the `OverflowRoutingNo` parameter

Analyze_Route (continued)

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

- **Standard route advance –** The switch attempts the first route in route list, and when unable to terminate using the route, attempts to establish the call using the next route in the route list.
- **CAIN routing parameter advance –** The switch attempts to route according to data provided in the next untried CAIN routing parameter. When unable to establish the connection using the data, call processing advances to the next CAIN parameter to derive a new route list and attempts to route the call.

An **`Analyze_Route`** message contains the routing information used to terminate the call. The **`Analyze_Route`** message may contain the following parameters to derive routing indexes. The precedence is as follows:

1. Direct termination routing using the ***`PrimaryTrunkGroup`*** parameter.
2. Direct termination routing using the ***`AlternateTrunkGroup`*** parameter.
3. Direct termination routing using the ***`SecondAlternateTrunkGroup`*** parameter.
4. Standard routing using the ***`Carrier`*** parameter.
5. Standard routing using the ***`AlternateCarrier`*** parameter.
6. Standard routing using the ***`SecondAlternateCarrier`*** parameter.
7. Standard routing using the ***`CalledPartyID`*** parameter and/or the `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs).
8. Standard routing using the ***`GenericAddressList`*** parameter's `OverflowRoutingNo`.

Note: Direct termination routing does not analyze dialed addresses.

Route advancing

Each of these routing choices leads to the determination of a route list which is used to attempt termination. If translations are not successful with individual routing parameters, CAIN provides the ability to continue through each of the received indexes without further consultation of the SCP or trigger tables.

Route advancing can be performed and controlled either through the provisioning of *Network_Busy*, *O_Called_Party_Busy*, and *O_No_Answer*, or through standard route advancing. Following are descriptions of the busy conditions which are considered by NetworkBuilder processing:

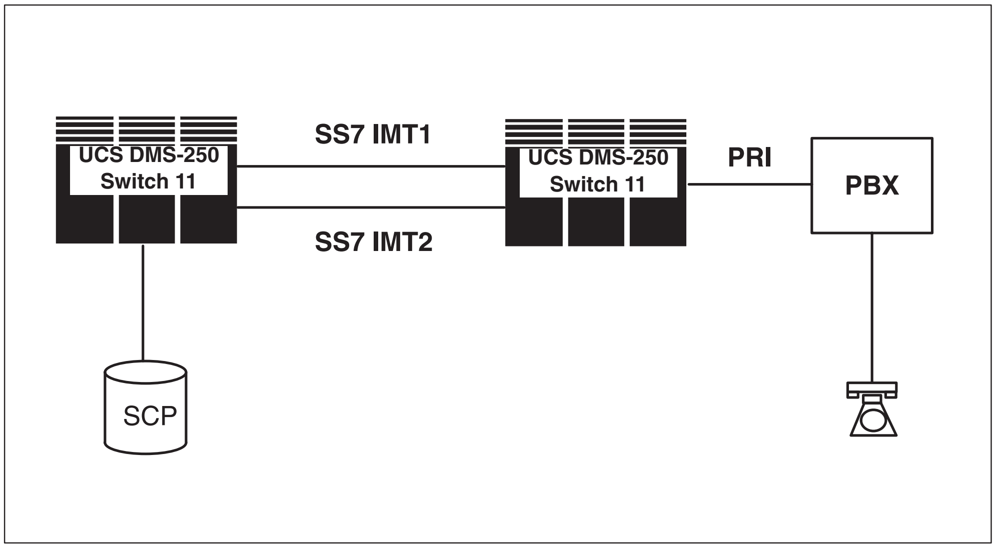
- Route Busy – Occurs when attempting to route a call over a trunk group which is busy at the SSP which queried.
- Network Busy – Occurs in each of the following scenarios:
  - — When the Route Busy condition occurs and no additional routes are available at the querying SSP.
  - — The querying SSP receives a release cause from the terminating trunk indicating that a route was busy at a switch other than the SSP. This requires PRI or SS7 connectivity.
- User Busy – Occurs in each of the following scenarios:
  - — The querying SSP attempts to route to a DAL or AXXESS trunk with ONNETTRK = Y which is busy. If ONNETTRK = N, the Route Busy condition is detected.
  - — The querying SSP receives a release cause from the terminating trunk indicating that the end user was busy. This requires PRI or SS7 connectivity.
- No Answer – Occurs when the No Answer timer has expired on the querying SSP.

Encountering the Network Busy, User Busy, and No Answer scenarios can lead to encountering the Route busy scenario when multiple route choices are available. Thus, either of the Network Busy, User Busy, or No Answer scenarios could be encountered and acted on according to the detection point evaluation resulting in a route advancing action. When this route attempt is unable to leave the querying switch, the Route Busy condition is then detected again, and route advancing could occur.

Analyze_Route (continued)

Using the following Network Configuration diagram, an example of each of the Busy conditions is given.

Figure 10-2
Network Configuration



1 Route Busy – Occurs when the SSP is attempting to route to the phone, but the SS7 IMT1 trunk is busy.

2 Network Busy – Occurs when the SSP is attempting to route to the phone and both SS7 IMT trunks are busy.

3 User Busy – Occurs when the SSP is attempting to route to the phone, but the phone is busy.

4 No Answer – Occurs when there is successful termination to the phone, but there is no answer detected during the time allotted by the No Answer timer.

The action associated with *Network_Busy*, *O_Called_Party_Busy*, and *O_No_Answer* may direct the call to route advance based on the evaluation of the current routing condition at the time the detection point is encountered.

- If there are unattempted route list choices or CAIN routing parameters, the routing condition criteria is RTEAVAIL.
- If there are no unattempted route list or CAIN routing parameters, the routing condition criteria is RTESDONE.

- If there are additional route list or CAIN routing parameters and a release cause of TERM_RESOURCE_UNAVAILABLE was received from the network, the routing condition criteria is TERMRTE_GNCT.

The type of route advancing performed depends on the action associated with the detection point, either NEXTRTE or NEXTCNRTE.

An action of NEXTRTE causes the current route list to be completed and if possible, to route advance through the CAIN routing parameters in the following order:

— ***PrimaryTrunkGroup***

— ***AlternateTrunkGroup***

— ***SecondAlternateTrunkGroup***

— ***Carrier***

— ***AlternateCarrier***

— ***SecondAlternateCarrier***

— ***CalledPartyID*** and/or the `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs).

— ***GenericAddressList*** parameter's `OverflowRoutingNo`.

An action of NEXTCNRTE causes the call to stop attempting to process the current route list and proceed to the next CAIN routing parameter. However, the carrier parameters and the called party are treated as a single routing choice. Therefore, the parameters are processed in this order.

— ***PrimaryTrunkGroup***

— ***AlternateTrunkGroup***

— ***SecondAlternateTrunkGroup***

— ***Carrier***, ***AlternateCarrier***, ***SecondAlternateCarrier*** and ***CalledPartyID*** and/or the `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs).

— ***GenericAddressList*** parameter's `OverflowRoutingNo`.

Since the routing condition criteria is part of the key for determining the appropriate action for the detection points, only one action may be associated with each the RTEAVAIL and RTESDONE criteria. There is an implied dependency on this provisioning limitation which links the provisioned action for the RTEAVAIL tuple to the determination of the RTESDONE criteria.

Analyze_Route (continued)

When the RTEAVAIL tuple action is NEXTRTE, the RTESDONE condition can only be achieved by completely exhausting all route possibilities (route list and CAIN). When the RTEAVAIL tuple action is NEXTCNRTE, the RTESDONE criteria is achieved once each of the provided CAIN routing parameters are attempted (even though additional route list choices may still be available).

If multiple CAIN groups are provisioned with NEXTRTE and NEXTCNRTE actions for a particular trigger, RTESDONE will only be encountered when all standard and CAIN routing parameters are exhausted.

Direct termination routing

The CAIN platform allows the switch to serve as an originating, intermediary, or terminating switch. The originating switch has access to the following:

- originating switch-based virtual private network (VPN) data
- connected switching facilities data (through CCS7 TCAP transactions)

When a call originates from an agent carrying a CAIN call, the originating switch passes the called number to the SCP through a CCS7 TCAP transaction. The SCP returns a destination switch identifier (SWID) and a destination trunk group number in the ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, and ***SecondAlternateTrunkGroup*** parameters.

Note: Every switch in the network must be assigned a unique SWID.

The switch performs direct termination routing through the use of two tables: TANDMRTE (Tandem Routing) and TERMRTE (Termination Routing). Table TANDMRTE directs the call from switch to switch until the terminating switch in the network is reached. Then, table TERMRTE directs the call to the terminating route index.

Note: The originating switch may also be the terminating switch, in which case table TANDMRTE is not used.

Note: SS7 Global-IMTs should use the ADDR dialing plan for direct termination routing through table TANDMRTE.

Limitations and restrictions

Direct termination through a tandem switch (using table TANDMRTE) is only supported over IMTs that are datafilled to support UCS-to-UCS ISUP protocol. When any other protocol is used, retranslation occurs at the tandem switch.

- Direct terminating routing to a terminating ISUP92/Q.767 trunk is supported through table TERMRTE only.

Table routing

The **`Analyze_Route`** message may contain the following routing information in ***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, or ***`SecondAlternateTrunkGroup`*** parameters used for direct termination routing:

- destination switch ID (SWID)
- destination trunk group number

Call processing checks the destination SWID against the current switch's SWID (located in table OFCVAR, parameter ORIG_SWITCH_ID). If the values are equal, the originating switch is also the terminating switch and table TERMRTE directs the routing.

However, if the SWID values are not equal, the destination SWID indexes table TANDMRTE to access the intermediary routing switch. The destination SWID and trunk group information is encoded into an ISUP generic digits parameter (GDP) of the initial address message (IAM) delivered to the intermediary switch.

The Called Party Number of the IAM may contain one of the following (in order of precedence):

1 appropriate outpulse number for the selected route choice is from the
   - ***`PrimaryTrunkGroup`***, contains the ***`OutpulseNumber`***
   - ***`AlternateTrunkGroup`***, contains the `AlternateOutpulseNo` (from the ***`GenericAddressList`***, when available) or the ***`OutpulseNumber`***
   - ***`SecondAlternateTrunkGroup`***, contains the `SecondAlternateOutpulseNo` (from the ***`GenericAddressList`***, when available) or the ***`OutpulseNumber`***

2 ***`CalledPartyID`*** (returned from the SCP)

Note: If an SS7 Global-IMT has not used the ADDR dialing plan, additional digits are included, based on the dialing plan used. Refer to the *UCS DMS-250 Global Application Guide* for more information on outpulsing digits for SS7 Global-IMTs.

3 original value of ***`CalledPartyID`*** (delivered to the SCP in the query)

When the tandem switch receives the IAM message, the SWID in the generic digits parameter (GDP) is checked against this switch's ORIG_SWITCH_ID parameter. If the destination SWID does not equal

Analyze_Route (continued)

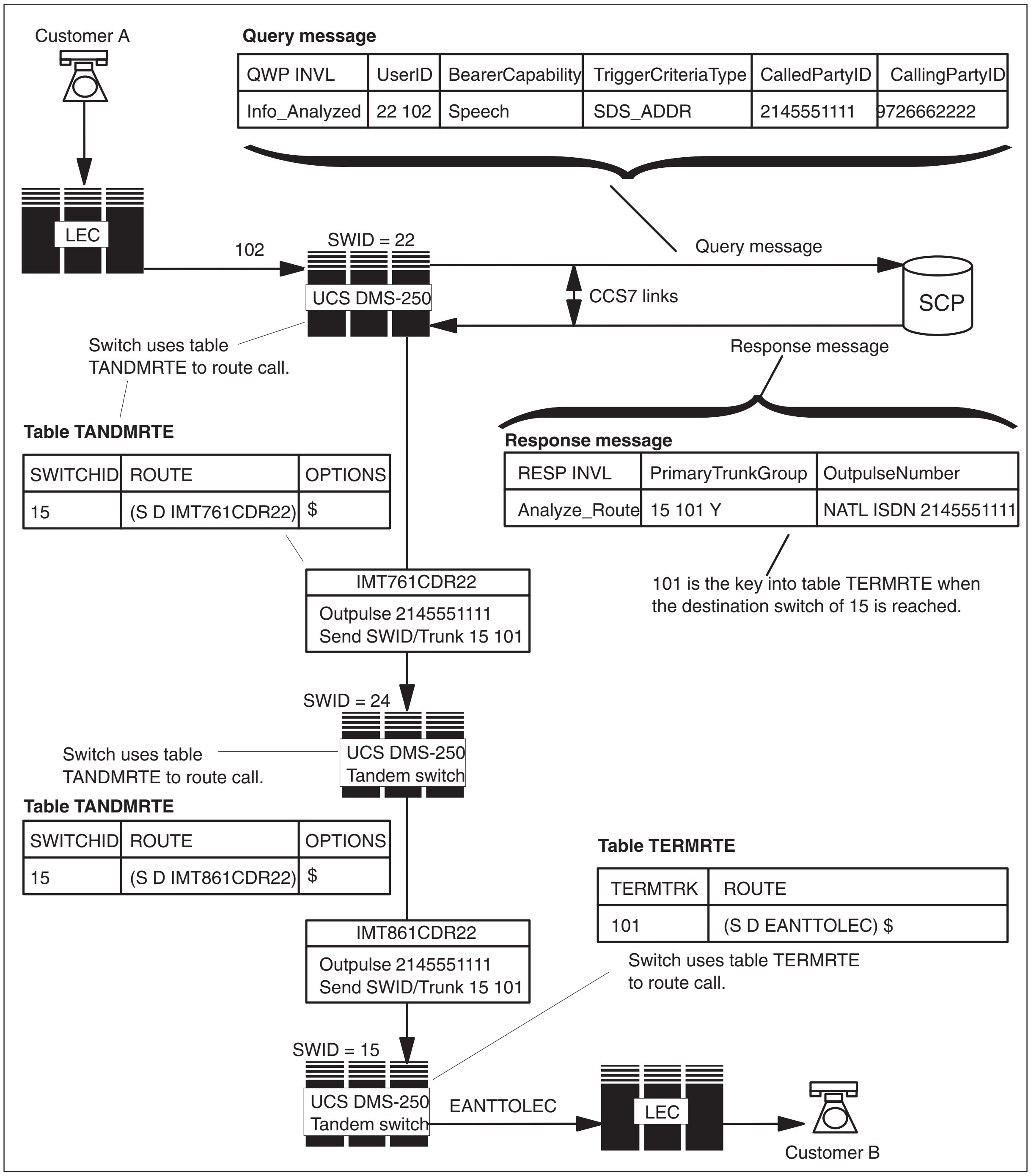
tandem switch’s identifier, table TANDMRTE is again indexed and the call is routed to another tandem switch.

Tandem routing continues until the terminating SWID equals the tandem switch’s ORIG_SWITCH_ID. Table TERMRTE is then indexed and the call is routed to the appropriate agency. The digits outpulsed to the terminating agency are the Called Party Number from the IAM.

Figure 10-3 shows how tables TERMRTE and TANDMRTE perform direct termination routing.

Analyze_Route (continued)

Figure 10-3
Direct termination routing example



Query message

QWP INVL	UserID	BearerCapability	TriggerCriteriaType	CalledPartyID	CallingPartyID
Info_Analyzed	22 102	Speech	SDS_ADDR	2145551111	9726662222

Response message

RESP INVL	PrimaryTrunkGroup	OutpulseNumber
Analyze_Route	15 101 Y	NATL ISDN 2145551111

Table TANDMRTE

SWITCHID	ROUTE	OPTIONS
15	(S D IMT761CDR22)	$

Table TANDMRTE

SWITCHID	ROUTE	OPTIONS
15	(S D IMT861CDR22)	$

Table TERMRTE

TERMTRK	ROUTE
101	(S D EANTTOLEC) $

Analyze_Route (continued)

Figure 10-4 and the following steps provide an additional direct termination routing example.

1 A CAIN-provisioned call triggers at Switch 11 and queries the SCP.

2 The SCP returns an **`Analyze_Route`** response containing ***`PrimaryTrunkGroup`*** and ***`CalledPartyID`***.

3 Switch 11 decodes the ***`PrimaryTrunkGroup`*** in order to perform direct termination routing. The switch compares the SWID returned in the **`Analyze_Route`** response (77) to the current SWID (11). Since the numbers don't match, call processing accesses tuple 77 of table TANDMRTE. Call processing routes to the first index (IMT55A) and delivers the required SWID (77) in an IAM.

4 Switch 55 receives the IAM and compares the current SWID (55) to the received SWID (77). Since the SWIDs don't match, call processing accesses tuple 77 of table TANDMRTE. Call processing routes to the index (77A) and delivers the required SWID (77) in the Generic Digits parameter of the outgoing IAM.

5 Switch 77 receives the IAM and compares the current SWID (77) to the received SWID (77). Now that the SWIDs match, call processing accesses tuple 24 (the trunk number received in the **`Analyze_Route`** message) and routes to DAL77CUST.

IN/1 SACREMOT

The switch may send an IN/1 SACREMOT query if the following conditions are true:

- the switch has received an **`Analyze_Route`** query message with a ***`ChargeNumber`*** parameter and a `cainGroup` extension parameter
- the call is in the **Analyze_Information** PIC
- reevaluation of the ***Info_Analyzed*** triggers has not been performed
- table CAINXDFT has the DEFPARMS field set to IN1_REQUERY for the CAIN group specified by the **`Analyze_Route`** `cainGroup` extension parameter

Relevant **`Analyze_Route`** parameters are used to populate the SACREMOT query. NetworkBuilder controls neither the encoding nor the decoding of the IN/1 SACREMOT query. For Further information on SACREMOT functions, refer to the *UCS DMS-250 Transaction Capabilities Application Part (TCAP) Application Guide*.

Analyze_Route message (continued)

Figure 10-4
Direct termination routing example

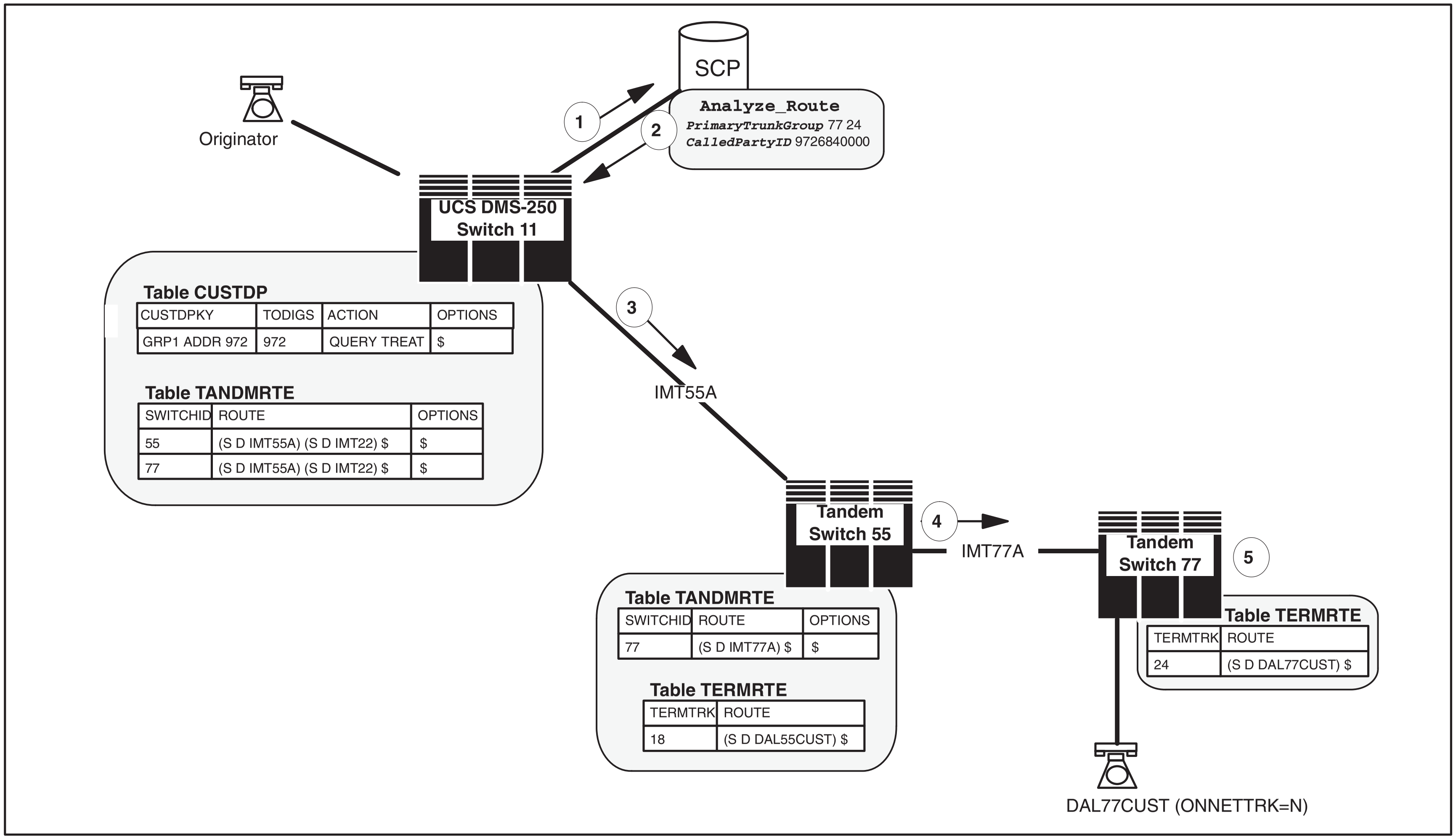


Table CUSTDP

CUSTDPKY	TODIGS	ACTION	OPTIONS
GRP1 ADDR 972	972	QUERY TREAT	$

Table TANDMRTE

SWITCHID	ROUTE	OPTIONS
55	(S D IMT55A) (S D IMT22) $	$
77	(S D IMT55A) (S D IMT22) $	$

Table TANDMRTE

SWITCHID	ROUTE	OPTIONS
77	(S D IMT77A) $	$

Table TERMRTE

TERMTRK	ROUTE
18	(S D DAL55CUST) $

Table TERMRTE

TERMTRK	ROUTE
24	(S D DAL77CUST) $

Analyze_Route (continued)

Standard routing

Note: LNP call routing is handled differently. Refer to *UCS DMS-250 Local Number Portability Application Guide* for more information.

The switch normally routes a call using standard routing where call processing analyzes the dialed address along with other call data (such as STS and call type) to determine a route index.

When the switch receives an **`Analyze_Route`** from the SCP containing standard parameters, call processing attempts to route the call using the following precedence:

1 ***`Carrier`***

2 ***`AlternateCarrier`***

3 ***`SecondAlternateCarrier`***

4 ***`CalledPartyID`*** and/or the `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs)

5 ***`GenericAddressList`***’s `OverflowRoutingNo`

When standard routing parameters are included in the **`Analyze_Route`** message, the following steps are performed to derive the routing indexes:

- translations index is determined
- address is determined
- address is analyzed based on determined translations index
- ***Info_Analyzed*** TDP is re-evaluated

Determine translations index

The translations index is determined as follows:

- for parameters ***`Carrier`***, ***`AlternateCarrier`***, and ***`SecondAlternateCarrier`***, the `servTranslationScheme` value is obtained from table CICROUTE
- for the ***`CalledPartyID`*** and/or `servTranslationScheme` the value is obtained by using the following precedence:
  - — the SCP-returned `servTranslationScheme` value is used, if available
  - — the default `servTranslationScheme` value is used, if provisioned, in table CAINXDFT, for the current CAIN group
  - — in-switch determined `servTranslationScheme` value is used

- for the ***GenericAddressList*** parameter's `OverflowRoutingNo` the value is obtained by using the following precedence:
  - — the SCP-returned `servTranslationScheme` value is used if available
  - — the default `servTranslationScheme` value is used, if provisioned, in table CAINXDFT, for the current CAIN group
  - — in-switch determined `servTranslationScheme` value is used

Note: For SS7 Global-IMTs the following precedence is used:

- — if a `univIdx` extension parameter was returned in the **`Analyze_Route`**, call processing uses this new extension parameter for translations
- — if the SCP does not return a `univIdx` extension parameter, call processing uses the default `univIdx` from table CAINXDFT (if available)
- — if a default `univIdx` is not provisioned, the switch uses the `univIdx` related to the call before the query was sent

Determine address

The address is determined as follows:

- for parameters ***Carrier***, ***AlternateCarrier***, and ***SecondAlternateCarrier***, the SCP-returned ***CalledPartyID*** value is used if available; otherwise the in-switch determined ***CalledPartyID*** is used
- for the ***CalledPartyID*** and/or `servTranslationScheme`, the SCP-returned ***CalledPartyID*** value is used if available; otherwise the in-switch determined ***CalledPartyID*** is used
- for the ***GenericAddressList*** parameter's `OverflowRoutingNo`, the SCP-returned ***GenericAddressList*** parameter's `OverflowRoutingNo` value is used

Analyze address

The translation table is selected based on the nature of address (NOA) of the address as follows:

- national (NATL) NOA – table HNPACONT is used
- international (INTL) NOA – table CCTR is used
- international partitioned (IP) NOA – tables STS2CCDB, CTHEAD, CTCODE, CTRTE are used

Analyze_Route (continued)

Info_Analyzed TDP Re-evaluation

When a successful translation has been performed, the ***Info_Analyzed*** TDP is re-evaluated using the translated address (XLAADDR) digit criteria.

Note: Refer to Volume 1, Chapter 5, "Analyze_Information PIC," for more information on re-evaluation of ***Info_Analyzed***.

Message parameters

Table 10-8 lists the parameters that may be returned by the SCP.

Note: **`Analyze_Route`** messages are handled differently for LNP calls. Refer to *UCS DMS-250 Local Number Portability Application Guide* for more information.

Table 10-8
Analyze_Route parameters

Parameter	Definition
ChargeNumber	This parameter contains the billing number. See "Parameter processing" for further details on this parameter.
CallingPartyID	This parameter contains the address used for caller identification purposes. See "Parameter processing" for further details on this parameter.
ChargePartyStationType	This parameter contains the information digits for the call. See "Parameter processing" for further details on this parameter.
CalledPartyID	This parameter contains the translated address of the called party.
OutpulseNumber	This parameter contains the digits to be outpulsed when the ***PrimaryTrunkGroup*** numbered outpulse flag is 0. This number is used when the outpulse number associated with ***AlternateTrunkGroup*** or ***SecondAlternateTrunkGroup*** is not available.

Note 1: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Note 2: Refer to Chapter 8, "Send_Call PIC," Chapter 9, "O_Alerting PIC," and "O_Active and O_Suspended PIC" for more information about reorigination.

—continued—

Table 10-8
Analyze_Route parameters (continued)

Parameter	Definition
OverflowBillingIndicator	This parameter provides customized billing information for the UCS DMS-250 CDR.
PrimaryTrunkGroup	This parameter contains the SCP-determined primary route index for table TERMRTE or TANDMRTE.
AlternateTrunkGroup	This parameter contains the SCP-determined alternate route index for table TERMRTE or TANDMRTE.
SecondAlternateTrunkGroup	This parameter contains the SCP-determined second alternate route index for table TERMRTE or TANDMRTE.
Carrier	This parameter contains the carrier selection information (CSI) and the carrier identification code (CIC) to route the call. See “Parameter processing” for further details on this parameter.
AlternateCarrier	This parameter contains the alternate carrier selection information and the carrier identification code (CIC) to route the call. See “Parameter processing” for further details on this parameter.
SecondAlternateCarrier	This parameter contains the second alternate carrier selection information and the carrier identification code (CIC) to route the call. See “Parameter processing” for further details on this parameter.
AMAAlternateBillingNumber	This parameter identifies an alternate billing number to which the AIN service should be billed. The switch uses the contents of this parameter to populate the ALTBILL CDR field.
AMALineNumber	This parameter contains digits used to populate a CDR.
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	This parameter contains digit strings to be populated into a CDR.

Note 1: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Note 2: Refer to Chapter 8, “Send_Call PIC,” Chapter 9, “O_Alerting PIC,” and “O_Active and O_Suspended PIC” for more information about reorigination.

—continued—

Analyze_Route (continued)

Table 10-8
Analyze_Route parameters (continued)

Parameter	Definition
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.
servTranslationScheme	This extension parameter contains a serving translation scheme.
callType	This extension parameter contains the network call type.
satRestriction	This extension parameter indicates the call should not terminate to a satellite-based trunk.
classOfSvc	This extension parameter contains an index into table MULTICOS for COS screening.
callBranding	This extension parameter contains an announcement or tone to be played prior to routing.
billSequenceNumber	This extension parameter contains 32 bits of SCP-defined billing data that is stored in the CDR.
cainGroup	This extension parameter contains the group number (field GRPNUM, table CAINGRP) for the CAIN group to be used for SCP subscription.
reorigAllowed (Note)	This extension parameter contains an integer indicating the number of successive reoriginations allowed, 0 indicates that reorigination is not allowed.
univIdx	This extension parameter contains the universal translation scheme to use for the call (for SS7 Global IMTs only).
netinfo	This extension parameter contains business group information to be sent in the outgoing ISUP IAM.

Note 1: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Note 2: Refer to Chapter 8, “Send_Call PIC,” Chapter 9, “O_Alerting PIC,” and “O_Active and O_Suspended PIC” for more information about reorigination.

—continued—

Table 10-8
Analyze_Route parameters (continued)

Parameter	Definition
`callCtrl`	This extension parameter contains a call control value enhancing control over TDP evaluation for the call in progress. ***Note:*** The `callCtrl` extension parameter is ignored if received in a conversation package.
`primaryTrunkGroupSTS`	This extension parameter contains the STS value associated with the returned ***`PrimaryTrunkGroup`*** parameter.
`alternateTrunkGroupSTS`	This extension parameter contains the STS value associated with the returned ***`AlternateTrunkGroup`*** parameter.
`secondAlternateTrunkGroup STS`	This extension parameter contains the STS value associated with the returned ***`SecondAlternateTrunkGroup`*** parameter.
`overflowRoutingNoSTS`	This extension parameter contains the STS value associated with the ***`GenericAddressList`*** parameter's returned `OverflowRoutingNo`.
`accountCode`	This extension parameter contains the account code collected from in-switch translations or by the *O_Feature_Requested* trigger.
`pinDigits`	This extension parameter contains the collected PIN code, when available, or a PIN collected at *O_Feature_Requested*. It also contains digits collected during conversational digit collection.
`billingNumber`	This extension parameter contains a non-standard charge number (for example, CARD, AUTH, ACCT, PIN, or N00). See "Parameter processing" for further details on this parameter.
`GenericAddressList`	

Note 1: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Note 2: Refer to Chapter 8, "Send_Call PIC," Chapter 9, "O_Alerting PIC," and "O_Active and O_Suspended PIC" for more information about reorigination.

—continued—

Analyze_Route (continued)

Table 10-8
Analyze_Route parameters (continued)

Parameter	Definition
`AlternateOutpulseNo`	This extension parameter contains the digits to be outpulsed when the ***`AlternateTrunkGroup`*** number to outpulse field is 0.
`SecondAlternateOutpulseNo`	This extension parameter contains the digits to be outpulsed when the ***`SecondAlternateTrunkGroup`*** number to outpulse field is 0.
`OverflowRoutingNo`	This extension parameter contains an overflow routing number
`DialedNoInwardService`	This extension parameter contains a DNIS value specific to the Dialed Number Inward Service specified.
`PortedDialedNo`	This extension parameter contains the original called party address of a successful LNP query.
`ForwardCallIndicator`	This extension parameter indicates that an LNP check has been performed at the SCP.

Note 1: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Note 2: Refer to Chapter 8, “Send_Call PIC,” Chapter 9, “O_Alerting PIC,” and “O_Active and O_Suspended PIC” for more information about reorigination.

—end—

Parameter processing

Note: **`Analyze_Route`** message parameter processing is handled differently for LNP calls. Refer to *UCS DMS-250 Local Number Portability Application Guide* for more information.

CAIN call processing uses the steps listed in Table 10-9 to process the parameters in an **`Analyze_Route`** message:

Table 10-9
Parameter processing steps

Step	Action
1	Standard parameters and extension parameters are validated and the appropriate call processing and billing values are updated.
2	When the `classofSvc` extension parameter is provided, multi-COS screening is performed. ***Note 1:*** If multi-COS screening fails, the call is treated (usually COSX). ***Note 2:*** Multi-COS screening may route the call through standard routing.
3	If the `callBranding` extension parameter is present in the SCP response, a tone or announcement is played before routing. ***Note:*** Branding is also performed if a default `callBranding` parameter is datafilled in table CAINXDFT.
—end—	

If the ***`CallingPartyID`*** parameter is included in the **`Analyze_Route`** message, it is used to populate the CLGPTYNO CDR field, and its value is passed to the terminating agent as the calling party identification, subject to in-switch restrictions on delivery of that identification.

When carrier parameters (***`Carrier`***, ***`AlternateCarrier`***, and ***`SecondAlternateCarrier`***) are included in the **`Analyze_Route`** message, and if no trunk group parameters (***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, and ***`SecondAlternateTrunkGroup`***) are included in the message or the switch cannot route the call based on the trunk group information, the call will be routed based on the carrier information.

If the ***`ChargeNumber`*** parameter is included in the **`Analyze_Route`** message, it overwrites the ***`ChargeNumber`*** information the switch has currently. The ***`ChargeNumber`*** is used as the MF ANI or the ISUP Charge Number for populating the ANISP CDR field, and for subsequent queries acting on the same call.

If the ***`ChargePartyStationType`*** parameter is included in the **`Analyze_Route`** message, it is used as the MF II digits for outpulsing, or as

Analyze_Route (continued)

the ISUP OLI parameter for constructing the IAM message for SS7 trunks. If the ***ChargePartyStationType*** parameter is not included and the trigger associated with the response does not have line attributes that apply, the switch determines the value of this parameter from a pseudo ANI value associated with the call, if present.

If the billingNumber extension parameter is included in the **Analyze_Route** message, it is used to populate the BILLNUM CDR field. The digits used to populate this extension parameter may be used by call processing later in the call for populating future queries (for example, populating the outgoing billingNumber extension parameter).

If ***OutpulseNumber*** parameter is included in the **Analyze_Route** message, it overrides the functionality on the switch to buffer an NXX number on origination and place it in a Generic Address parameter of the outgoing Initial Address message (for SS7 termination agencies) or send the number as an address (for PTS and PRI terminating agents).

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-10 shows fatal application errors that can occur after an **Analyze_Route** SCP response is returned:

Note: Fatal application errors are handled differently for AXXESS agents. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Table 10-10
Analyze_Route fatal application errors

Error type	Definition	Log generated	Reported to SCP?	Error action performed
Unexpected message sequence	The **Request_Report_BCM_Event** component is missing from the conversation package.	CAIN200	Yes	ERRACT in trigger table (Note)
Erroneous data value	SCP sends one or more routing parameters, but CAIN call processing is unable to identify a route index.	CAIN200	Yes	ERRACT in trigger table (Note)

Note: The OFFCCODE trigger table does not provision ERRACT, it defaults to the ROUTE error action. Trigger tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision an error action. AINF is used for all other triggers.

Nonfatal application errors

Table 10-11 shows nonfatal errors that can occur while the `Analyze_Route` parameters are being processed.

Table 10-11
Analyze_Route nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle.	CAIN102	No	Extension parameters are ignored
Invalid direct termination parameters	CAIN300	No	Parameters are ignored

Associated logs

CAIN100, CAIN102, CAIN200, CAIN300, VAMP901

Note: For more information on logs, refer to the *UCS DMS-250 Logs Reference Manual*.

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Authorize_Termination

Use

The SCP sends an `Authorize_Termination` in response switch-originated `Termination_Attempt` TDP-Request message. Call processing continues the call attempting the termination which triggered the CAIN interaction.

Message parameters

Table 10-12 lists the parameters that may be returned by the SCP.

Table 10-12
Authorize_Termination parameters

Parameter	Definition
CallingPartyID	This parameter contains the address used for caller identification purposes.
DisplayText	This parameter contains display data to be sent across the network to the end user.
AMAAlternateBillingNumber	This parameter identifies an alternate billing number to which the AIN service should be billed.
AMALineNumber	This parameter contains digits used to populate a CDR.
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	This parameter contains digit strings to be populated into the CDR.
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.
billSequenceNumber	This extension parameter contains 32 bits of SCP-defined billing data that is stored in the CDR.

Fatal application errors

None

Nonfatal application errors

Table 10-13 shows nonfatal errors that can occur while the `Authorize_Termination` parameters are being processed.

Table 10-13
Authorize_Termination nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle.	CAIN102	No	Extension parameters are ignored.

Associated logs

CAIN102, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Call_Info_To_Resource

Use

The **`Call_Info_To_Resource`** message is used to provide a response to the intermediate information received from the IP through the switch during an active STR-connection. The maximum message size is 5 bytes for both for PRI and ISUP trunk type.

Message parameters

Table 10-14 lists the parameters that may be returned by the SCP:

Table 10-14
Call_Info_To_Resource parameters

Parameter	Definition
`ResourceType`	This parameter contains the value `Play announcements`, `Play announcements and collect digits`, or `FlexParameterBlock`.
`StrParameterBlock`	This parameter contains the value `AnnouncementBlock`, `AnnouncementDigitBlock`, or `FlexParameterBlock`.
`ExtensionParameter`	***`ExtensionParameter`***s require the CAIN0200 SOC option.
`amaDigits`	This extension parameter contains digit strings to be populated into one or more of the following CDR fields: PINDIGS, ACCTCD, BILLNUM, CIC, ORIGPVN, or TERMPVN.

Fatal application errors

None

Nonfatal application errors

Table 10-15 shows nonfatal errors that can occur while the **`Call_Info_To_Resource`** parameters are being processed.

Table 10-15
Call_Info_To_Resource nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle.	CAIN102	No	Extension parameters are ignored.

Associated logs

CAIN102, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Close

Use

The SCP sends a `Close` to indicate a nonfatal unexpected communication error.

Message parameters

Table 10-16 lists the parameters that may be returned by the SCP.

Table 10-16
Close parameters

Parameter	Usage	Definition
UserID	Optional	This parameter contains the caller's network identity.
BearerCapability	Optional	This parameter contains the bearer capability of the call when the message is built.
CloseCause	Optional	This parameter indicates the reason a `Close` message is sent to end a TCAP transaction between the switch and the SCP.

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-17 shows errors that can occur after a `Close` SCP response is returned:

Table 10-17
Close fatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
`Close` received	CAIN200 and CAIN201	Yes	ERRACT in trigger table (Note)
`Close` received in response to a call-related message	CAIN201	Yes	ERRACT in trigger table (Note)
Note: The OFFCCODE trigger table does not provision ERRACT, it defaults to the ROUTE error action. Trigger tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision an error action. AINF is used for all other triggers.			

Associated logs

CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGR

Collect_Information

Use

The `Collect_Information` message is sent by the SCP in response to a switch-originated `O_Mid_Call` TDP-Request message. The `Collect_Information` message is used to pass the call to the dialing plan. The reorigination dialing plan enabled on the switch is enabled.

Note: The SCP may include the `Request_Report_BCM_Event` non-call related component with the `Collect_Information` message in a conversational TCAP package. Refer to Chapter 3, “Event processing,” for more information.

Message parameters

Table 10-18 lists the parameters that may be returned by the SCP.

Table 10-18
Collect_Information parameters

Parameter	Definition
CallingPartyID	This parameter contains the address used for caller identification purposes.
AMAAlternateBillingNumber	This parameter identifies an alternate billing number to which the AIN service should be billed.
AMALineNumber	This parameter contains digits used to populate a CDR.
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	This parameter contains digit strings to be populated into a CDR.
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.
cainGroup	This extension parameter contains the group number (field GRPNUM, table CAINGRP) for the CAIN group to be used for SCP subscription.

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-19 shows fatal application errors that can occur after a **`Collect_Information`** SCP response is returned.

Table 10-19
Collect_Information fatal application errors

Error type	Definition	Log generated	Reported to SCP?	Error action performed
Unexpected message sequence	The **`Request_Report_BCM_Event`** component is missing from the conversation package.	CAIN200	Yes	Switch applies AINF

Nonfatal application errors

Table 10-20 shows nonfatal errors that can occur while the **`Collect_Information`** parameters are being processed.

Table 10-20
Collect_Information nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle	CAIN102	No	Extension parameters are ignored

Associated logs

CAIN102, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Continue

Use

The SCP sends a **`Continue`** in response to a switch-originated query. The **`Continue`** directs the switch to continue processing the subscription methods at the trigger originating the query and to continue through the call model once all subscription methods are analyzed.

Note: The SCP may include the **`Request_Report_BCM_Event`** non-call related component with the **`Continue`** message in a conversational TCAP package. Refer to Chapter 3, “Event processing,” for more information.

When the switch receives a **`Continue`** message in response to a **`CTR_Clear`** message at the *Timeout* event, it allows the calling and called parties to remain connected. If the **`Continue`** message is received in a conversation package along with a **`Request_Report_BCM_Event`** message requesting the *Timeout* event, the TimeoutTimer is restarted. The timer’s duration is determined by the ***`TimeoutTimer`*** parameter received in the **`Request_Report_BCM_Event`** message, or from the value of the TIMEOUT_TIMER tuple in table CAINPARM.

The CAIN framework accepts **`Continue`** messages in response to the following outgoing messages:

- **`CTR_Clear`** at the ***O_Mid_Call*** EDP
- **`Info_Analyzed`**
- **`O_Called_Party_Busy`**
- **`O_No_Answer`**
- **`Origination_Attempt`**

Message parameters

Table 10-21 lists the parameters that may be returned by the SCP.

Table 10-21
Continue parameters

Parameter	Definition
`AMAAlternateBillingNumber`	This parameter identifies an alternate billing number to which the AIN service should be billed.
`AMALineNumber`	This parameter contains digits used to populate a CDR.

—continued—

Table 10-21
Continue parameters (continued)

Parameter	Definition
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	This parameter contains digit strings to be populated into a CDR field.
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.
billSequenceNumber	This extension parameter contains 32 bits of SCP-defined billing data that is stored in the CDR.
connectToSCU	When present, this extension parameter indicates that CAIN call processing should be terminated and call control should be passed to the programmable service node control unit.
callCtrl	This extension parameter contains a call control value enhancing control over TDP evaluation for the call in progress. ***Note:*** callCtrl is ignored if received in a conversation package.
—end—	

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-22 shows fatal application errors that can occur after a **Continue** SCP response is returned:

Continue (end)

Table 10-22
Continue fatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
Unexpected message sequence	CAIN200	Yes	ERRACT in trigger table (Note)
`Continue` received in response to `Info_Collected`, `O_Feature_Requested`, or `Network_Busy` queries	CAIN200 and CAIN201	Yes	Switch applies AINF
The `Request_Report_BCM_Event` component is missing from the conversation package	CAIN200	Yes	ERRACT in trigger table (Note)
Note: The OFFCCODE trigger table does not provision ERRACT, it defaults to the ROUTE error action. Trigger tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision an error action. AINF is used for all other triggers.			

Nonfatal application errors

Table 10-23 shows nonfatal errors that can occur while the `Continue` parameters are being processed.

Table 10-23
Continue nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle.	CAIN102	No	Extension parameters are ignored

Associated logs

CAIN102, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Connect_To_Resource

Use

The **`Connect_To_Resource`** message is sent by the SCP in response to a switch-originated **`O_Mid_Call`** TDP-Request message. The **`Connect_To_Resource`** message is used by the SCP to control the reorigination dialing plan. It instructs the switch to perform one of the following:

- play an announcement and disconnect
- play one or more announcement and collect digits (Refer to Volume 4, "Conversational processing," for more information.)
- route to an IP (Refer to Volume 4, "Conversational processing," for more information.)

Note: Digit collection and IP routing are available during conversational messaging.

The switch handles a **`Connect_To_Resource`** message in response to a **`CTR_Clear`** message at the *Timeout* event in the same manner that a **`Connect_To_Resource`** message in response to a **`Timeout`** EDP–Request is handled.

Note: CAIN does not allow a **`Request_Report_BCM_Event`** message to be received in a package with a **`Connect_To_Resource`** message. This is considered a fatal application error. In order to request events after the resource is completed, the SCP should include a **`Request_Report_BCM_Event`** message in its response to a **`CTR_Clear`** message.

The T1 timer is canceled on receipt of a message in a response package.

Play announcement

The play resource capability allows the the SCP to instruct the switch to play an announcement or tone over the voice channel to a single leg associated with a stable call, or to an entire call. Any digits that the users dial are ignored by the switch, and do not interrupt the resource. Regardless of the number of legs connected to the resource, the entire call terminates at the end of the interaction.

If the switch receives a **`Connect_To_Resource`** message and determines that it cannot play the resource because the requested resource is busy or not installed (or not implemented) on the switch, a **`CTR_Clear`** message is sent to the SCP in a conversation package with a ***`ClearCause`*** parameter value of ABORT. The call is not cleared toward the controlling leg or the passive leg.

Connect_To_Resource (continued)

If the switch receives a **`Connect_To_Resource`** message with a ***`LegID`*** provided, and determines that it cannot play the resource because the requested resource is busy or not installed (or implemented) on the switch, a **`CTR_Clear`** message is sent to the SCP in a conversation package with a ***`ClearCause`*** parameter value of ABORT. The call is not cleared toward the controlling leg or the passive leg.

A play announcement **`Connect_To_Resource`** message is transmitted in a Response package with an Invoke_Last component.

This **`Connect_To_Resource`** message contains the parameters listed in Table 10-24 when the ***`ResourceType`*** is `Play Announcement`:

Table 10-24
Play announcement parameters

Parameter	Usage	Definition
`ResourceType` (Note)	Required	This parameter contains `Play Announcement`.
`StrParameterBlock`	Required	This parameter contains an `AnnouncementBlock` encoded as an `UninterAnnounceBlock`. The `UninterAnnounceBlock` contains a Play Announcement tag and up to 3 uninterruptible announcement elements. The announcement elements contain a resource identifier (index into table CAINRSRC). ***Note 1:*** Only one announcement block is expected. ***Note 2:*** Any dialed digits are ignored.
`LegID`	Optional	When set to 0, indicates the calling party is connected to a resource. When set to 1, indicates the called party is connected to a resource.
`DisconnectFlag`	Optional	The presence of this parameter indicates the switch disconnects the call after announcement or tone is played, regardless of the ***`LegID`*** parameter value.
Note: When the ***`DestinationAddress`*** parameter is present, the ***`ResourceType`*** parameter is ignored.		

—continued—

Connect_To_Resource (continued)

Table 10-24
Play announcement parameters (continued)

Parameter	Usage	Definition
AMAAlternateBillingNumber	Optional	This parameter identifies an alternate billing number to which the AIN service should be billed.
AMALineNumber	Optional	This parameter contains digits used to populate into a CDR.
AMAslpID	Optional	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	Optional	This parameter contains digit strings to be populated into the CDR.
DestinationAddress	Optional	This parameter contains the address of an IP.
AMAMeasure	Optional	This parameter indicates that a duration time measurement is required.
ExtensionParameter	Optional	***ExtensionParameter***s require the CAIN0200 SOC option.
treatment	Optional	When present, this extension parameter indicates the treatment to be set as a result of SCP service logic. ***Note:*** After the resource identified in the **Send_To_Resource** message, any resources identified by the treatment are played.
strConnectionType	Optional	This extension parameter Indicates what type of connection protocol is to be used to establish communication between the SCP, switch, and an IP resource.
pretranslatorName	Optional	This extension parameter indicates the new pretranslator to be used for the call in progress.

Note: When the ***DestinationAddress*** parameter is present, the ***ResourceType*** parameter is ignored.

—end—

Connect_To_Resource (continued)

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-25 shows errors that can occur after an `Connect_To_Resource` SCP response is returned.

Table 10-25
Connect_To_Resource fatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
DestinationAddress parameter included ***Note:*** The ***DestinationAddress*** contains the address for an IP.	Response	CAIN200	Yes	Switch applies AIND.
Play Announcement and Collect Digits in response package with ***DisconnectFlag***	Response	CAIN200	Yes	Switch applies AIND.
Play Announcement and Collect Digits in conversation package with ***DisconnectFlag***	Conversation	CAIN200	Yes	Switch applies AINF.
ResourceType set to Play Announcement, and ***STRParameterBlock*** set to any value other than Play Announcement	Response	CAIN200		

—continued—

Table 10-25
Connect_To_Resource fatal application errors (continued)

Error type	**Package**	**Log generated**	**Reported to SCP?**	**Action performed**
`ResourceType` set to Play Announcement and Collect Digits in response package	Response	CAIN200	Yes	The switch sends a **`CTR_Clear`** message to the SCP with a ***`ClearCause`*** parameter value of ABORT; the call is not cleared toward the controlling leg or the passive leg.
`ResourceType` set to an unexpected value	Response	CAIN200	Yes	The switch sends a **`CTR_Clear`** message to the SCP with a ***`ClearCause`*** parameter value of ABORT; the call is not cleared toward the controlling leg or the passive leg.

—end—

Connect_To_Resource (continued)

Nonfatal application errors

Table 10-26 shows errors that can occur while the **`Connect_To_Resource`** parameters are being processed.

Table 10-26
Connect_To_Resource nonfatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Missing ***`DisconnectFlag`***	Response	CAIN100	No	The switch continues the call as though the ***`DisconnectFlag`*** were present.
Play Announcements in conversation package	Conversation	CAIN100	Yes	The switch attempts to process message as a non-conversational play announcement request.
Play Announcements in conversation package with ***`DisconnectFlag`***	Conversation	CAIN100	Yes	The switch attempts to process as a non-conversational play announcement request. The switch plays the announcements and applies AIND.
`treatment` extension parameter present	Conversation	CAIN100	No	This parameter is ignored; call processing continues.
Invalid `treatment` extension parameter received	Response	CAIN100	No	This parameter is ignored; call processing continues.
SOC for extension parameters (CAIN200) is idle	Response	CAIN102	No	Extension parameters are ignored; call processing continues.

Limitations and restrictions

Only 255 subscribers can be connected to a digital recording announcement machine (DRAM) at one time.

Associated logs

CAIN100, CAIN102, CAIN200, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG, CAINUIF

Disconnect

Use

Upon receipt of the **`Disconnect`** message, the switch stops CAIN call processing and applies treatment.

The switch handles a **`Disconnect`** message in response to a **`CTR_Clear`** message at the *Timeout* event in the same manner that it handles a **`Disconnect`** message in response to a **`Timeout`** EDP–Request. The switch performs the following actions:

1 provides treatment to the originator (AIN Disconnect treatment [AIND] or whatever treatment is specified in the `treatment` extension parameter)

2 idles the terminator

Message parameters

Table 10-27 lists the parameters that may be returned by the SCP.

Table 10-27
Disconnect message parameter

Parameter	Definition
AMAAlternateBillingNumber	This parameter identifies an alternate billing number to which the AIN service should be billed.
AMALineNumber	This parameter contains digits used to populate a CDR.
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	This parameter contains digit strings to be populated into a CDR field
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.

—continued—

Note: When the CAIN parameter TRTMTCD_COMPCODE_ZAPPED_ZERO in table CAINPARM is 'Y', the TRTMTCD and COMPCODE fields in the CDR are zapped for TBT (Take-Back & Transfer) calls terminated by the DISCONNECT message.

Table 10-27
Disconnect message parameter (continued)

Parameter	Definition
`billSequenceNumber`	This extension parameter contains 32 bits of SCP-defined billing data that is stored in the CDR.
`treatment`	When present, this extension parameter indicates the treatment to be set as a result of SCP service logic. The SCP must return a valid treatment. Refer to Volume 1, Chapter 2, “Provisioning NetworkBuilder,” for a list of valid treatments.
—end—	

Fatal application errors

None

Nonfatal application errors

Table 10-28 shows errors that can occur while the **`Disconnect`** parameters are being processed.

Table 10-28
Disconnect nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle	CAIN102	No	Extension parameters are ignored.

Associated logs

CAIN102, VAMP901

The switch generates a TRK138 log when the treatment identified in the `treatment` parameter is applied to the call.

Disconnect (end)

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG

Use

The **Disconnect_Leg** message directs the switch to release a specified leg involved in a call.

Message parameters

Table 10-29 lists the parameters that may be returned by the SCP.

Table 10-29
Disconnect_Leg parameters

Parameter	Usage	Definition
LegID	Required	When set to 0, this parameter indicates the calling party is connected to a resource. When set to 1, this parameter indicates the called party is connected to a resource.
CsID	Optional	This parameter specifies to which call segment the message applies.
ExtensionParameter	Optional	Extension parameters are not supported for this message.

Fatal application errors

None

Nonfatal application errors

None

Associated logs

VAMP901

Associated OMs

CAINMSGR

Furnish_AMA_Information

Use

The SCP sends a **`Furnish_AMA_Information`** message when Bellcore AMA Format (BAF) modules are to be included with the switch CDR.

Message parameters

Table 10-30 lists the parameters that may be included with this message.

Table 10-30
Furnish_AMA_Information parameters

Parameter	Usage	Definition
`AMABAFModules`	Optional	This parameter contains the SCP-generated billing data in the form of one or more BAF modules.
`AMASetHexABIndicator`	Optional	This parameter contains a flag set by the SCP when the ***`AMABAFModules`*** parameter contents are suspected to have errors.

Fatal application errors

A **`Furnish_AMA_Information`** message is received as the first component in the TCAP package. The UCS DMS-250 switch ignores the package. CAIN200 log is generated by the switch.

Nonfatal application errors

Call-related message does not contain an ***`AMAslpID`*** parameter. The UCS DMS-250 switch ignores the **`Furnish_AMA_Information`** message. CAIN100 log is generated by the switch.

The **`Furnish_AMA_Information`** message accompanies a message other than those supported. The UCS DMS-250 switch ignores the **`Furnish_AMA_Information`** message. CAIN100 log is generated by the switch.

Associated logs

VAMP901, CAIN100, CAIN200

Associated OMs

CAINMSGR

Billing considerations

The information provided by the SCP is appended to the switch's CDR.

Limitation and Restrictions

The **`Furnish_AMA_Information`** message is a non-call related message only supported with the **`Analyze_Route`** and **`Send_to_Resource`** call-related messages.

Merge_Call (end)

Use

When the switch receives a **`Merge_Call`** message in Call Configuration 6, it attempts to allocate a three port conference port. If conference ports are not available, the switch sends a **`Failure_Outcome`** message with a ***`FailureCause`*** of `unavailableResources`.

Note 1: There is no need for a conference port when the switch receives a **`Merge_Call`** message in any other Call Configuration.

Note 2: The **`Merge_Call`** message is only supported for Call Configurations 4 and 6.

Message parameters

NetworkBuilder does not expect to receive any parameters in this message.

Fatal application errors

None

Nonfatal application errors

None

Associated logs

VAMP901

Associated OMs

CAINMSGR

Originate_Call

Use

The **`Originate_Call`** message enables the switch to place a stable two-party call on hold and originate an associated call to another party.

Note: No parameters used for the first call segment are reused for the second call segment. For example, when no ***`CallingPartyID`*** is returned in the **`Originate_Call`** message, and the terminator is an SS7 agent, no ***`CallingPartyNumber`*** is included in the outgoing IAM.

Call routing precedence

The following list gives the order of precedence NetworkBuilder uses to determine a route for the call:

1. ***`PrimaryTrunkGroup`*** (Note 1)
2. ***`AlternateTrunkGroup`*** (Note 1)
3. ***`SecondAlternateTrunkGroup`*** (Note 1)
4. ***`Carrier`*** (Note 2)
5. ***`AlternateCarrier`*** (Note 2)
6. ***`SecondAlternateCarrier`*** (Note 2)
7. called number and STS, if present (Note 3)
8. ***`GenericAddressList`*** (`OverflowRoutingNo`) and STS, if present (Note 3)

Note 1: To route with a trunk group parameter, either a ***`CalledPartyID`*** or the appropriate ***`OutpulseNumber`*** must be returned.

Note 2: A ***`CalledPartyID`*** must be returned with a carrier parameter in order for NetworkBuilder to use a carrier parameter for routing.

Note 3: An STS returned without a ***`CalledPartyID`*** does not constitute a route for the **`Originate_Call`** message.

Originate_Call (continued)

Message parameters

Table 10-31 lists the parameters that may be returned by the SCP.

Table 10-31
Originate_Call parameters

Parameter	Definition
CallingPartyID	This parameter contains the address used for caller identification purposes.
DisplayText	This parameter contains display data to be sent across the network to the end user.
ChargeNumber	This parameter contains the billing number.
ChargePartyStationType	This parameter contains the information digits for the call.
CalledPartyID	This parameter contains the translated address of the called party.
OutpulseNumber	This parameter contains the digits to be outpulsed when the ***PrimaryTrunkGroup*** numbered outpulse flag is 0. This number is used when the outpulse number associated with ***AlternateTrunkGroup*** or ***SecondAlternateTrunkGroup*** is not available.
PrimaryTrunkGroup	This parameter contains the SCP-determined primary route index for table TERMRTE or TANDMRTE.
AlternateTrunkGroup	This parameter contains the SCP-determined alternate route index for table TERMRTE or TANDMRTE.
SecondAlternateTrunkGroup	This parameter contains the SCP-determined second alternate route index for table TERMRTE or TANDMRTE.
Carrier	This parameter contains the carrier selection information and the carrier identification code (CIC) to route the call.

—continued—

Table 10-31
Originate_Call parameters (continued)

Parameter	Definition
AlternateCarrier	This parameter contains the alternate carrier selection information (CSI) and the carrier identification code (CIC) to route the call.
SecondAlternateCarrier	This parameter contains the second alternate carrier selection information and the carrier identification code (CIC) to route the call.
GenericAddressList	
AlternateOutpulseNo	This parameter contains the digits to be outpulsed when the ***AlternateTrunkGroup*** number to outpulse field is 0.
SecondAlternate OutpulseNo	This parameter contains the digits to be outpulsed when the ***SecondAlternateTrunkGroup*** number to outpulse field is 0.
OverflowRoutingNo	This parameter contains an overflow routing number.
DialedNoInwardService	This parameter contains a DNIS value specific to the Dialed Number Inward Service specified.
PortedDialedNo	This parameter contains the original called party address of a successful LNP query.
ForwardCallIndicator	This parameter indicates that an LNP check has been performed at the SCP.
AMAAlternateBillingNumber	This parameter identifies an alternate billing number to which the AIN service should be billed.
AMALineNumber	This parameter contains digits used to populate one or more of the following CDR fields: ANISP, CLGPTYNO, or PRESIND.
AMAslpID	This parameter contains a digit string to be populated into the SLPID CDR field.

—continued—

Originate_Call (continued)

Table 10-31
Originate_Call parameters (continued)

Parameter	Definition
AMADigitsDialedWC	This parameter contains digit strings to be populated into one or more of the following CDR fields: PINDIGS, ACCTCD, BILLNUM, CIC, ORIGPVN, PRJCODE, or TERMPVN.
ExtensionParameter	***ExtensionParameter***s require the CAIN0200 SOC option.
servTranslationScheme	This extension parameter contains a serving translation scheme.
billSequenceNumber	This extension parameter contains 32 bits of SCP-defined billing data that is stored in the CDR.
netinfo	This extension parameter contains business group information to be sent in the outgoing ISUP IAM.
primaryTrunkGroupSTS	This extension parameter contains the STS value associated with the returned ***PrimaryTrunkGroup*** parameter.
alternateTrunkGroupSTS	This extension parameter contains the STS value associated with the returned ***AlternateTrunkGroup*** parameter.
secondAlternateTrunk GroupSTS	This extension parameter contains the STS value associated with the returned ***SecondAlternateTrunkGroup*** parameter.
overflowRoutingNoSTS	This extension parameter contains the STS value associated with the ***GenericAddressList*** parameter's returned OverflowRoutingNo.
accountCode	This extension parameter contains the account code collected from in-switch translations or by the *O_Feature_Requested* trigger.

—continued—

Table 10-31
Originate_Call parameters (continued)

Parameter	Definition
`pinDigits`	This extension parameter contains the collected PIN code, when available, or a PIN collected at *O_Feature_Requested*. It also contains digits collected during conversational digit collection.
`billingNumber`	This extension parameter contains a non-standard charge number (for example, CARD, AUTH, ACCT, PIN, or N00).
—end—	

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-32 shows fatal application errors that can occur after an **`Originate_Call`** SCP response is returned.

Table 10-32
Originate_Call fatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
message contains neither ***`CalledPartyID`*** nor a trunk group parameter (***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, or ***`SecondAlternateTrunkGroup`***) with an ***`OutpulseNumber`***	CAIN200	Yes	**`Failure_Outcome`** sent

Nonfatal application errors

Table 10-33 shows nonfatal errors that can occur while the **`Originate_Call`** parameters are being processed.

Originate_Call (continued)

Table 10-33
Originate_Call nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is idle.	CAIN102	No	Extension parameters are ignored.

Associated logs

CAIN102, CAIN200, VAMP901

Associated OMs

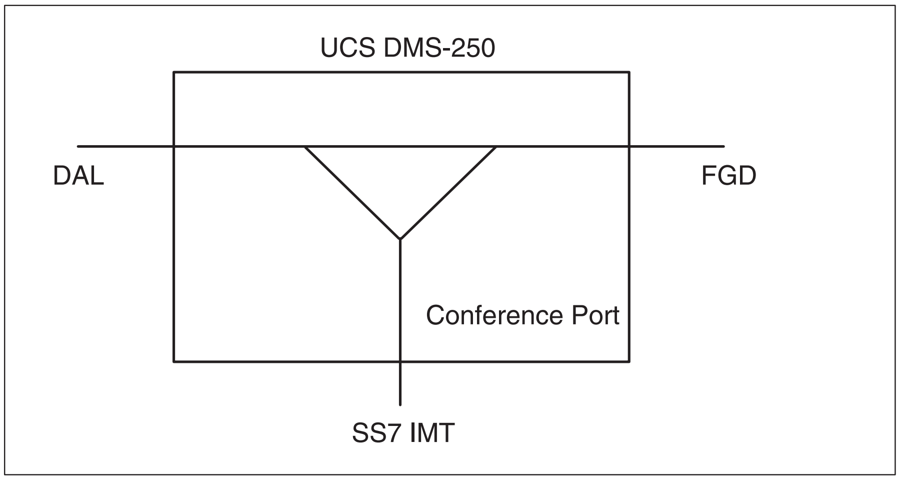
CAINMSGR

Billing considerations

The `Originate_Call` message causes two events of interest in regards to billing. First, the call configuration transitions to CC4. Secondly, a second recording unit (RU) is obtained for the second call segment.

The following call scenario summarizes how the CDR will be populated due to the `Originate_Call` message. A dedicated access line (DAL) trunk makes a call to a feature group D trunk (FGD). EAN660 performs the switch hook flash, which results in an message and creates a call to an SS7 inter-machine trunk (IMT), as shown in Figure 10-5.

Figure 10-5
Originate_Call call scenario



There are now two CDRs involved in the call. CDR1 is generated when either the DAL or the FGD releases. CDR2 is generated when the IMT releases. The means of the release is irrelevant, either a user initiates the release, or the SCP may initiate the release. The call's call configuration at the time the user releases is irrelevant to the timing of CDR generation. The CDR is generated upon the release. Table 10-34 summarizes CDR population for this call scenario.

Table 10-34
CDR field summary

CDR field(s)	CDR1	CDR2
sequence number	see Note.	see Note.
originating trunk information	the DAL trunk	the FGD trunk
terminating trunk information	the FGD trunk	the SS7 IMT trunk
routing information	information related to routing from the DAL to the FGD	information related to routing from the FGD to the SS7 IMT
CAIN specific fields	all CAIN information for this call is stored in this CDR.	always empty

Note: The sequence number for CDR1 and CDR2 is the same.

Limitations and Restrictions

If the SCP does not send a ***`CalledPartyID`*** parameter, the switch does not populate the CALLEDNO field of the CDR.

If the switch does not receive the STS in the **`Originate_Call`** message, the switch uses the DEFAULT_STS in table OFCVAR to route the call.

Send_Notification (end)

Use

The `Send_Notification` instructs the switch to send a `Termination_Notification` when the call is released.

Message parameters

Table 10-35 lists the parameter that is returned by the SCP.

Table 10-35
Close parameters

Parameter	Usage	Definition
EchoData	Required	The SCP sends this parameter to correlate the `Termination_Notification` with the appropriate `Send_Notification` message.

Associated logs

VAMP901

Associated OMs

CAINAGOM, CAINMSGR, CAINTRIG

Send_To_Resource

Use

The **`Send_To_Resource`** message instructs the switch to perform one of the following actions:

- play an announcement and disconnect
- play one or more announcement and collect digits (Refer to Volume 4, “Conversational processing,” for more information.)
- route to an IP (Refer to Volume 4, “Conversational processing,” for more information.)

Note 1: Digit collection and IP routing are available during conversational messaging.

Note 2: Although the *Termination_Attempt* trigger may be encountered during routing to an IP, based on processing of other triggers, routing to an IP following a *Termination_Attempt* trigger is not allowed due to restrictions on originating agent types supported for IP connections.

Note 3: An ***`AnswerIndicator`*** parameter received in a **`Send_To_Resource`** message may initiate the sending of an answer message (ANM) back through the network. This message potentially affects billing on the originating switch. An early ANM message will cause the tracking of the call duration for the call to be started on the originating switch even before the call termination attempt is not successful. Since call detail records (CDRs) are not generated when a call is tandemed through the SS7 Intra-IMT agency, billing will be affected on the originating switch.

Play announcement

The play resource capability allows the the SCP to instruct the switch to play an announcement or tone over the voice channel to the caller.

A play announcement **`Send_To_Resource`** message is transmitted in a Response package with an Invoke_Last component.

Table 10-36 lists the parameters the **`Send_To_Resource`** message may contain.

Send_To_Resource (continued)

Table 10-36
Play announcement parameters

Parameter	Usage	Definition
ResourceType (Note 1)	Required	This parameter contains `Play Announcement`.
StrParameterBlock	Required	This parameter contains an `AnnouncementBlock` encoded as an `UninterAnnounceBlock`. The `UninterAnnounceBlock` contains a Play Announcement tag and up to 3 uninterruptible announcement elements. The announcement elements contain a resource identifier (index into table CAINRSRC). ***Note 1:*** Only one announcement block is expected. ***Note 2:*** Any dialed digits are ignored.
DisconnectFlag	Required	The presence of this parameter indicates the switch disconnects the call after announcement or tone is played.
AnswerIndicator	Optional	The presence of this parameter instructs the switch to provide answer supervision to the originating agent while the caller is connected to the resource. The switch sends answer indication to the caller in response to the `Play Announcement` request if answer indication has not already been sent. ***Note 1:*** ***AnswerIndicator*** is only used for SS7 and PRI originators. ***Note 2:*** ***AnswerIndicator*** does not affect billing at the querying switch.
AMAAlternateBillingNumber	Optional	This parameter identifies an alternate billing number to which the AIN service should be billed.

Note 1: When the ***DestinationAddress*** parameter is present, the ***ResourceType*** parameter is ignored.
Note 2: When the ***DestinationAddress*** parameter is present in a **Send_To_Resource** message received in response to a **Termination_Attempt** TDP-Request, the call is handled as if an **Authorize_Termination** message was received.

—continued—

Table 10-36
Play announcement parameters (continued)

Parameter	Usage	Definition
AMALineNumber	Optional	This parameter contains digits used to populate one or more of the following CDR fields: ANISP, CLGPTYNO, or PRESIND.
AMAslpID	Optional	This parameter contains a digit string to be populated into the SLPID CDR field.
AMADigitsDialedWC	Optional	This parameter contains digit strings to be populated into the CDR.
DestinationAddress	Optional	This parameter contains the address of an IP.
AMAMeasure	Optional	This parameter indicates that a duration time measurement is required.
PrimaryBillingIndicator	Optional	This parameter provides customized billing information for the UCS DMS-250 CDR.
ExtensionParameter	Optional	***ExtensionParameter***s require the CAIN0200 SOC option.
treatment	Optional	When present, this extension parameter indicates the treatment to be set. ***Note:*** After the resource identified in the **Send_To_Resource** message, any resources identified by the treatment are played.
strConnectionType	Optional	This extension parameter indicates what type of connection protocol is to be used to establish communication between the SCP, switch, and an IP resource.
pretranslatorName	Optional	This extension parameter indicates the new pretranslator to be used for the call in progress, if address collection is required later in the call, or reorigination occurs.

Note 1: When the ***DestinationAddress*** parameter is present, the ***ResourceType*** parameter is ignored.
Note 2: When the ***DestinationAddress*** parameter is present in a **Send_To_Resource** message received in response to a **Termination_Attempt** TDP-Request, the call is handled as if an **Authorize_Termination** message was received.

—end—

Send_To_Resource (continued)

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 10-37 shows errors that can occur after an **`Send_To_Resource`** SCP response is returned.

Table 10-37
Send_To_Resource fatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
`DestinationAddress` parameter included ***Note:*** The ***`DestinationAddress`*** contains the address for an IP.	Response	CAIN200	Yes	Switch applies AIND.
Play Announcement and Collect Digits in response package with ***`DisconnectFlag`***	Response	CAIN200	Yes	Switch applies AIND.
Play Announcement and Collect Digits in conversation package with ***`DisconnectFlag`***	Conversation	CAIN200	Yes	Switch applies AINF.

—continued—

Table 10-37
Send_To_Resource fatal application errors (continued)

Error type	Package	Log generated	Reported to SCP?	Action performed
ResourceType set to Play Announcement and Collect Digits in response package	Response	CAIN200	Yes	The switch sends a **Resource_Clear** message to the SCP with a ***ClearCause*** parameter value of ABORT; the call is not cleared toward the controlling leg or the passive leg.
ResourceType set to an unexpected value	Response	CAIN200	Yes	The switch sends a **Resource_Clear** message to the SCP with a ***ClearCause*** parameter value of ABORT; the call is not cleared toward the controlling leg or the passive leg.
—end—				

Send_To_Resource (continued)

Nonfatal application errors

Table 10-38 shows errors that can occur while the **`Send_To_Resource`** parameters are being processed.

Table 10-38
Send_To_Resource nonfatal application errors

Error type	Package	Log generated	Reported to SCP?	Action performed
Missing ***`DisconnectFlag`***	Response	CAIN100	No	The switch continues the call as though the ***`DisconnectFlag`*** were present.
Play Announcements in conversation package	Conversation	CAIN100	Yes	Switch attempts to process message as a non-conversational play announcement request.
Play Announcements in conversation package with ***`DisconnectFlag`***	Conversation	CAIN100	Yes	Switch attempts to process as a non-conversational play announcement request. The switch plays the announcements and applies AIND.
`treatment` extension parameter present	Conversation	CAIN100	No	This parameter is ignored; call processing continues.
Invalid `treatment` extension parameter received	Response	CAIN100	No	This parameter is ignored; call processing continues.
SOC for extension parameters (CAIN200) is idle	Response	CAIN102	No	Extension parameters are ignored; call processing continues.

Limitations and restrictions

Only 255 subscribers can be connected to a digital recording announcement machine (DRAM) at one time.

Associated logs

CAIN100, CAIN102, CAIN200, VAMP901

Associated OMs

CAINMSGR, CAINAGOM, CAINTRIG, CAINUIF

Incoming IN/1 messages

When the switch receives a message from the SCP, the CAIN framework processes the response. Response processing consists of interpreting, validating, and performing instructions provided in the SCP TCAP message. These instructions direct the switch on how to proceed with the call. IN/1 supports the call-related incoming messages listed in Table 11-1.

Note: SCP response messages, parameters, and extension parameters are handled differently for LNP calls; refer to *UCS DMS-250 Local Number Portability Application Guide* for more information. SCP response parameters are also handled differently for AXXESS agents; refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Table 11-1
Call-related incoming IN/1 messages

Response	Description
`Play_Announcement`	The SCP sends this message when it is unable to respond with routing instructions due to an irregular user action.
`Connect`	The SCP sends this message to provide the switch with call routing directions.

IN/1 supports the non-call related incoming messages listed in Table 11-2.

Table 11-2
Non-call-related incoming IN/1 messages

Response	Description
`ACG`	The SCP sends this message to specify an ACG control to be added, updated, or deleted from the ACG control list.
`Termination`	The SCP sends this message to request that the switch send a `Termination_Information` message when the call is released.

IN/1 supports the incoming error messages listed in Table 11-3.

Table 11-3
Incoming IN/1 error messages

Response	Description
`Reject`	The SCP sends this message when a protocol error occurs in a query message.
`Return_Error`	The SCP sends this message when it cannot provide routing instruction because of improper or invalid data in a query message.

Fatal application errors

For more information on fatal application errors, refer to Chapter 1, “TCAP messaging.”

Nonfatal application errors

For more information on nonfatal application errors, refer to Chapter 1, “TCAP messaging.”

Associated logs

VAMP901

Associated OMs

CAINMSGR, TFREE533

Use

The SCP sends an **ACG** message to specify an ACG control to be added, updated, or deleted from the ACG control list.

Message parameters

The **ACG** message is sent with a component type of Invoke (Last).

Table 11-4 lists the parameters sent by the SCP.

Table 11-4
ACG parameters

Parameter	Usage	Definition
Digits (called party number)	Required	The ***Digits*** (called party number) parameter contains the 6 or 10-digit number to which ACG should be applied.
AutomaticCallGapIndicators	Required	The ***AutomaticCallGapIndicators*** parameter contains the gap interval, the control duration, and the cause of the ACG control.

Associated logs

VAMP901

Associated OMs

VAMPACG

Connect (end)

Use

The SCP sends a `Connect` message to provide the switch with call routing directions.

Message parameters

The message is sent in a response package, with a component type of Invoke (Not Last).

Table 11-5 lists the parameters sent by the SCP.

Table 11-5
Connect parameters

Parameter	Usage	Definition
`Digits` (carrier)	Required	The ***`Digits`*** (carrier) parameter identifies the carrier for the call.
`BillingIndicators`	Required	The ***`BillingIndicators`*** parameter contains various billing data.
`Digits` (routing number)	Required	The ***`Digits`*** (routing number) parameter contains the routing number for the call.

Associated logs

CAIN101, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGS

Use

The SCP sends a **`Play_Announcement`** message when it is unable to respond with routing instructions due to an irregular user action.

Message parameters

The **`Play_Announcement`** message is sent with a component type of Invoke (Last).

Table 11-6 lists the parameter sent by the SCP.

Table 11-6
Play_Announcement parameters

Parameter	Usage	Definition
`StandardAnnouncement`	Required	The ***`StandardAnnouncement`*** parameter contains the code corresponding to the announcement that should be played.

Associated logs

CAIN101, CAIN200, CAIN201, VAMP901

Associated OMs

TFREE533

Reject (end)

Use

The SCP sends a `Reject` message when a protocol error occurs in a query message.

Message parameters

The `Reject` message is sent in a response package, with a component type of Reject.

Table 11-7 lists the parameter sent by the SCP.

Table 11-7
Reject parameters

Parameter	Usage	Definition
`ProblemCode`	Required	The *`ProblemCode`* parameter identifies the reason for sending the `Reject` message.

Associated logs

CAIN200, CAIN201

Associated OMs

None

Use

The SCP sends a **`Return_Error`** message when it cannot provide routing instruction because of improper or invalid data in a query message.

Message parameters

The **`Return_Error`** message is sent in a response package, with a component type of Return Error.

Table 11-8 lists the parameters sent by the SCP.

Table 11-8
Return_Error parameters

Parameter	Usage	Definition
`ErrorCode`	Required	The ***`ErrorCode`*** parameter indicates if an error condition has resulted from an unexpected data value, unexpected component sequence, unavailable network resource, or unavailable data.
`ProblemData`	Required	The ***`ProblemData`*** parameter contains the Identifier, Length of Contents, and Contents of an erroneous data element.

Associated logs

None

Associated OMs

None

Termination (end)

Use

The SCP sends a `Termination` message to request that the switch send a `Termination_Information` message when the call is released.

Message parameters

The `Termination` message is sent with a component type of Invoke (Last).

Table 11-9 lists the parameter sent by the SCP.

Table 11-9
Termination parameters

Parameter	Usage	Definition
EchoData	Required	The ***EchoData*** parameter correlates the termination information with the received request for information.

Associated logs

CAIN101, CAIN200, CAIN201, VAMP901

Associated OMs

CAINMSGS

Incoming CAIN message parameters

ATTENTION
Extension parameters require the CAIN0200 SOC option. Refer to Volume 5, Chapter 5, “NetworkBuilder SOC functionality,” for more information.

Table 12-1 lists the parameters that may be sent from the SCP within a TCAP response message.

Table 12-1
Incoming CAIN message parameters

Parameter	Usage
ACGGlobalOverride	Required
AlternateCarrier	Optional
AlternateTrunkGroup	Optional
AMAAlternateBillingNumber (Note)	Optional
AMABAFModules	Optional
AMADigitsDialedWC (Note)	Optional
AMALineNumber (Note)	Optional
AMAMeasure	Optional
AMASetHexABIndicator	Optional
AMAslpID	Optional
AnswerIndicator	Optional

Note: This parameter uses the AIN Digits format.

—continued—

Table 12-1
Incoming CAIN message parameters (continued)

Parameter	Usage
`CalledPartyID` (Note)	Optional
`CallingPartyID` (Note)	Optional
`Carrier`	Optional
`ChargeNumber` (Note)	Optional
`ChargePartyStationType`	Optional
`ControlCauseIndicator`	Required
`DestinationAddress` (Note)	Optional
`DisconnectFlag`	Optional
`DisplayText`	Optional
`EchoData`	Required
`EDPNotification`	Optional
`EDPRequest`	Optional
`ExtensionParameter`	Optional
`accountCode`	Optional
`alternateTrunkGroupSTS`	Optional
`amaDigits`	Optional
`billingNumber`	Optional
`billSequenceNumber`	Optional
`cainGroup`	Optional
`callBranding`	Optional
`callCtrl`	Optional
`callType`	Optional
`classOfSvc`	Optional
`connectToSCU`	Optional
`edpBuffer`	Optional
`netinfo`	Optional

Note: This parameter uses the AIN Digits format.

—continued—

Table 12-1
Incoming CAIN message parameters (continued)

Parameter	Usage
`networkBusyActions`	Optional
`oCalledPartyBusyActions`	Optional
`oNoAnswerActions`	Optional
`overflowRoutingNoSTS`	Optional
`pinDigits`	Optional
`pretranslatorName`	Optional
`primaryTrunkGroupSTS`	Optional
`reorigAllowed`	Optional
`satRestriction`	Optional
`secondAlternateTrunkGroupSTS`	Optional
`servTranslationScheme`	Optional
`strConnectionType`	Optional
`shfelegs`	Optional
`treatment`	Optional
`univIdx`	Optional
`ForwardCallIndicator`	Optional
`GapDuration`	Required
`GapInterval`	Required
`GenericAddressList`	Optional
`AlternateOutpulseNo`	Optional
`DialedNoInwardService`	Optional
`OverflowRoutingNo`	Optional
`PortedDialedNo`	Optional
`SecondAlternateOutpulseNo`	Optional
`GlobalTitleAddress`	Required
`LegID`	Optional

Note: This parameter uses the AIN Digits format.

—continued—

Table 12-1
Incoming CAIN message parameters (continued)

Parameter	Usage
ONoAnswerTimer	Optional
OutpulseNumber (Note)	Optional
OverflowBillingIndicator	Optional
PrimaryBillingIndicator	Optional
PrimaryTrunkGroup	Optional
ResourceType	Required for **Send_To_Resource** and **Connect_To_Resource**; optional for **Call_Info_To_Resource**
SecondAlternateCarrier	Optional
SecondAlternateTrunkGroup	Optional
StrParameterBlock	Required
TimeoutTimer	Optional
TranslationType	Required

Note: This parameter uses the AIN Digits format.

—end—

Fatal application errors

Fatal application errors occur when CAIN call processing is unable to continue due to an unexpected error. Table 12-2 shows errors beyond those listed in Table 1-3 that can occur after an SCP response is returned.

Table 12-2
Incoming message parameter fatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
Charge number unavailable (Note 2)	CAIN200	Yes	ERRACT in trigger table (Note 1)

Note 1: Tables OFFHKIMM, SPECFEAT, CUSTDP, SPECDIG, and TERMATT provision the error action; OFFCCODE defaults to ROUTE, but does not require provisioning; all other triggers perform an error action of TREAT.
Note 2: A charge number wasn't included in the query, wasn't returned by the SCP, and/or the message was truncated.

Nonfatal application errors

For more information on nonfatal application errors, refer to Volume 4, "Conversational processing," Chapter 1, "TCAP messaging," Table 1-4, or each specific parameter within this chapter.

Associated logs

CAIN100, CAIN101, CAIN102, CAIN200, CAIN201

Associated OMs

CAINMSGR, CAINMGR2, CAINOM, CAINAGOM, CAINTRIG, CAINUIF, VTCAPERR, VTCAPSNT, VPTCAPRCV

ACGGlobalOverride (continued)

Parameter definition

This parameter reinitializes the ACG control list according to the option provided.

Message types

The following message supports the ***`ACGGlobalOverride`*** parameter:

- **`ACG_Global_Ctrl_Restore`**

Usage

Required

Range of values

Table 12-3 shows the valid options for the ***ACGGlobalOverride*** parameter.

Table 12-3
ACGGlobalOverride parameter values

Enumerated Type	Enumerated Value	Definition
allitems	0	This parameter option removes all controls from both the SCP and SOCC control lists.
scpOverloadItems	1	This parameter option removes all controls from the SCP control list.
smsInitCntrlExceptZero Gap	2	This parameter option removes all controls from the SOCC control list except zero-gap controls.
smsInitCntrl	3	This parameter option removes all controls form the SOCC control list.
not supported	4	
not supported	5	
craftInitCntrlExceptZero Gap	6	This parameter option removes all craft (ACGCNTRL CI) initiated controls except zero-gap controls.
craftInitCntrl	7	This parameter option removes all craft (ACGCNTRL CI) initiated controls from the SOCC control list.

CDR population

None

Application errors

None

Associated logs

None

ACGGlobalOverride (end)

Restrictions

None

Parameter definition

This parameter contains alternate carrier selection information and carrier identification code (CIC) to route the call.

Message types

The following messages support the ***`AlternateCarrier`*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Note: When the SCP sends the ***`AlternateCarrier`*** parameter in an **`Analyze_Route`** message, the switch will retain the parameter for future use. Refer to Chapter 10 for further information on the **`Analyze_Route`** message.

Usage

Optional

Range of values

Carrier Selection Field

- NO_INDICATION
- PRESUBSCRIBED_AND_NOT_INPUT
- PRESUBSCRIBED_AND_INPUT
- PRESUBSCRIBED_AND_NO_INDICATION
- NOT_PRESUBSCRIBED_AND_INPUT

Carrier ID field

- 0000–9999

CDR population

If the CIC digits of the ***`AMADigitsDialedWC`*** parameter are not present, CIC and CARRSEL CDR fields are populated from this parameter when it is used to route the call.

AlternateCarrier (end)

Application errors

Table 12-4 shows errors that can occur with regards to the `AlternateCarrier` parameter.

Table 12-4
AlternateCarrier nonfatal application errors

Error type	Log generated	Error action performed
CIC not datafilled in table CICRTE	CAIN100	The parameter is ignored.

Associated logs

CAIN100

Restrictions

None

Parameter definition

This parameter contains the alternate route index for table TERMRTE or TANDMRTE and consists of the terminating switch identifier and a trunk group number for the terminating agent.

Message types

The following messages support the ***AlternateTrunkGroup*** parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

SWID: 0–999
trunk group number: 0–8191 (field ADNUM from table CLLI)
number to outpulse: 0–1

Outpulsing

When a call is routed using the alternate route index, the following outpulse precedence rules apply:

1. Outpulse the ***GenericAddressList***'s AlternateOutpulseNo, when present and number to outpulse is set to 0.
2. Outpulse the ***OutpulseNumber***, when present, and number to outpulse is set to 0.
3. Outpulse the ***CalledPartyID***, when present or the current translated address.
4. Normal in-switch outpulsing rules apply.

CDR population

None

Application errors

Table 12-5 shows errors that can occur with regards to the ***AlternateTrunkGroup*** parameter.

AlternateTrunkGroup (end)

Table 12-5
AlternateTrunkGroup parameter processing errors

Error type	Log generated	Error action performed
Unknown TERMRTE index (trk number not found in TERMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action
Unknown TANDMRTE index (switchid not found in TANDMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action

Note: The actions performed in response to these errors assume that there are no other routing parameters in the message (the action may be different if there are other routing parameters in the message).

Associated logs

CAIN300

Restrictions

None

Parameter definition

The ***AMAAlternateBillingNumber*** parameter identifies an alternate billing number to which the AIN service should be billed. It is primarily used to bill the end-to-end call to third-party North American Numbering Plan (NANP) numbers.

Message types

The following messages support the ***AMAAlternateBillingNumber*** parameter:

- **Analyze_Route**
- **Authorize_Termination**
- **Collect_Information**
- **Connect_To_Resource**
- **Continue**
- **Disconnect**
- **Originate_Call**
- **Send_To_Resource**

Usage

Optional

Range of values

1 to 11 digits in AINDigits format

Note 1: The switch assumes that any ***AMAAlternateBillingNumber*** value received from the SCP is valid.

Supported NOAs

The switch ignores any values for the NOA and the Numbering Plan fields.

CDR population

The switch uses the contents of the ***AMAAlternateBillingNumber*** parameter to populate the ALTBILL CDR field.

Application errors

None

Associated logs

None

AMAAlternateBillingNumber (end)

Restrictions

None

Parameter definition

The ***`AMABAFModules`*** parameter contains SCP-generated billing data in the form of one or more BAF modules.

Message types

The **`Furnish_AMA_Information`** message supports the ***`AMABAFModules`*** parameter.

Usage

Optional

Range of values

A string of hexadecimal digits of no less than 2 and up to 128 bytes.

CDR population

The switch uses the contents of the ***`AMABAFModules`*** parameter to populate the AMABAFMD field. The AMASIZE field contains the size of the AMABAFMD field.

Application errors

Table 12-6 describes the non-fatal application errors for the ***`AMABAFModules`*** parameter.

AMABAFModules (continued)

Table 12-6
AMABAFModules nonfatal application errors

Error type	Log generated	Error action performed
TCAP package does not contain a ***AMAslpID*** parameter.	CAIN100	The UCS DMS-250 switch ignores the data received in the **Furnish_AMA_Information** component. The switch pegs the LAMAFAIL register.
No Furnish AMA extension blocks available.	CAIN100	The UCS DMS-250 switch ignores the data in the **Furnish_AMA_Information** component. The switch pegs the LAMAREXH register.
Furnish_AMA_Information component received with a call-related message other than a **Analyze_Route** or **Send_to_Resource**.	CAIN100	The UCS DMS-250 switch ignores the data received in the **Furnish_AMA_Information** component. The switch pegs the LAMAFAIL register.
TCAP package contains a **Furnish_AMA_Information** component, but the package does not contain a supported call-related message.	None	The UCS DMS-250 switch ignores the data received in the **Furnish_AMA_Information** component.
The size of the ***AMABAFModules*** parameter is larger than the Bellcore-specified maximum length of 128 bytes or smaller than the minimum of 2 bytes.	None	The UCS DMS-250 switch ignores the data received in the **Furnish_AMA_Information** component. The switch pegs the LAMAFAIL register.

Table 12-7 describes the fatal application errors.

Table 12-7
AMABAFModules fatal application errors

Error type	Log generated	Error action performed
A **`Furnish_AMA_Information`** component is received as the first component in the TCAP package or the first component is not call-related.	CAIN200	The package is ignored. The UCS DMS-250 switch pegs the LAMAFAIL register.

Associated OMs

CAINMSGR, CAINOM

Restrictions

Although the CDR field, AMABAFMD, can hold up to 128 bytes of BAF table information, the CDR273 log that displays this information displays only up to 16 bytes of information. However, all information stored in the AMABAFMD field is passed on to the SDM.

AMADigitsDialedWC (continued)

Parameter definition

This parameter contains digit strings to be populated into one of the following CDR fields: PINDIGS, PRJCODE, ACCTCD, BILLNUM, CIC, ORIGPVN, or TERMPVN. Up to six ***AMADigitsDialedWC*** parameters can be sent (in sequence) within one response message.

Message types

The following messages support the ***AMADigitsDialedWC*** parameter:

- **Analyze_Route**
- **Continue**
- **Disconnect**
- **Send_To_Resource**
- **Connect_To_Resource**
- **Collect_Information**
- **Authorize_Termination**
- **Originate_Call**

Usage

Optional

Range of values

3 to 27 digits

Note: The first 3 digits identify the CDR field and the remaining digits are populated into the CDR.

Supported NOAs

This parameter does not use NOAs.

CDR population

The handling of the ***AMADigitsDialedWC*** parameter is controlled by the CAIN_PROTOCOL_VERSION parameter in table CAINPARM. Tables 12-8 through 12-10 provides information about the fields that are populated in the CDR when the***AMADigitsDialedWC*** parameter is received.

AMADigitsDialedWC (continued)

Table 12-8 shows the possible fields that are populated in the CDR if the CAIN_PROTOCOL_VERSION is set to V2 or lower when the ***AMADigitsDialedWC*** parameter is received.

Table 12-8
AMADigitsDialedWC CDR population when CAIN_PROTOCOL_VERSION is V2 or lower

CDR field	Number of digits in CDR	Digits code	Population
PINDIGS	4	001	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber’s PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.
ACCTCD	12	002	ACCOUNT CODE. This field is populated with the account code digits collected for the call.
BILLNUM	23	003	BILLING NUMBER. This field is populated with one of the following and identifies the billing number for a call: • Travel card number • Authcode • N00 called number (NXX-NXX-XXXX). • SCP-identified number
CIC	4	004	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—continued—

AMADigitsDialedWC (continued)

Table 12-8
AMADigitsDialedWC CDR population when CAIN_PROTOCOL_VERSION is V2 or lower (continued)

CDR field	Number of digits in CDR	Digits code	Population
ORIGPVN	15	005	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	006	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—end—

Table 12-9 shows the possible fields that are populated in the CDR if the CAIN_PROTOCOL_VERSION is set to V3 when the ***AMADigitsDialedWC*** parameter is received.

Table 12-9
AMADigitsDialedWC CDR population
when CAIN_PROTOCOL_VERSION is V3 or higher

CDR field	Number of digits in CDR	Digits code	Population
BILLNUM	23	001	BILLING NUMBER. This field is populated with one of the following and identifies the billing number for a call: • Travel card number • Authcode • N00 called number (NXX-NXX-XXXX). • SCP-identified number
ACCTCD	12	002	ACCOUNT CODE. This field is populated with the account code digits collected for the call.
Note 1		003	None
Note 1		004	None
Note 1		005	None
Note 1		006	None
Note 1		011	None
Note 1		012	None
PINDIGS	4	301	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber’s PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.

Note 1: CAIN call processing allows these digit codes but does not act upon or populate any CDR field with them.
Note 2: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 3: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—continued—

AMADigitsDialedWC (continued)

Table 12-9
AMADigitsDialedWC CDR population
when CAIN_PROTOCOL_VERSION is V3 or higher (continued)

CDR field	Number of digits in CDR	Digits code	Population
CIC	4	302	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.
ORIGPVN	15	303	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	304	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.
Note 1		999	None

Note 1: CAIN call processing allows these digit codes but does not act upon or populate any CDR field with them.
Note 2: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 3: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—end—

Table 12-10 shows the possible fields that are populated in the CDR when the ***AMADigitsDialedWC*** parameter is received regardless of the CAIN_PROTOCOL_VERSION setting.

Table 12-10
AMADigitsDialedWC CDR population regardless of the CAIN_PROTOCOL_VERSION

CDR field	Number of digits in CDR	Digits code	Population
PINDIGS	4	301	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber’s PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.
CIC	4	302	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.
ORIGPVN	15	303	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	304	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the ***AMADigitsDialedWC*** parameter. Non-CAIN calls do not populate this field.
PRJCODE	8	305	This field contains the project code which maybe used to refine the account code.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

Application errors

None

Associated logs

None

AMADigitsDialedWC (end)

Restrictions

When there are multiple ***`AMADigitsDialedWC`*** parameters having the same CDR field associated with them, the last one processed takes precedence in populating the CDR field.

Parameter definition

The ***AMALineNumber*** parameter contains digits used to populate one or more of the following CDR fields: ANISP, CLGPTYNO, and PRESIND.

Message types

The following messages support the ***AMALineNumber*** parameter:

- **Analyze_Route**
- **Authorize_Termination**
- **Collect_Information**
- **Connect_To_Resource**
- **Continue**
- **Disconnect**
- **Originate_Call**
- **Send_To_Resource**

Usage

Optional

Range of values

3 to 15 digits in AINDigits format

Note 1: The first 3 digits identify the CDR fields to be populated, and the remaining digits are populated into the corresponding CDRs.

Note 2: The switch assumes that any ***AMALineNumber*** value received from the SCP is valid.

Supported NOAs

The switch ignores any values for the NOA and the Numbering Plan fields.

AMALineNumber (continued)

CDR population

Table 12-11 shows the possible fields populated in the CDR, based on the information provided by the ***AMALineNumber*** parameter.

Table 12-11
AMALineNumber CDR population

Three digit code	CDR field	Number of digits in CDR	Population
005	CLGPTYNO	15	CLGPTYNO. The digits received for this field are prefixed with zeros, if necessary. The prefixed zeros cannot have significance for the numbers recorded. If the leading zeros do have significance, the number should be sent in the ***AMADigitsDialedWC*** parameter.
	PRESIND		PRESIND. This field is mapped directly from the presentation restriction indicator field of the ***AMALineNumber*** parameter. ***Note:*** When both a Calling Party Number and an ***AMALineNumber*** are available, the switch gives precedence to the information contained in the ***AMALineNumber*** for CDR population.
006	ANISP	12	ANISP. When both an ***AMALineNumber*** of type ANI and a ***ChargeNumber*** of type NATL, ANI, or I2ANI, the ***AMALineNumber*** takes precedence in populating the ANISP CDR field.
Note 1: Any three-digit code other than 005 or 006 is ignored, and therefore the following digits do not populate any CDR field. ***Note 2:*** Any digits exceeding the allowed lengths of the corresponding CDR fields are ignored.			

Application errors

None

Associated logs

None

Restrictions

When the switch receives multiple ***`AMALineNumber`*** parameters having the same CDR fields associated with them, the last parameter processed takes precedence in populating the CDR fields.

AMAMeasure (end)

Parameter definition

This parameter indicates that a duration time measurement is required. The duration time measures how long the user is connected to a resource located at the switch or at an IP.

Note: Prior to UCS08, information the ***`AMAMeasure`*** parameter currently supplies was supplied by the ***`AnswerIndicator`*** parameter.

Message types

The following messages support the ***`AMAMeasure`*** parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

connectTimeRecordedDestinationSSP
connectTimeRecordedDesinationSCP
connectTimeNotRecorded

Note: If the value is connectTimeRecordedDesinationSCP or connectTimeNotRecorded no timing is begun.

CDR population

CDR fields ANSTYPE and CALLDUR are affected by the ***`AMAMeasure`*** parameter when the switch interacts with an IP and when the switch simulates interaction with an IP. Refer to Volume 4, “Conversational processing,” for more information.

Application errors

None

Associated logs

None

Restrictions

If more than one STR- or CTR-Connection is sent during a single call, subsequent ***`AMAMeasure`*** parameters will not reset or stop timing, but will start timing if it has not already begun.

Parameter definition

The ***AMASetHexABIndicator*** parameter contains a flag set by the SCP when the ***AMABAFModules*** parameter contents are suspected to have errors.

Message types

The **Furnish_AMA_Information** message supports the ***AMASetHexABIndicator*** parameter.

Usage

Optional

Range of values

TRUE, FALSE

CDR population

The switch uses the contents of the ***AMASetHexABIndicator*** parameter to populate the HEXID field.

Application errors

None

Associated logs

None

Restrictions

None

AMAslpID (end)

Parameter definition

The ***AMAslpID*** parameter indicates that the switch should override normal switch-based recording and invoke AIN AMA record generation that is controlled by the SCP.

Message types

The following messages support the ***AMAslpID*** parameter:

- **`Analyze_Route`**
- **`Authorize_Termination`**
- **`Collect_Information`**
- **`Connect_To_Resource`**
- **`Continue`**
- **`Disconnect`**
- **`Originate_Call`**
- **`Send_To_Resource`**

Usage

Optional

Range of values

9 digits (encoded in Binary Coded Decimal format)

CDR Population rules

The switch uses the contents of the ***AMAslpID*** parameter to populate the SLPID and AMASC CDR fields.

Application errors

None

Associated logs

None

Restrictions

None

Parameter definition

The ***Amp1*** parameter is supported in an outgoing **Info_Collected** message and an incoming **Analyze_Route** response messages as a test parameter and does not affect call processing. If the ***Amp1*** parameter is received in a response message other than **Analyze_Route**, the rest of the message is processed as usual, and a CAIN101 log is generated to indicate an unexpected parameter.

Message types

The following incoming message supports the ***Amp1*** parameter:

- **Analyze_Route**

Usage

optional

CDR population

- If an ***Amp1*** parameter is received in a corresponding **Info_Collected** message by the SCP simulator, the incoming ***Amp1*** parameter values are used to populate the outgoing ***Amp1***.
- If no ***Amp1*** parameter is received in the corresponding query message by the SCP simulator, the ***Amp1*** parameter is populated with a set of default values corresponding to 0 for the unused fields and “Null current YR 1/1 00:00.”
- Only the AMPTime field of the ***Amp1*** parameter is used by the SSP and the SCP, the others fields are populated with 0s. The AMPTime field specifies the time and date that the SCP should use when processing a message and is used to test services that make time-sensitive call processing decisions.

Associated logs

CAIN101, VAMP901, VAMP902

Restrictions

None

AnswerIndicator (end)

Parameter definition

The parameter, when present, instructs the switch to provide answer supervision to the originating agent while the caller is connected to the resource. The switch sends answer indication to the caller in response to the **`Send_To_Resource`** if answer indication has not already been sent.

Message types

The following message supports the ***`AnswerIndicator`*** parameter:

- **`Send_To_Resource`**

Usage

Optional

Range of values

Presence or Absence

CDR population

None

Application errors

None

Associated logs

None

Restrictions

- ***`AnswerIndicator`*** is only used for SS7 and PRI originators.
- ***`AnswerIndicator`*** does not affect billing at the querying switch.
- Prior to UCS08, information the ***`AMAMeasure`*** parameter currently supplies was supplied by the ***`AnswerIndicator`*** parameter.

Parameter definition

This parameter contains the translated address or directory number of the called party.

Note 1: The ***CalledPartyID*** is translated to provide a route index through route advancing, or when the ***PrimaryTrunkGroup***, ***AlternateTrunkGroup***, or ***SecondAlternateTrunkGroup*** parameters are unavailable.

Note 2: This number is outpulsed when the ***OutpulseNumber*** is unavailable.

Note 3: This parameter replaces the original called party number for call processing and billing.

Note 4: This parameter is handled differently for LNP responses. Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information.

Message types

The following messages support the ***CalledPartyID*** parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

Maximum of 24 digits in AIN Digits format

Supported NOAs

Table 12-12 shows the nature of address (NOA) supported for the ***CalledPartyID*** parameter.

CalledPartyID (continued)

Table 12-12
CalledPartyID-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	Number does not conform to the National Numbering Plan.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
INTL	ISDN	International	International (Notes 1 and 2)
PART	ISDN	International	International partition numbers (Note 1)

Note 1: CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
Note 2: When the NOA is INTL, the INTL_XLA_TYPE parameter (in table CAINPARM) is checked for the NOA type to use to route the call.

—end—

CDR population

Table 12-13 shows the fields populated in the CDR when the `CalledPartyID` parameter is received.

Table 12-13
CalledPartyID CDR population

CDR field	Population
CALLEDNO	This field contains the called party number. This field stores up to 15 digits. Excess digits are truncated.
CNPREDIG	This field identifies the prefix digits of the called party.

Note: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

Application errors

Table 12-14 shows errors that can occur with regards to the ***CalledPartyID*** parameter.

Table 12-14
CalledPartyID nonfatal application errors

Error type	Log generated	Error action performed
Invalid address digits	CAIN100	The switch does not process the parameter. Processing continues, to allow for alternate routing parameters.
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues

Associated logs

CAIN100

Restrictions

None

Parameter definition

The ***CallingPartyID*** contains the address used for caller identification purposes. A presentation indicator is included which determines whether the number may be displayed to the called party.

ATTENTION
Normally, the switch does not build the ***CallingPartyID*** parameter. CAIN call processing on the switch builds the ***CallingPartyID*** parameter when directed by the SCP. Call origination is not taken into account. Refer to *FCC Report and Order for Further Notice of Proposed Rulemaking in the Matter of Rules and Policies Regarding Calling Number Identification Service* for more information on FCC regulations.

Message types

The following messages support the ***CallingPartyID*** parameter:

- **Analyze_Route**
- **Collect_Information**
- **Authorize_Termination**
- **Originate_Call**

Note: When the SCP sends the ***CallingPartyID*** parameter in an **Analyze_Route** message, the switch retains the parameter for future use. Refer to Chapter 10 for further information on the **Analyze_Route** message.

Usage

Optional

Note: ***CallingPartyID*** is only used for outpulsing when terminating to SS7 or PRI agents.

Range of values

Maximum of 24 digits in AIN Digits format

PRI terminations

For PRI terminations, the ***CallingPartyID*** is used to build the optional calling party number information element for the outgoing SETUP message when the ANIDELV option of the terminating PRI's tuple in table CALLATTR is set to CPNONLY or CPNPREF and the

CallingPartyNumber Presentation Indicator (PI) is set to "allowed.". For tandem scenarios (PRI-to-PRI or SS7-to-PRI), the SCP-provided ***CallingPartyID*** replaces any calling party parameters received from the originating agency.

If the ***CallingPartyID*** parameter is received from the SCP, the type of number (TON) and the numbering plan indicator (NPI) for the calling party information element which is to be placed into the outgoing SETUP message must be set. If the nature of address (NOA) for the ***CallingPartyID*** parameter is NATL, the TON is set to "ton_national" and the NPI is set to "npi_e164". If the NOA is anything other than NATL the TON is set to "ton_local" and the NPI is set to "npi_private".

SS7 terminations

For SS7 terminations, the ***CallingPartyID*** is used to build the optional calling party parameter for the outgoing IAM. For tandem scenarios (PRI-to-SS7 or SS7-to-SS7), the SCP-provided ***CallingPartyID*** replaces any calling party parameters received from the originating agency.

Supported NOAs

Table 12-15 shows the NOAs supported for the ***CallingPartyID*** parameter.

Table 12-15
CallingPartyID supported NOAs

NOA	Numbering Plan	Type of call	Description
UNK	UNK	UNK	
SUBR	ISDN	ONNET	ISDN
NATL	ISDN	OFFNET	Number conforms to the North American Numbering Plan.
INTL	ISDN	OFFNET	International
VPN	PRVT	ONNET	Number does not conform to the North American Numbering Plan.
NON_UNI_SUBR	PRVT	UNK	UNI – unique
NON_UNI_NATL	PRVT	UNK	UNI – unique
NON_UNI_INTL	PRVT	UNK	UNI – unique
TEST	PRVT	UNK	test line test code

CallingPartyID (continued)

CDR population

Table 12-16 shows the fields populated in the CDR when the `CallingPartyID` parameter is received.

Table 12-16
CallingPartyID CDR population

CDR field	Population
CLGPTYNO	This field contains the calling party digits. This field stores up to 15 digits. Excess digits are truncated.
PRESIND	This field identifies the presentation indicator of the calling party parameter. The following values are used: • 0 = Presentation allowed • 1 = Presentation restricted
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-17 shows errors that can occur with regards to the `CallingPartyID` parameter.

Table 12-17
CallingPartyID nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues.

Associated logs

CAIN100

Restrictions

- ***CallingPartyID*** is only used for outpulsing when terminating to SS7 or PRI agents.
- Delivery of the ***CallingPartyID*** in the outgoing message (IAM or Setup) is controlled by feature AD6849 (UCS ANI Delivery Enhancements). (Refer to the *UCS05 Software Release Document*, feature AD6849, for more information.) This feature introduces controls on parameter delivery based on the originating and terminating agencies. The following values can be datafilled for both originating and terminating agencies:
  - ALWAYS – deliver ***CallingPartyID***, ***ChargeNumber***, and OLI when available
  - NEVER – do not deliver ***CallingPartyID***, ***ChargeNumber***, or OLI
  - CPNONLY – deliver only the ***CallingPartyID*** when available
  - CGNONLY – deliver only the ***ChargeNumber*** and OLI when available

  CAIN overrides the controls placed on originating agencies. Therefore, the controls placed on the originating agency are not used in determining parameter delivery. This override has the same effect as datafilling ALWAYS for the originating agency. The terminating controls are still functional.

- Feature AX0993 provides the customer the option of choosing between an ISUP CallingPartyNumber or ChargeNumber when outpulsing on a terminating PRI and DAL- TIE agents. The ANIDELV option of table CALLATTR will include sub-option ANIDELV with values CPNONLY, CGNONLY, CPNPREF and CGNPREF that will allow the user to specify whether an incoming ISUP CallingPartyNumber or ChargeNumber parameter should be used to populate an outgoing CgPN IE. If the ISUP CPN is used to populate the outgoing CLID, the PI bit screening occurs. (Refer to the *UCS12 Software Release Document*, feature AX0993, for more information.)
- Delivery of the ***CallingPartyID*** is subject to in-switch feature restrictions.
- Table RTEATTR provides a way to control the ***CallingPartyID*** and ***ChargeNumber*** delivery based on a terminating route. The controls in table RTEATTR, when provisioned, override any pre-existing controls.

  When the ***ChargeNumber*** is included in table RTEATTR, the ***ChargeNumber*** and OLI are delivered when:
  - A ***ChargeNumber*** is available.

CallingPartyID (end)

— No ***CallingPartyID*** is being delivered or the ***CallingPartyID***’s address is different than the ***ChargeNumber***’s address.

When the ***CallingPartyID*** is included in table RTEATTR, the ***CallingPartyID*** and OLI are delivered when:

— A ***CallingPartyID*** is available.

— Excluding the ***CallingPartyID*** stops delivery of the appropriate parameters.

Note: (Refer to the *UCS06 Software Release Document*, feature AD8792, for more information.)

Parameter definition

This parameter contains carrier selection information and carrier identification code (CIC) to route the call.

Message types

The following messages support the ***`Carrier`*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Note: When the SCP sends the ***`Carrier`*** parameter in an **`Analyze_Route`** message, the switch retains the parameter for future use. Refer to Chapter 10 for further information on the **`Analyze_Route`** message.

Usage

Optional

Range of values

Carrier Selection Field

- NO_INDICATION
- PRESUBSCRIBED_AND_NOT_INPUT
- PRESUBSCRIBED_AND_INPUT
- PRESUBSCRIBED_AND_NO_INDICATION
- NOT_PRESUBSCRIBED_AND_INPUT

Carrier ID field

- 0000–9999

CDR population

If the CIC digits of the ***`AMADigitsDialedWC`*** parameter are not present, CIC and CARRSEL CDR fields are populated from this parameter when it is used to route the call. Also, when routing through direct termination, the CDR is populated from this parameter when present.

Application errors

The following table shows errors that can occur with regards to the ***`Carrier`*** parameter.

Carrier (end)

Table 12-18
AlternateCarrier nonfatal application errors

Error type	Log generated	Error action performed
CIC not datafilled in table CICRTE	CAIN100	The parameter is ignored.

Associated logs

CAIN100

Restrictions

None

Parameter definition

The CAINTEST command and the VAMP901 log display the `CarrierUsage` parameter when received, but this parameter does not affect call processing.

Note: This parameter is only usable with the CAINTEST command and the SCP simulator.

Message types

The **`Analyze_Route`** message supports the `CarrierUsage` parameter.

Usage

Optional

Range of values

The range of values are:

- alwaysOverride.
- onlyInterLATAOverride.
- overridePICsOfNOCsSent.

CDR population rules

None

Associated logs

VAMP901

Restrictions

None

ChargeNumber (continued)

Parameter definition

This parameter holds the billing number. Based on the NOA, this parameter may contain an ANI, authcode, credit card number, or an N00 number. The appropriate CDR fields are updated with the billing number received from the SCP.

Note: The ANI is delivered, when available, on the outgoing trunk and is subject to existing ANI delivery restrictions.

Message types

The following messages support the ***`ChargeNumber`*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Note: When the SCP sends the ***`ChargeNumber`*** parameter in an **`Analyze_Route`** message, the switch retains the parameter for future use. Refer to Chapter 10 for further information on the **`Analyze_Route`** message.

Usage

Optional

Range of values

Maximum of 13 digits (for I2ANI) and 11 digits (for ANI). A maximum of 24 digits for AUTH, MCCS, and N00.

Additional information

The following sections contain information relevant to the ***`ChargeNumber`*** parameter.

SCP delivers NOA of ANI

When the SCP delivers an NOA of ANI, the charge number digits replace the existing ANI for the remainder of the call. The information digits are set to 00.

SCP delivers NOA of I2ANI

When the SCP delivers an NOA of I2ANI, the charge number digits replace the existing information digits and ANI for the remainder of the call.

PTS terminations

When terminating to a PTS trunk, the SCP delivers information digits and ANI subject to existing in-switch restrictions and limitations. Previously, the outgoing PTS ANI was populated with information from the outgoing ISUP ***ChargeNumber*** parameter. Feature AX0993 now allows datafill of options CPNPREF, CPNONLY, CGNPREF and CGNONLY under sub-option ANIDELV in table CALLATTR to decide between a ***CallingPartyNumber*** and ***ChargeNumber*** for outpulsing. (Refer to the *UCS12 Software Release Document*, feature AX0993, for more information.)

SS7 terminations

When terminating to an SS7 trunk, CAIN call processing builds the ISUP Charge Number and OLI parameters to be placed in the outgoing IAM.

Tandem (SS7-to-SS7) terminations

The SCP-provided values replace any charge number or OLI parameters received from the incoming trunk.

PRI terminations

CAIN call processing builds the calling party number information element using the charge number digits when the following are true:

- The originating agent is not a PRI agent.
- The ANIDELV option of the terminating PRI's tuple in table CALLATTR is set to CGNONLY or CGNPREF.
- The CAIN ***ChargeNumber*** will also be used to populate the PRI ***CallingPartyNumber*** Information Element (CgPN IE) if ANIDELV is set to CPNREF, but the ***CallingPartyNumber*** Presentation Indicator (PI) bit is set to "restricted."

If the ***ChargeNumber*** parameter is received from the SCP, the type of number (TON) and the numbering plan indicator (NPI) for the calling party information element which is to be placed into the outgoing SETUP message must be set. The TON is set to "ton_unknown" and the NPI is set to "npi_e164," regardless of the value of the nature of address (NOA) for the ***ChargeNumber*** parameter.

Supported NOAs

Table 12-19 shows the NOAs supported by the UCS08 software release for the ***ChargeNumber*** parameter.

ChargeNumber (continued)

Table 12-19
ChargeNumber supported NOAs

NOA	Numbering Plan	Description
ANI, NATL	ISDN	The parameter contains only an ANI. If a ***ChargePartyStationType*** is not returned, then information digits of 00 are assumed by the switch.
I2ANI	PRVT	The parameter contains two information digits and an ANI.
AUTH	PRVT	The parameter contains an authcode.
MCCS	PRVT	The parameter contains a credit card number.
N00	PRVT	The parameter contains an N00 number.
SUBR	ISDN	ANI of the calling party; Subscriber number
NATL	ISDN	ANI of the calling party; National number

CDR population

Table 12-20 shows the fields populated in the CDR when the ***ChargeNumber*** parameter is received.

Table 12-20
ChargeNumber CDR population

CDR field	Population
INFODIG	This field contains the information digits when the NOA is ANI or I2ANI. ***Note:*** When NOA is ANI, INFODIG = 00.
ANISP	This field contains the ANI when the NOA is ANI or I2ANI. This field can store up to 10 digits. Excess digits are truncated.
BILLNUM	This field contains the billing number when the NOA is authcode, MCCS, or N00. This field can store up to 23 digits. Excess digits are truncated.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

The following table shows errors that can occur with regards to the `ChargeNumber` parameter.

Table 12-21
ChargeNumber nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	N00 NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues. ***Note:*** This error is only valid for NOAs of I2ANI and ANI.

Associated logs

CAIN100

Restrictions

- If the NOA returned by the SCP is ANI, then this new ANI is used at any subsequent triggers when required for criteria matching.
- The new data received in the `ChargeNumber` parameter is delivered to the SCP in any subsequent queries until the SCP provides a new `ChargeNumber`.
- Delivery of the `ChargeNumber` and OLI parameters in the outgoing IAM message is controlled by feature AD6849 (UCS ANI Delivery Enhancements). (Refer to the *UCS05 Software Release Document*, feature AD6849, for more information.) This feature introduces controls on parameter delivery based on the originating and terminating agencies. The following values can be datafilled for both originating and terminating agencies:
  - ALWAYS – deliver `CallingPartyID`, `ChargeNumber`, and OLI when available
  - NEVER – do not deliver `CallingPartyID`, `ChargeNumber`, or OLI
  - CPNONLY – deliver only the `CallingPartyID` when available
  - CGNONLY – deliver only the `ChargeNumber` and OLI when available

ChargeNumber (end)

CAIN overrides the controls placed on originating agencies. Therefore, the controls placed on the originating agency are not used in determining parameter delivery. This override has the same effect as datafilling ALWAYS for the originating agency. The terminating controls are still functional.

- Feature AX0993 provides the customer the option of choosing between an ISUP***CallingPartynumber*** or ***ChargeNumber*** when outpulsing on a terminating PRI and DAL- TIE agents. The ANIDELV option of table CALLATTR will include sub-option ANIDELV with values CPNONLY, CGNONLY, CPNPREF and CGNPREF that will allow the user to specify wheather an incoming ISUP***CallingPartynumber*** or ***ChargeNumber*** should be used to populate an outgoing CgPN IE. If the ISUP CPN is used to populate the outgoing CLID, the PI bit screening occurs. (Refer to the *UCS12 Software Release Document*, feature AX0993, for more information.)
- Delivery of the ***ChargeNumber*** parameter is subject to in-switch feature restrictions.
- Table RTEATTR provides a way to control the ***CallingPartyID*** and ***ChargeNumber*** delivery based on a terminating route. The controls in table RTEATTR, when provisioned, override any pre-existing controls.

  When the ***ChargeNumber*** is included in table RTEATTR, the ***ChargeNumber*** and OLI are delivered when:

  — A ***ChargeNumber*** is available.

  — No ***CallingPartyID*** is being delivered or the ***CallingPartyID***'s address is different than the ***ChargeNumber***'s address.

  When the ***CallingPartyID*** is included in table RTEATTR, the ***CallingPartyID*** and OLI are delivered when:

  — A ***CallingPartyID*** is available.

  — Excluding the ***CallingPartyID*** will stop delivery of the appropriate parameters.

  Note: (Refer to the *UCS06 Software Release Document*, feature AD8792, for more information.)

Parameter definition

This parameter contains the information digits for the call.

Message types

The following messages support the ***ChargePartyStationType*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Note: When the SCP sends the ***ChargePartyStationType*** parameter in an **`Analyze_Route`** message, the switch retains the parameter for future use. Refer to Chapter 10 for further information on the **`Analyze_Route`** message.

Usage

Optional

Range of values

0–99

CDR population

The ***ChargePartyStationType*** parameter populates the INFODIG CDR field.

Application errors

None

Associated logs

None

Restrictions

None

Parameter definition

This parameter specifies whether a control is an SCP overload control, or an SMS initiated control. ***`ControlCauseIndicator`*** also contains the number of digits to which the control is applied.

Message types

The following message supports the ***`ControlCauseIndicator`*** parameter:

- **`ACG`**

Usage

Required

Range of values

Table 12-22 shows the range of values for each field that may be present in the `ControlCauseIndicator` parameter.

Table 12-22
ControlCauseIndicator field values

Field name	Range of values	Explanation
SCP Overload Controls Indicator	0	No SCP overload controls
	1	SCP overload controls
SMS Initiated Controls Indicator	0	No SMS initiated controls
	1	SMS initiated controls
Number of Digits to which Control is applied	000001	1 digit control
	000010	2 digit control
	000011	3 digit control
	000100	4 digit control
	000101	5 digit control
	000110	6 digit control
	000111	7 digit control
	001000	8 digit control
	001001	9 digit control
	001010	10 digit control

CDR population

None

Application errors

None

ControlCauseIndicator (end)

Associated logs

None

Restrictions

None

Parameter definition

The ***CsID*** parameter specifies to which call segment the message applies.

Message types

The following messages support the ***CsID*** parameter:

- **`Acknowledge`**
- **`Disconnect_Leg`**
- **`O_Abandon`**

Usage

Optional

Range of values

1-2

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Parameter definition

This parameter, when present, contains the address of an intelligent peripheral. The call is routed to this address.

Message types

The following messages support the ***`DestinationAddress`*** parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

Maximum of 18 digits for VPN and national numbers in AIN Digits format

Note: The CAIN application allows 18-digit National numbers, but the switch may be unable to process this number.

Supported NOAs

Table 12-23 shows the nature of address (NOA) supported for the ***`DestinationAddress`*** parameter.

Table 12-23
DestinationAddress-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	Number does not conform to the North American Numbering Plan.
NATL	ISDN	OFFNET	Number conforms to the North American Numbering Plan.
INTL	ISDN	International	International (Note 1)
PART	ISDN	International	International partition numbers (Note 1)

Note 1: CAIN prefixes international numbers received from the SCP with 011 (except for SS7 Global-IMTs).
Note 2: When the NOA is INTL, the INTL_XLA_TYPE parameter (in table CAINPARM) is checked for the NOA type to use to route the call.

—end—

CDR population

None

Application errors

Table 12-24 shows errors that can occur with regards to the ***DestinationAddress*** parameter.

Table 12-24
DestinationAddress nonfatal application errors

Error type	Log generated	Error action performed
Invalid address digits	CAIN100	The switch does not process the parameter. Processing continues, to allow for alternate routing parameters.
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues.

Associated logs

CAIN100

Restrictions

None

DisconnectFlag (continued)

Parameter definition

The parameter indicates whether the call should be disconnected after a **`Send_To_Resource`** or **`Connect_To_Resource`** interaction is completed.

Message types

The following messages support the ***`DisconnectFlag`*** parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

Presence or Absence

CDR population

None

Application errors

Table 12-25 shows errors associated with the ***`DisconnectFlag`*** parameter.

Table 12-25
DisconnectFlag nonfatal errors

Error type	Log generated	Error action performed
`Send_To_Resource` or **`Connect_To_Resource`** in a Response package without ***`DisconnectFlag`***	CAIN100	Process request as though ***`DisconnectFlag`*** were present
`Send_To_Resource` or **`Connect_To_Resource`** in a Conversation package with a ***`DisconnectFlag`***	CAIN100	Process request as though ***`DisconnectFlag`*** were not present

Associated logs

CAIN100

Restrictions

None

DisplayText (end)

Parameter definition

This parameter provides display data that is sent across the network to the end user.

Message types

The following messages support the ***`DisplayText`*** parameter:

- **`Authorize_Termination`**
- **`Originate_Call`**

Usage

Optional

Range of values

Supported values for the DisplayInformation specifier within the ***`DisplayText`*** parameter include:

callingPartyName,
originalCalledName,
redirectingName

In addition, a display information string is associated with each value. When duplicate DisplayInformation specifiers are received, the string closest to the end of the message is retained and all other strings are discarded.

CDR population

None

Application errors

None

Associated logs

CAIN100

Restrictions

The ***`DisplayText`*** parameter is only valid for PRI terminations.

Parameter definition

This parameter is used to match the SCP **`Send_Notification`** message with the switch **`Termination_Notification`** message.

Message types

The following message supports the ***`EchoData`*** parameter:

- **`Send_Notification`**

Usage

Required

Range of values

A string of 12 hexadecimal digits

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

EDPNotification (end)

Parameter definition

This parameter contains a list of EDPs that should be armed to send an EDP-Notification to the SCP.

Message types

The following message supports the ***`EDPNotification`*** parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

The values of this parameter are contained in a bit string, with each bit corresponding to an associated event. The value for each bit can be one of the following:

0 = Notification is not requested for the event
1 = Notification is requested for the event

Note: Currently the oAnswer, oDisconnect, and oTermSeized events are supported.

CDR population

None

Application errors

Refer to Chapter 3, “Event processing,” for information on arming restrictions.

Associated logs

None

Restrictions

Refer to Chapter 3, “Event processing,” for information on arming restrictions.

Parameter definition

This parameter contains a list of EDPs that should be armed to send an EDP-Request to the SCP.

Message types

The following message supports the ***`EDPRequest`*** parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

The values of this parameter are contained in a bit string, with each bit corresponding to an associated event. The value for each bit can be one of the following:

0 = EDP Request is not requested for the event
1 = EDP Request is requested for the event

Note: Currently networkBusy, oAbandon, oCalledPartyBusy, oDisconnect, oNoAnswer, switchHookFlash, and timeout events are supported.

CDR population

None

Application errors

Refer to Chapter 3, “Event processing,” for information on arming restrictions.

Associated logs

None

Restrictions

Refer to Chapter 3, “Event processing,” for information on arming restrictions.

ExtensionParameter (end)

ATTENTION
Extension parameters require the CAIN0200 SOC option. Refer to Volume 5, Chapter 5, "NetworkBuilder SOC functionality," for more information. A CAIN102 log is generated when the switch receives extension parameters without the CAIN0200 SOC option.

Parameter definition

The ***ExtensionParameter*** set is provided to receive optional feature-specific information from the SCP. Refer to Table 12-NO TAG for a complete list of incoming message extension parameters supported by CAIN call processing. Detailed descriptions of individual extension parameters follow in this chapter.

Note: CAINXDFT provides extension parameters to be used if the SCP message does not contain valid extension parameters or values. The use of extension parameters in CAINXDFT is not dependent on the CAIN0200 SOC option.

Parameter definition

The ***ForwardCallIndicator*** parameter is used to indicate that a local number portability (LNP) check has been performed by the SCP by setting bit M of the parameter.

Note: Refer to the *UCS DMS-250 Local Number Portability Application Guide* for more information on LNP.

Message types

The following messages support the ***ForwardCallIndicator*** parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

2 octets (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P)

CDR population

None

Application errors

None

Associated logs

None

Restrictions

Only bit M of the FCI parameter is used.

GapDuration (continued)

Parameter definition

The ***GapDuration*** parameter contains the length (in seconds) that an ACG control should be applied before it times out and is removed from the ACG control list by the switch.

Message types

The following message supports the ***GapDuration*** parameter:

- **ACG**

Usage

Required

Range of values

Table 12-26 shows the valid gap durations for the ***GapDuration*** parameter.

Table 12-26
ACG Control Gap Durations

Enumerated Type	Enumerated Value	Time (in seconds)
no1Second	1	1
no2Second	2	2
no4Second	3	4
no8Second	4	8
no16Second	5	16
no32Second	6	32
no64Second	7	64
no128Second	8	128
no256Second	9	256
no512Second	10	512
no1024Second	11	11024
no2048Second	12	2048
infinity	13	

Note: If the given Gap Duration value is infinity, and the given Gap Interval is also infinity, a value of 4096 seconds is substituted for the Gap Duration value.

CDR Population rules

None

Application errors

None

Associated logs

None

Restrictions

None

Parameter definition

The ***`GapInterval`*** parameter contains the minimum length (in seconds) that the switch must wait before sending another query of the type under the control.

Message types

The following message supports the ***`GapInterval`*** parameter:

- **`ACG`**

Usage

Required

Range of values

Table 12-27 shows the valid gap intervals for SOCCs.

Table 12-27
CAIN SOCC gap intervals (National Gap Intervals)

Enumerated Type	Enumerated Value	Time (in seconds)
removeGapControl	0	
no0Second	1	0
no010Seconds	2	0.1
no025Seconds	3	0.25
no050Seconds	4	0.5
no1Seconds	5	1.0
no2Seconds	6	2.0
no5Seconds	7	5.0
no10Seconds	8	10.0
no15Seconds	9	15.0
no30Seconds	10	30.0
no60Seconds	11	60.0
no120Seconds	12	120.0
no300Seconds	13	300.0
no600Seconds	14	600.0
stopAllCalls	15	infinity

GapInterval (continued)

Table 12-28 shows the valid gap intervals for SCP Overload Controls.

Table 12-28
CAIN SCP Overload Control gap intervals (Private Gap Intervals)

Enumerated Type	Enumerated Value	Time (in seconds)
no0Seconds	0	0
no3Second	1	3
no4Seconds	2	4
no6Seconds	3	6
no8Seconds	4	8
no11Seconds	5	11
no16Seconds	6	16
no22Seconds	7	22
no30Seconds	8	30
no42Seconds	9	42
no58Seconds	10	58
no81Seconds	11	81
no112Seconds	12	112
no156Seconds	13	156
no217Seconds	14	217
no300Seconds	15	300
removeGapControl	16	
no010Seconds	17	0.10
no025Seconds	18	0.25
no050Seconds	19	0.50
no050Seconds	20	1
no2Seconds	21	2

CDR Population rules

None

Application errors

None

Associated logs

None

Restrictions

None

GenericAddressList (continued)

Parameter definition

CAIN call processing uses the ***GenericAddressList*** parameter to provide the parameters listed in Table 12-29.

Table 12-29
GenericAddressList parameters

GenericAddressList parameter	Definition
`AlternateOutpulseNo`	This parameter, when present, contains an outpulse number to be used with the ***AlternateTrunkGroup***.
`SecondAlternateOutpulseNo`	This parameter, when present, contains an outpulse number to be used with the ***SecondAlternateTrunkGroup***.
`OverflowRoutingNo`	This parameter, when present, contains an additional routing number.
`DialedNumberInwardService`	This parameter contains a DNIS value specific to the Dialed Number Inward Service specified
`PortedDialedNo`	This parameter contains the called party address of a successful LNP query

Message types

The following message supports the ***GenericAddressList*** parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

Refer to the appropriate ***GenericAddressList*** parameter for more information.

Restrictions

The generic address parameters are used only for SCP-to-UCS DMS-250 communication. When terminating to an SS7, ISUP generic address parameters associated with the above parameters are not built into the outgoing IAM.

Note: This restriction does not apply to the `DialedNumberInwardService` or `PortedDialedNo` parameters.

GenericAddressList AlternateOutpulseNo (continued)

GenericAddressList parameter definition

This parameter contains the digits to be outpulsed when the ***AlternateTrunkGroup*** number to outpulse flag is set to 0.

An NOA is required for outpulsing. The following apply:

- On PTS FGD terminations, the NOA is used to determine the dialing plan for outpulsing.
- On some agencies, international numbers are outpulsed differently than VPN and national numbers.
- On SS7 terminations, the NOA is used to set the NOA of the ISUP called party number parameter in the outgoing IAM.
- If the ***AlternateTrunkGroup*** number to outpulse is set to 0, and this parameter is not present, the ***OutpulseNumber*** parameter is used, and then the ***CalledPartyID***.

Message types

The following messages support the `AlternateOutpulseNo` parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

Maximum of 15 digits for international numbers; 18 digits for VPN and national numbers.

Supported NOAs

Table 12-30 shows the NOAs supported for the `AlternateOutpulseNo` parameter.

Table 12-30
AlternateOutpulseNo-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	An incoming Outpulse Number with an NOA of cain_vpn_number is considered a National number.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
PART	ISDN	International Partitioned	An incoming Outpulse Number with an NOA of cain_partitioned_number is converted to international.
SUBR	ISDN	Subscriber number	
UNK	ISDN	Not applicable	
SUBR_OP	ISDN	Subscriber number, operator requested (0+ call)	
NATL_OP	ISDN	National number, operator requested (0+ call)	

—continued—

GenericAddressList AlternateOutpulseNo (continued)

Table 12-30
AlternateOutpulseNo-supported NOAs (continued)

NOA	Numbering Plan	Type of call	Description
INTL_OP	ISDN	International number, operator requested (0+ call)	
NO_NUM_OP	ISDN	International number, operator requested (0–, 10XXX + 0(0), or 00– call)	
NO_NUM_CT	ISDN	No address present, cut-through call to carrier	
—end—			

CDR population

Table 12-31 shows the fields populated in the CDR when the `AlternateOutpulseNo` parameter is outpulsed.

Table 12-31
AlternateOutpulseNo CDR population

CDR field	Population
OUTPULNO	This field contains the digits that are outpulsed on the terminating trunk. This field stores up to 15 digits. Excess digits are truncated.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-32 shows errors that can occur with regards to the `AlternateOutpulseNo` parameter.

Table 12-32
AlternateOutpulseNo nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues.

Associated logs

CAIN100

Restrictions

CAIN call processing uses existing switch logic to perform outpulsing. Each trunk performs a series of steps to determine the outpulsed address. In some cases (dependent on agents and features used), digits are stripped or modified. Therefore, the actual number outpulsed may not match the number received from the SCP. The following two examples provide scenarios.

1 The SCP provides a 10-digit national ***`OutpulseNumber`*** and a trunk parameter indicating DAL termination. However, field DIGSOUTP in table TRKGRP indicates that only seven digits should be outpulsed on the terminating DAL. Therefore, only the last seven digits of the ***`OutpulseNumber`*** are outpulsed.

2 The SCP provides a 10-digit national ***`OutpulseNumber`*** and a trunk parameter indicating FGD termination. If the NPA of the ***`OutpulseNumber`*** matches the connecting NPA (field CONNGNPA in table TRKGRP), then only the last seven digits are outpulsed. Otherwise, the entire 10-digit number is outpulsed.

Table RTEATTR provides the capability to manipulate parameters in the outgoing IAM. Normally, CAIN allows these changes to take place. However, when the SCP returns an outpulse number, the CAIN framework

GenericAddressList AlternateOutpulseNo (end)

overrides the datafill in table RTEATTR and outpulses the SCP-supplied digits without any restrictions.

GenericAddressList parameter definition

This parameter, when present, contains a dialed number inward service (DNIS) value specific to the DNIS service specified.

Message types

The following messages support the `DialedNoInwardService` parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

Type of Address= 0000 0000 (Dialed Number)
Nature of Address=set depending on type of address field
Odd/Even= set depending on number of digits to follow
Encoding Scheme= 00 (Presentation Allowed)
Numbering Plan= 000 to 111 (Unknown)
Address Signals= 1–28 Binary Coded Digits

Supported NOAs

Table 12-33 shows the NOAs supported for the `DialedNoInwardService` parameter.

Table 12-33
DialedNoInwardService-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	Number does not conform to the National Numbering Plan.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)

GenericAddressList DialedNoInwardService (continued)

CDR population

Table 12-34 shows the fields populated in the CDR when the `DialedNoInwardService` parameter is received.

Note: The CDR is not populated with this value until the switch attempts to route using the `DialedNoInwardService` parameter.

Table 12-34
DialedNoInwardService CDR population

CDR field	Population
CALLEDNO	This field contains the called party number. This field stores up to 15 digits. Excess digits are truncated.
CNPREDIG	This field identifies the prefix digits of the called party.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-35 shows errors that can occur with regards to the `DialedNoInwardService` parameter.

Table 12-35
DialedNoInwardService nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues.

Associated logs

CAIN100

Restrictions

None

GenericAddressList parameter definition

The parameter contains an overflow routing number for standard routing.

Message types

The following messages support the `OverflowRoutingNo` parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

Maximum of 18 digits for VPN and national numbers
Maximum of 15 digits for international numbers

Note: The CAIN application allows 18-digit national numbers, but the switch may be unable to process this number.

Supported NOAs

Table 12-36 shows the NOAs supported for the `OverflowRoutingNo` parameter.

Table 12-36
OverflowRoutingNo-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	Number does not conform to the National Numbering Plan.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
Note: When the NOA is INTL, the INTL_XLA_TYPE parameter (in table CAINPARM) is checked for the NOA type to use to route the call.			

—continued—

Table 12-36
OverflowRoutingNo-supported NOAs (continued)

NOA	Numbering Plan	Type of call	Description
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
PART	ISDN	International	International partition numbers ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
NS	N/A	International	Number is specific to the terminating network that provides GVNS service.

Note: When the NOA is INTL, the INTL_XLA_TYPE parameter (in table CAINPARM) is checked for the NOA type to use to route the call.

—end—

GenericAddressList OverflowRoutingNo (continued)

CDR population

Table 12-37 shows the fields populated in the CDR when the switch attempts to route using the `OverflowRoutingNo` parameter.

Table 12-37
OverflowRoutingNo CDR population

CDR field	Population
DIALEDNO	This field contains the dialed number. This field stores up to 15 digits. Excess digits are truncated.
CNPREDIG	This field identifies the prefix digits of the called party.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-38 shows errors that can occur with regards to the `OverflowRoutingNo` parameter.

Table 12-38
OverflowRoutingNo nonfatal application errors

Error type	Log generated	Error action performed
Invalid address digits	CAIN100	The switch does not process the parameter. Processing continues, to allow for alternate routing parameters.
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues

Associated logs

CAIN100

Restrictions

None

GenericAddressList PortedDialedNo (continued)

GenericAddressList parameter definition

This parameter contains the called party address of a successful LNP query.

Message types

The following messages support the `PortedDialedNo` parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

Type of Address= 1100 0000 (Ported Dialed Number)
Nature of Address= set depending on type of address field
Odd/Even= set depending on number of digits to follow
Encoding Scheme= 00 (Presentation Allowed)
Numbering Plan= 000 to 111 (Unknown)
Address Signals= 1–28 Binary Coded Digits

Supported NOAs

Table 12-39 shows the NOAs supported for the `PortedDialedNo` parameter.

Table 12-39
PortedDialedNo-supported NOAs

NOA	Numbering Plan	Type of call	Description
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)

CDR population

Table 12-40 shows the fields populated in the CDR when the `PortedDialedNo` parameter is received.

Table 12-40
PortedDialedNo CDR population

CDR field	Population
PORTEDNO	The field contains the called party number. This field stores up to 10 digits. Excess digits are truncated.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide* or the *UCS DMS-250 Local Number Portability Application Guide*	

Application errors

None

Associated logs

None

Restrictions

None

GenericAddressList SecondAlternateOutpulseNo (continued)

GenericAddressList parameter definition

This parameter contains the digits to be outpulsed when the ***SecondAlternateTrunkGroup*** number to outpulse is set to 0.

An NOA is required for outpulsing. The following apply:

- On PTS-FGD terminations, the NOA is used to determine the dialing plan for outpulsing.
- On some agencies, international number are outpulsed differently than VPN and national numbers.
- On SS7 terminations, the NOA is used to set the NOA of the ISUP called party number parameter in the outgoing IAM.
- If the ***SecondAlternateTrunkGroup*** number to outpulse is set to 0, and this parameter is not present, the ***OutpulseNumber*** parameter is used.

Message types

The following messages support the `SecondAlternateOutpulseNo` parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

Maximum of 15 digits for international numbers; 18 digits for VPN and national numbers.

Supported NOAs

Table 12-41 shows the NOAs supported for the `SecondAlternateOutpulseNo` parameter.

GenericAddressList SecondAlternateOutpulseNo (continued)

Table 12-41
SecondAlternateOutpulseNo-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	An incoming Outpulse Number with an NOA of cain_vpn_number is considered a National number.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
PART	ISDN	International Partitioned	An incoming Outpulse Number with an NOA of cain_partitioned_number is converted to international.
SUBR	ISDN	Subscriber number	
UNK	ISDN	Not applicable	
SUBR_OP	ISDN	Subscriber number, operator requested (0+ call)	
NATL_OP	ISDN	National number, operator requested (0+ call)	

—continued—

GenericAddressList SecondAlternateOutpulseNo (continued)

Table 12-41
SecondAlternateOutpulseNo-supported NOAs (continued)

NOA	Numbering Plan	Type of call	Description
INTL_OP	ISDN	International number, operator requested (0+ call)	
NO_NUM_OP	ISDN	International number, operator requested (0–, 10XXX + 0(0), or 00– call)	
NO_NUM_CT	ISDN	No address present, cut-through call to carrier	
—end—			

CDR population

Table 12-42 shows the fields populated in the CDR when the `SecondAlternateOutpulseNo` parameter is outpulsed.

Table 12-42
SecondAlternateOutpulseNo CDR population

CDR field	Population
OUTPULNO	This field contains the digits that are outpulsed on the terminating trunk. This field stores up to 15 digits. Excess digits are truncated.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-43 shows errors that can occur with regards to the `SecondAlternateOutpulseNo` parameter.

Table 12-43
SecondAlternateOutpulseNo nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated and parameter processing continues.

GenericAddressList SecondAlternateOutpulseNo (end)

Associated logs

CAIN100

Restrictions

CAIN call processing uses existing switch logic to perform outpulsing. Each trunk performs a series of steps to determine the outpulsed address. In some cases (dependent on agents and features used), digits are stripped or modified. Therefore, the actual number outpulsed may not match the number received from the SCP. The following two examples describes these scenarios.

1. The SCP provides a 10-digit national ***OutpulseNumber*** and a trunk parameter indicating DAL termination. However, field DIGSOUTP in table TRKGRP indicates that only seven digits should be outpulsed on the terminating DAL. Therefore, only the last seven digits of the ***OutpulseNumber*** are outpulsed.
2. The SCP provides a 10-digit national ***OutpulseNumber*** and a trunk parameter indicating FGD termination. If the NPA of the ***OutpulseNumber*** matches the connecting NPA (field CONNGNPA in table TRKGRP), then only the last seven digits are outpulsed. Otherwise, the entire 10-digit number is outpulsed.

Table RTEATTR provides the capability to manipulate parameters in the outgoing IAM. Normally, CAIN allows these changes to take place. However, when the SCP returns an outpulse number, the CAIN framework overrides the datafill in table RTEATTR and outpulses the SCP-supplied digits without any restrictions.

Parameter definition

The ***GlobalTitleAddress*** parameter specifies the global title address of an ACG control.

Message types

The following message supports the ***GlobalTitleAddress*** parameter:

- **ACG**

Usage

Required

Range of values

10 digits (encoded in Binary Coded Decimal format)

CDR Population rules

None

Application errors

None

Associated logs

None

Restrictions

None

Parameter definition

The ***LegID*** parameter identifies a leg in a call segment.

Message types

The following messages support the ***LegID*** parameter:

- **`Connect_To_Resource`**
- **`Disconnect_Leg`**

Usage

Optional

Range of values

The possible values for the ***LegID*** parameter are as follows:

- 0 - controlling leg
- 1 - passive leg
- 2 - additional passive leg

CDR population

None

Application errors

None

Associated logs

None

Restrictions

For **`Connect_To_Resource`** on reorigination, the ***LegID*** parameter is always set to 0.

Parameter definition

Message types

The following message supports the ***ONoAnswerTimer*** parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

1–120

CDR population

None

Application errors

Table 12-44 shows errors that can occur with regards to the ***ONoAnswerTimer*** parameter.

Table 12-44
ONoAnswerTimer parameter nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
ONoAnswerTimer parameter is received but the **`O_No_Answer`** EDP is not requested in the **`Request_Report_BCM_Event`**	CAIN100	No	The parameter is ignored.

Associated logs

CAIN100

Restrictions

None

OutpulseNumber (continued)

Parameter definition

This parameter contains the digits to be outpulsed when the ***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, or ***`SecondAlternateTrunkGroup`*** number to outpulse is set to 0. Multiple outpulse numbers may be defined in the ***`GenericAddressList`***. (Refer to the ***`GenericAddressList`*** parameter section for more details.)

An NOA is required for outpulsing. The following apply:

- For PTS-FGD terminations, the NOA is used to determine the dialing plan used for outpulsing.
- On some agencies, international number are outpulsed differently than VPN and national numbers.
- For SS7 terminations, the NOA is used to set the NOA of the ISUP called party number parameter in the outgoing IAM.

Message types

The following messages support the ***`OutpulseNumber`*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

Maximum of 15 digits for international numbers; 18 digits for VPN and national numbers.

Supported NOAs

Table 12-45 shows the NOAs supported for the ***`OutpulseNumber`*** parameter.

OutpulseNumber (continued)

Table 12-45
OutpulseNumber-supported NOAs

NOA	Numbering Plan	Type of call	Description
VPN	PRVT	ONNET	An incoming Outpulse Number with an NOA of cain_vpn_number is considered a National number.
NATL	ISDN	OFFNET	Number conforms to the National Numbering Plan.
INTL	ISDN	International	International ***Note:*** CAIN prefixes international numbers received from the SCP with 011. (Except for SS7 Global-IMTs.)
PART	ISDN	International Partitioned	An incoming Outpulse Number with an NOA of cain_partitioned_number is converted to international.
SUBR	ISDN	Subscriber number	
UNK	ISDN	Not applicable	
SUBR_OP	ISDN	Subscriber number, operator requested (0+ call)	
NATL_OP	ISDN	National number, operator requested (0+ call)	
INTL_OP	ISDN	International number, operator requested (0+ call)	

—continued—

OutpulseNumber (continued)

Table 12-45
OutpulseNumber-supported NOAs (continued)

NOA	Numbering Plan	Type of call	Description
NO_NUM_OP	ISDN	International number, operator requested (0–, 10XXX + 0(0), or 00– call)	
NO_NUM_CT	ISDN	No address present, cut-through call to carrier	
—end—			

CDR population

Table 12-46 shows the fields populated in the CDR when the ***OutpulseNumber*** parameter is outpulsed.

Table 12-46
OutpulseNumber CDR population

CDR field	Population
OUTPULNO	This field contains the digits that are outpulsed on the terminating trunk. This field stores up to 15 digits. Excess digits are truncated.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-47 shows errors that can occur with regards to the ***OutpulseNumber*** parameter.

Table 12-47
OutpulseNumber nonfatal application errors

Error type	Log generated	Error action performed
Invalid NOA	CAIN100	National NOA is assumed and parameter processing continues.
Maximum number of digits allowed is exceeded	CAIN100	Excess digits are truncated, parameter processing continues.

Associated logs

CAIN100

Restrictions

- CAIN call processing uses existing switch logic to perform outpulsing. Each trunk performs a series of steps to determine the outpulsed address. In some cases (depending on agents and features used), digits are stripped or modified. Therefore, the actual number outpulsed may not match the number received from the SCP. The following examples provide two scenarios:
  - — The SCP provides a 10-digit national ***OutpulseNumber*** and a trunk parameter indicating DAL termination. However, field DIGSOUTP in table TRKGRP indicates that only seven digits should be outpulsed on the terminating DAL. Therefore, only the last seven digits of the ***OutpulseNumber*** are outpulsed.
  - — The SCP provides a 10-digit national ***OutpulseNumber*** and a trunk parameter indicating FGD termination. If the NPA of the ***OutpulseNumber*** matches the connecting NPA (field CONNGNPA in table TRKGRP), then only the last seven digits are outpulsed. Otherwise, the entire 10-digit number in outpulsed.
- Table RTEATTR provides the capability to manipulate parameters in the outgoing IAM. Normally, CAIN allows these changes to take place. However, when the SCP returns an outpulse number, the CAIN framework overrides the datafill in table RTEATTR and outpulses the SCP-supplied digits without any restrictions.

Parameter definition

The ***OverflowBillingIndicator*** parameter provides customized billing information for the UCS DMS-250 CDR.

The CAINTEST command and the VAMP901 log are enhanced to allow a display of the ***OverflowBillingIndicator*** parameter in a response message.

Usage

Optional

Range of values

The range of values consists of 3 binary coded digits (000–999) for each of the two fields in the parameter. The two field names are AMA Call Type and Service Feature Identification.

CDR population

The ***OverflowBillingIndicator*** parameter contains data the switch uses to populate the AMACALLT and SVCFTR CDR fields. The non-zero value for the AMA Call Type field in the parameter populates the AMACALLT CDR field. The non-zero value for the Service Feature Identification field populates the SVCFTR CDR field. The MODMAP field in the CDR is also populated to identify the BAF module to be generated.

Note: If the UCS DMS-250 switch receives the ***OverflowBillingIndicator*** parameter with one field having a non-zero value and the other field having a zero value, the non-zero field's value populates the appropriate field of the CDR.

Application errors

None

Associated logs

None

Restrictions

The UCS DMS-250 switch assumes all BillingIndicator information is correct and does not edit any fields.

If the UCS DMS-250 switch receives an ***OverflowBillingIndicator*** parameter with a value of zero for both fields, the switch assumes the SCP did not have any data provide and ignores the information it receives. The switch also ignores information received in this parameter if the switch does not receive an ***AMAslpID*** parameter in the same TCAP package.

PrimaryBillingIndicator (continued)

Parameter definition

The ***PrimaryBillingIndicator*** parameter provides customized billing information for the UCS DMS-250 CDR.

The CAINTEST command and the VAMP901 log are enhanced to allow a display of the ***PrimaryBillingIndicator*** parameter in a response message.

Usage

Optional

Range or values

The range of values consists of 3 binary coded digits (000–999) for each of the two fields in the parameter. The two field names are AMA Call Type and Service Feature Identification.

CDR population

The ***PrimaryBillingIndicator*** parameter contains data the switch uses to populate the AMACALLT and SVCFTR CDR fields. The non-zero value for the AMA Call Type field in the parameter populates the AMACALLT CDR field. The non-zero value for the Service Feature Identification field populates the SVCFTR CDR field. The MODMAP field in the CDR is also populated to identify the BAF module to be generated.

Note: If the UCS DMS-250 switch receives the ***PrimaryBillingIndicator*** parameter with one field having a non-zero value and the other field having a zero value, the non-zero field's value populates the appropriate field of the CDR.

Application errors

None

Associated logs

None

Restrictions

The UCS DMS-250 switch assumes all BillingIndicator information is correct and does not edit any fields

If the UCS DMS-250 switch receives an ***PrimaryBillingIndicator*** parameter with a value of zero for both fields, the switch assumes the SCP did not have any data provide and ignores the information it receives. The switch also ignores information received in this parameter if the switch does not receive an ***AMAslpID*** parameter in the same TCAP package.

PrimaryTrunkGroup (continued)

Parameter definition

This parameter contains the primary route index for table TERMRTE or TANDMRTE and consists of the terminating switch identifier and a trunk group number for the terminating agent.

Message types

The following messages support the ***PrimaryTrunkGroup*** parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

SWID: 0–999
trunk group number: 0–8191 (field ADNUM from table CLLI)
number to outpulse: 0–1

Outpulsing

When a call is routed using the primary route index, the following outpulse precedence rules apply:

1 Outpulse the ***OutpulseNumber*** when present, and the number to outpulse is set to 0.

2 Outpulse the ***CalledPartyID*** when present, or the current translated address.

3 Normal in-switch outpulsing rules apply.

CDR population

None

Application errors

Table 12-48 shows errors that can occur with regards to the ***PrimaryTrunkGroup*** parameter.

Table 12-48
PrimaryTrunkGroup parameter processing errors

Error type	Log generated	Error action performed
Unknown TERMRTE index (trk number not found in TERMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action
Unknown TANDMRTE index (switchid not found in TANDMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action

Note: The actions performed in response to these errors assume that there are no other routing parameters in the message. The action may be different if there are other routing parameters in the message.

Associated logs

None

Restrictions

None

ResourceType (continued)

Parameter definition

The ***ResourceType*** parameter indicates the type of resource the call requires. CAIN call processing connects the call to the resource.

Message types

The following messages support the ***ResourceType*** parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**
- **`Call_Info_To_Resource`**

Usage

Required

Optional in **`Call_Info_To_Resource`** message

Range of values

PlayAnnouncement: 0
PlayAnnouncementAndCollectDigits: 1
FlexParameterBlock: 4

CDR population

None

Application errors

Table 12-49 shows errors that can occur in regard to the ***ResourceType*** parameter.

Table 12-49
ResourceType parameter application errors

Error	Log generated	Reported to SCP?	Error action performed
ResourceType is not “Play Announcement” in a response package	CAIN200	No	AIND treatment is applied.
ResourceType is “Play Announcement” in a conversation package	CAIN100	Yes	***DisconnectFlag*** is assumed; resource play occurs and AINF treatment is applied. Switch returns an RRL component in a response package

Associated logs

CAIN200, CAIN100

Restrictions

When ***DestinationAddress*** is received, ***ResourceType*** is sent on to the IP without interpretation by the switch.

SecondAlternateCarrier (continued)

Parameter definition

This parameter contains second alternate carrier selection information (CSI) and carrier identification code (CIC) to route the call.

Message types

The following messages support the ***`SecondAlternateCarrier`*** parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Note: When the SCP sends the ***`SecondAlternateCarrier`*** parameter in an **`Analyze_Route`** message, the switch retains the parameter for future use. Refer to Chapter 10 for additional information on the **`Analyze_Route`** message.

Usage

Optional

Range of values

Carrier Selection Field

- NO_INDICATION
- PRESUBSCRIBED_AND_NOT_INPUT
- PRESUBSCRIBED_AND_INPUT
- PRESUBSCRIBED_AND_NO_INDICATION
- NOT_PRESUBSCRIBED_AND_INPUT

Carrier ID field

- 0000–9999

CDR population

If the CIC digits of the ***`AMADigitsDialedWC`*** parameter are not present, CIC and CARSEL CDR fields are populated from this parameter when it is used to route the call.

Application errors

Table 12-50 shows errors that can occur with regards to the ***SecondAlternateCarrier*** parameter.

Table 12-50
SecondAlternateCarrier nonfatal application errors

Error type	Log generated	Error action performed
CIC not datafilled in table CICRTE	CAIN100	The parameter is ignored.

Associated logs

CAIN100

Restrictions

None

SecondAlternateTrunkGroup (continued)

Parameter definition

This parameter contains an additional route index for table TERMRTE or TANDMRTE and consists of the terminating switch identifier and a trunk group number for the terminating agent.

Message types

The following messages support the ***SecondAlternateTrunkGroup*** parameter:

- **Analyze_Route**
- **Originate_Call**

Usage

Optional

Range of values

SWID: 0–999
trunk group number: 0–8191 (field ADNUM from table CLLI)
number to outpulse: 0–1

Outpulsing

When a call is routed using the second alternate route index, the following outpulse precedence rules apply:

1. Outpulse the ***GenericAddressList***'s SecondAlternateOutpulseNumber when available, and the number to outpulse is set to 0.
2. Outpulse the ***OutpulseNumber*** when present, and the number to outpulse is set to 0.
3. Outpulse the ***CalledPartyID*** when present, or the current translated address.
4. Normal in-switch outpulsing rules apply.

CDR population

None

Application errors

Table 12-51 shows errors that can occur with regards to the ***SecondAlternateTrunkGroup*** parameter.

Table 12-51
SecondAlternateTrunkGroup parameter processing errors

Error type	Log generated	Error action performed
Unknown TERMRTE index (trk number not found in TERMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action
Unknown TANDMRTE index (switchid not found in TANDMRTE)	CAIN300	error action defined in the trigger table or AINF if no defined action

Note: The actions performed in response to these errors assume that there are no other routing parameters in the message (the action may be different if there are other routing parameters in the message).

Associated logs

CAIN300

Restrictions

None

Parameter definition

This parameter contains data required to perform the action specified by the ***ResourceType*** parameter. The ***ResourceType*** tells the switch how to decode the ***StrParameterBlock*** portion of the **Send_To_Resource** or **Connect_To_Resource** message, and indicates to the **Call_Info_To_Resource** message which resource type to use to connect. CAIN call processing uses the ***StrParameterBlock*** in these messages to provide information that the switch or IP requires to perform the function specified by the ***ResourceType*** parameter.

Table 12-52 lists the ***StrParameterBlock*** encoding choices.

Table 12-52
STRParameterBlock encoding choices

STRParameterBlock choice	Definition
AnnouncementBlock	This choice encodes as an UninterAnnounceBlock. The UninterAnnounceBlock contains a Play Announcement tag and up to 3 uninterruptible announcement elements. The announcement elements contain a resource identifier to index into table CAINRSRC. ***Note 1:*** Only one announcement block is expected. ***Note 2:*** Any dialed digits are ignored.

—continued—

Table 12-52
STRParameterBlock encoding choices (continued)

STRParameterBlock choice	Definition
AnnouncementDigitBlock	This choice contains the following information: • MaximumDigits – identifies the number of digits required — Fixed indicates that the switch should collect the specified number of digits (0–24). — Variable indicates that the switch should collect the number of digits in the range (0–24). — Variable “upto” indicates that the switch should collects the number of digits specified in the range (0–24) provided by the SCP. — Normal number of digits indicates that the switch should collect an address using the normal dialing plan for the agent. • UninterAnnounceBlock – contains a Play Announcement tag and up to 3 uninterruptible, audible tone or announcement resource identifiers. The announcement elements contain a resource identifier to index into table CAINRSRC. • InterAnnounceBlock – contains a Play Announcement tag and up to 3 interruptible, audible tone or announcement resource identifiers. The announcement elements contain a resource identifier to index into table CAINRSRC.

—continued—

StrParameterBlock (continued)

Table 12-52
STRParameterBlock encoding choices (continued)

STRParameterBlock choice	Definition
FlexParameterBlock	This choice contains one of the following elements: • For 1129-Style IP interactions: 1–95 bytes • For in-switch virtual IP (VIP) interactions: — Resource encoding authority, which defines how to interpret the subsequent data within the FlexParameterBlock choice — Flextag set to VIP — Up to 5 collectibles: – ANNC, ADDR, AUTH, CARD, ACCT, PIN, UNKNOWN – Resource information, such as play a tone or announcement, or play a tone or announcement and collect digits; type of tone or announcement to play, and whether tones and announcements are interruptible – Optionally, minimum and maximum digits to collect (0 to 24), used for fast interdigital timing, and to determine end of dialing for a collectible when multiple collectibles are entered without interruption (for all but ANNC). – Timer – 0–15, identifies the permanent signal timer value for the initial digit. The trunk group partial dial timer controls the interdigital timing for the remaining digits (for all but ANNC).
—end—	

Message types

The following messages support the ***StrParameterBlock*** parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**
- **`Call_Info_To_Resource`**

When a ***DestinationAddress*** parameter is received, the ***StrParameterBlock*** parameter is sent on to the IP without interpretation by the switch.

Usage

Required

Optional in **`Call_Info_To_Resource`** message

Range of values

AnnouncementID is an index (0–4095) into table CAINRSRC

AnnouncementDigitBlock:

MaximumDigits values:

— 0–24 is “fixed number of digits”

— 128–152 is “up to max number of digits”

Announcement elements

The announcement element contains an AnnouncementID and information digits. The information digits are ignored.

CDR population

None

Application errors

Table 12-53 lists the nonfatal application errors that can occur in regard to the ***`StrParameterBlock`*** parameter.

Table 12-53
StrParameterBlock nonfatal application errors

Error type	Log generated	Error action performed
Erroneous data value	CAIN200	**`Resource_Clear`** or **`CTR_Clear`** is sent with a value of `failure` and ***`FailureCause`*** of **`Application_Error`**
Erroneous data value	CAIN100	Change maximum to 24

Associated logs

CAIN100, CAIN200

StrParameterBlock (continued)

Associated OMs

ANN

Restrictions

- Information digits are ignored.

ATTENTION
Due to UCS DMS-250 switch size limitations for internal messaging, the UCS DMS-250 switch does not support the maximum size ***`STRParameterBlock`*** as specified in Bellcore's *GR-1129-CORE* and *GR-1299-CORE* documents. If the internal limit is exceeded, the switch sends a **`Resource_Clear`** message to the SCP in a conversation package with a ***`ClearCause`*** parameter value of `Abort`.

StrParameterBlock (end)

Table 12-54 identifies size limits by message and trunk type.

Table 12-54
STRParameterBlock (FlexParameterBlock choice) size limits

Message	Parameter breakdown	IP trunk type	
		PRI (Note)	ISUP
`Send_To_Resource`	ParmID, tag, and length bytes overhead (for both STRParameterBlock and FlexParameter)	5	5
	Payload of FlexParameterBlock	86	86
	Total	91	91
`Call_Info_To_Resource`	ParmID, tag, and length bytes overhead (for both STRParameterBlock and FlexParameter)	5	5
	Payload of FlexParameterBlock	86	111
	Total	91	116
Note: Requires XPM load ELI81AZ or later.			

Parameter definition

The ***TimeoutTimer*** parameter indicates the value, in minutes, of the switch terminating timeout timer. The switch uses this value to determine when the *Timeout* event is encountered.

Message types

The following message supports the ***TimeoutTimer*** parameter:

- **Request_Report_BCM_Event**

Usage

Optional

Range of values

1–180

CDR population

None

Application errors

Table 12-55 shows errors that can occur with regards to the ***TimeoutTimer*** parameter.

Table 12-55
TimeoutTimer parameter nonfatal application errors

Error	Log generated	Reported to SCP?	Error action performed
TimeoutTimer parameter is received, but the *Timeout* event is not requested in the **Request_Report_BCM_ Event**.	CAIN100	No	The parameter is ignored.

Associated logs

CAIN100

Restrictions

None

TranslationType (end)

Parameter definition

This parameter specifies the translation type of an ACG control.

Message types

The following message supports the ***`TranslationType`*** parameter:

- **`ACG`**

Usage

Required

Range of values

an integer from 0–255

CDR Population rules

None

Application errors

None

Associated logs

None

Restrictions

None

Extension parameter definition

The `accountCode` extension parameter contains the account code collected from in-switch translations or collected by the *O_Feature_Requested* trigger. This number may or may not be validated in-switch.

Message types

The following TDP-Request messages support the `accountCode` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

2–12 digits when collected in-switch
Up to 24 digits when collected using *O_Feature_Requested* feature processor

Supported values

The `accountCode` extension parameter should be encoded as AINDigits, with an NOA of UNKNOWN and a Numbering plan (NumPlan) of ISDN.

CDR population

The digits contained in this extension parameter populate the ACCTCD field of the CDR.

Note: If any ***`AMADigitsDialedWC`*** parameters are received later in the call that are to populate the same CDR field (ACCTCD), the ***`AMADigitsDialedWC`*** parameters have precedence over the digits sent in this extension parameter.

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They are not used to check for subscription.

ExtensionParameter alternateTrunkGroupSTS (continued)

Extension parameter definition

This extension parameter contains the serving translation scheme (STS) value associated with the returned ***AlternateTrunkGroup*** parameter. The STS returned by the SCP (or datafilled in table CAINXDFT) replaces the STS defined at the time of the query.

Refer to Chapter 10, "Incoming CAIN messages," for more information on how the `alternateTrunkGroupSTS` extension parameter is used in routing.

Figure 12-1 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter.

Figure 12-1
Precedence order for STS extension parameters

STS Extension Parameters / Route Parameters Returned by the SCP						table CAINXDFT					Pre Query STS
	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
`OverflowRoutingNo`				3	1				4	2	5

LEGEND

`PRISTS (primaryTrunkGroupSTS)`	– STS to be used with the ***PrimaryTrunkGroup*** parameter
`ALTSTS (alternateTrunkGroupSTS)`	– STS to be used with the ***AlternateTrunkGroup*** parameter
`SALTSTS (secondAlternateTrunkGroupSTS)`	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
`STS (servTranslationScheme)`	– STS to be used with the ***CalledPartyID*** parameter
`OVFLSTS (overflowRoutingNo)`	– STS to be used with the `OverflowRoutingNo` parameter

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

Message types

The following messages support the `alternateTrunkGroupSTS` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

0–999

CDR population

Table 12-56 shows the fields populated in the CDR when the ***`AlternateTrunkGroup`*** parameter is used, and the `alternateTrunkGroupSTS` extension parameter is received.

Table 12-56
alternateTrunkGroupSTS CDR population

CDR field	Population
OPART	This field contains the originating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
TPART	This field contains the terminating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

ExtensionParameter alternateTrunkGroupSTS (continued)

Application errors

Table 12-57 lists the application errors that can occur in regard to the `alternateTrunkGroupSTS` extension parameter.

Table 12-57
alternateTrunkGroupSTS application errors

Error type	Log generated	Reported to SCP?	Error action performed
STS not datafilled in table HNPACONT	CAIN100	No	Ignore the STS, and use the default from table CAINPARM when available. If unavailable, use the STS available before the query.
STS not datafilled in table STS2ACDB, STS2FTDB, STS2PXDB, STS2CTDB	OCC203	No	Call sent to treatment. ***Note:*** This error only applies to Global-IMTs.

Associated logs

CAIN100, OCC203

Restrictions

None

Datafill requirements

Provision a default alternateTrunkGroupSTS

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default alternateTrunkGroupSTS value using the following format:

>ADD caingrp ALTSTS sts_value
where

caingrp	is the CAIN group requiring default STS values.
sts_value	is the default STS value.

Sample entry: **>ADD caingrp1 altsts 761 $**

ExtensionParameter alternateTrunkGroupSTS (end)

A default alternateTrunkGroupSTS has been provisioned**.**

ExtensionParameter
amaDigits (continued)

Extension parameter definition

This extension parameter contains digit strings to be populated into one of the following CDR fields: PINDIGS, ACCTCD, BILLNUM, CIC, ORIGPVN, or TERMPVN. Up to six `amaDigits` can be sent (in sequence) within one response message.

Message types

The following message supports the `amaDigits` parameter:

- **`Call_Info_To_Resource`**

Usage

Optional

Range of values

3 to 27 digits

Note: The first 3 digits identify the CDR field and the remaining digits are populated into the CDR.

CDR population

The handling of the `amaDigits` extension parameter is controlled by the CAIN_PROTOCOL_VERSION parameter in table CAINPARM. Table 12-58 shows the possible fields that are populated in the CDR, regardless of the CAIN_PROTOCOL_VERSION setting, when the `amaDigits` extension parameter is received.

Table 12-58
amaDigits CDR population regardless of the CAIN_PROTOCOL_VERSION

CDR field	Number of digits in CDR	Digits code	Population
PINDIGS	4	301	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber’s PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.
CIC	4	302	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.
ORIGPVN	15	303	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	304	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

ExtensionParameter
amaDigits (continued)

Table 12-59 shows the possible fields that are populated in the CDR if the CAIN_PROTOCOL_VERSION is set to V2 or lower when the `amaDigits` extension parameter is received.

Table 12-59
amaDigits CDR population
when CAIN_PROTOCOL_VERSION is V2 or lower

CDR field	Number of digits in CDR	Digits code	Population
PINDIGS	4	001	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber’s PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.
ACCTCD	12	002	ACCOUNT CODE. This field is populated with the account code digits collected for the call.
BILLNUM	23	003	BILLING NUMBER. This field is populated with one of the following and identifies the billing number for a call: • Travel card number • Authcode • N00 called number (NXX-NXX-XXXX). • SCP-identified number
CIC	4	004	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—continued—

Table 12-59
amaDigits CDR population
when CAIN_PROTOCOL_VERSION is V2 or lower (continued)

CDR field	Number of digits in CDR	Digits code	Population
ORIGPVN	15	005	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	006	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.

Note 1: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 2: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—end—

ExtensionParameter
amaDigits (continued)

Table 12-60 shows the possible fields that are populated in the CDR if the CAIN_PROTOCOL_VERSION is set to V3 when the `amaDigits` extension parameter is received.

Table 12-60
amaDigits CDR population
when CAIN_PROTOCOL_VERSION is V3 or higher

CDR field	Number of digits in CDR	Digits code	Population
BILLNUM	23	001	BILLING NUMBER. This field is populated with one of the following and identifies the billing number for a call: • Travel card number • Authcode • N00 called number (NXX-NXX-XXXX). • SCP-identified number
ACCTCD	12	002	ACCOUNT CODE. This field is populated with the account code digits collected for the call.
Note 1		003	None
Note 1		004	None
Note 1		005	None
Note 1		006	None
Note 1		011	None
Note 1		012	None
PINDIGS	4	301	PERSONAL IDENTIFICATION NUMBER DIGITS. This field contains the subscriber's PINDIGS. An optional level of screening beyond authorization code and automatic number identification can be provided by this field.

Note 1: CAIN call processing allows these digit codes but does not act upon or populate any CDR field with them.
Note 2: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 3: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—continued—

Table 12-60
amaDigits CDR population
when CAIN_PROTOCOL_VERSION is V3 or higher (continued)

CDR field	Number of digits in CDR	Digits code	Population
CIC	4	302	CARRIER IDENTIFICATION CODE. This field identifies the long distance carrier for the call. A three digit CIC is prefixed with a 0. For example, a CIC of 233 is populated as 0233.
ORIGPVN	15	303	ORIGINATING PRIVATE NUMBER This field contains the customer-defined dialing plan number assigned to the station or the user originating the VPN call. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.
TERMPVN	15	304	TERMINATING PRIVATE NUMBER. This field contains the customer-defined dialing plan number assigned to the called party. This field is populated from a CAIN response received from the SCP in the `amaDigits` extension parameter. Non-CAIN calls do not populate this field.
Note 1		999	None

Note 1: CAIN call processing allows these digit codes but does not act upon or populate any CDR field with them.
Note 2: If the SCP-returned digits exceed the maximum, the additional digits are truncated.
Note 3: For more information on CDR fields, refer to *UCS DMS-250 Billing Records Application Guide*.

—end—

Application errors

None

Associated logs

None

ExtensionParameter
amaDigits (end)

Restrictions

When there are multiple `amaDigits` extension parameters having the same CDR field associated with them, the last one processed takes precedence in populating the CDR field.

Datafill requirements

A default value cannot be datafilled for the `amaDigits` extension parameter.

Extension parameter definition

The SCP populates the `billingNumber` extension parameter with a non-standard charge number (for example, CARD, AUTH, ACCT, PIN, or N00).

CDR population

The digits contained in this extension parameter populate the BILLNUM field of the CDR.

Note: If any ***AMADigitsDialedWC*** parameters are received later in the call that are to populate the same CDR field (BILLNUM), the ***AMADigitsDialedWC*** parameters have precedence over the digits sent in this extension parameter.

The SCP populates the `billingNumber` parameter with a non-standard charge number (for example, CARD, AUTH, ACCT, PIN, and N00).

Message types

The following messages support the `billingNumber` extension parameter:

- **Analyze_Route**
- **Originate_Call**

Note: When the SCP sends the `billingNumber` extension parameter in an **Analyze_Route** message, the switch retains the parameter for future use. Refer to Chapter 10 for additional information on the **Analyze_Route** message.

Usage

Optional

Range of values

Maximum of 24 digits

Supported values

The `billingNumber` extension parameter should be encoded as AINDigits. The NOA and Numbering Plan (NumPlan) vary based on the type of the billing number, as given in Table 12-61.

ExtensionParameter
billingNumber (end)

Table 12-61
Supported billing numbers and NOAs for the billingNumber extension parameter

Type of Billing Number	NOA	NumPlan
Calling Card, MCCS number	CARD	PRVT
Authcode	AUTH	PRVT
Account code	ACCT	PRVT
Personal Identification digits	PIN	PRVT
800/888 services	N00	PRVT
ANI	ANI	PRVT

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They will not be used to check for subscription.

Extension parameter definition

This extension parameter contains 32 bits of SCP-defined billing information that is stored in the CDR.

Message types

The following messages support the `billSequenceNumber` extension parameter:

- **`Analyze_Route`**
- **`Continue`**
- **`Disconnect`**
- **`Authorize_Termination`**
- **`Originate_Call`**

Usage

Optional

Range of values

32 bits of data

CDR population

Table 12-62 shows the fields populated in the CDR when the `billSequenceNumber` extension parameter is received.

Table 12-62
billSequenceNumber CDR population

CDR field	Population
SCPBILL	This field contains 32 bits of billing information provided by the SCP.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

None

ExtensionParameter billSequenceNumber (end)

Associated logs

None

Restrictions

None

Datafill requirements

A default value cannot be datafilled for the `billSequenceNumber` extension parameter.

ExtensionParameter cainGroup

ATTENTION

The `cainGroup` extension parameter requires the CAIN0601 SOC option. Refer to Volume 5, Chapter 5, "NetworkBuilder SOC functionality," for more information.

Extension parameter definition

This extension parameter contains the CAIN group identified by the SCP for SCP CAINGRP subscription.

The switch generates a CAIN902 log when this extension parameter is returned, indicating that SCP returned subscription has changed.

Message types

The following messages support the `cainGroup` extension parameter:

- **`Analyze_Route`**
- **`Collect_Information`**

Usage

Optional

Range of values

0 to 4095 (group number datafilled in table CAINGRP)

CDR population

None

Application errors

Table 12-63 lists the nonfatal application errors that can occur in regard to the `cainGroup` extension parameter.

ExtensionParameter
cainGroup (end)

Table 12-63
cainGroup nonfatal application errors

Error type	Log generated	Reported to SCP?	Error action performed
SOC for extension parameters (CAIN0200) is enabled; SOC for `cainGroup` (CAIN0601) is idle	CAIN102	No	Extension parameter is ignored.
SOC for extension parameters (CAIN0200) is idle	CAIN102	No	Extension parameter is ignored.
Invalid `cainGroup` returned	CAIN100	No	Extension parameter is ignored.

Associated logs

CAIN100, CAIN102, CAIN902

Restrictions

The SCP-returned value is used to access data required from table CAINGRP.

If the call is reoriginated, the new CAIN group applies to the new call.

Datafill requirements

The SCP-returned value must be defined in table CAINGRP.

ExtensionParameter callBranding

Extension parameter definition

This extension parameter, when received, provides an announcement or tone to be played prior to routing. The parameter contains an index into table CAINRSRC which identifies the appropriate announcement or tone.

Note: Refer to the *Digital Recorded Announcement Machine DRAM and EDRAM Guide* for information regarding announcements, and to the *UCS DMS-250 Data Schema Reference Manual* for information regarding tones (table TONES).

Call branding is performed before termination. Therefore, branding is always attempted when the extension parameter is received. Branding can be played on calls that are sent to treatment or routed to a busy trunk.

Message types

The following message supports the `callBranding` extension parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

0–4095

CDR population

None

Application errors

Table 12-64 lists the application errors that can occur in regard to the `callBranding` extension parameter.

ExtensionParameter
callBranding (continued)

Table 12-64
callBranding application errors

Error type	Log generated	Reported to SCP?	Error action performed
Resource identifier out of range	CAIN200	No	Branding is not performed; call is routed as directed by the **`Analyze_Route`** message received from the SCP.
Resource identifier not datafilled	None	No	Branding is not performed; call is routed as directed by the **`Analyze_Route`** message received from the SCP.
Resource not available	None	No	Branding is not performed; call is routed as directed by the **`Analyze_Route`** message received from the SCP.
Unexpected event or message	CAIN200	No	Branding is not performed; call is routed as directed by the **`Analyze_Route`** message received from the SCP.

Associated logs

CAIN200

Restrictions

None

Datafill requirements

Provision a default callBranding parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default callBranding parameter using the following format:

>ADD caingrp CALLBRND cain_rsrc
where

caingrp	is the CAIN group requiring a default call type.
cain_rsrc	is the index into table CAINRSRC, which identifies the tone or announcement to be played (0–4095).

Sample entry: **>ADD caingrp1 CALLBRND 9**

A default call branding parameter has been provisioned.

ExtensionParameter
callCtrl (end)

Extension parameter definition

This extension parameter enhances control over TDP evaluation for the call in progress.

Message types

The following messages support the `callCtrl` extension parameter:

- **`Analyze_Route`**
- **`Continue`**

Usage

Optional

Range of values

Possible values for the `callCtrl` extension parameter:

- 00000000 – Unrestricted
- 00000001 – Leave the current TDP (LEAVE_TDP)
- 00000010 – Continue, do not trigger (CONT_NOTRIG)
- 00000011–11111111 – Reserved

CDR population

None

Application errors

None

Associated logs

None

Restrictions

The `callCtrl` extension parameter is ignored when received in conversation.

Datafill requirements

Table CAINXDFT does not provision a default for `callCtrl`.

ExtensionParameter callType

Extension parameter definition

This extension parameter contains the network call type. This parameter is populated into the CDR, but has no impact on call processing.

Message types

The following message supports the `callType` extension parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

ONNET, OFFNET, FORCED_ONNET, VIRTUAL_ONNET

CDR population

Table 12-65 shows the fields populated in the CDR when the `callType` extension parameter is received.

Table 12-65
callType CDR population

CDR field	Population
CAINCT	This field contains the CAIN call type received from the SCP response within the `callType` extension parameter. 0 =OFFNET (default) 1 = ONNET 2 = FORCED_ONNET 3 = VIRTUAL_ONNET 4–5 = Reserved 6 –7 = Reserved for SCP-determined values
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

ExtensionParameter
callType (end)

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default callType

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default call type using the following format:

>ADD caingrp CALLTYPE calltype
where

caingrp	is the CAIN group requiring a default call type.
calltype	is the call type (OFFNET, ONNET, FORCED_ONNET, VIRTUAL_ONNET).

Sample entry: **>ADD caingrp2 CALLTYPE onnet**

A default call type has been provisioned.

Extension parameter definition

This extension parameter, when present, indicates that CAIN call processing should be terminated and call control should be passed to the Programmable Service Node Control Unit (SCU). Refer to the *UCS DMS-250 Programmable Service Node Application Guide* for more information.

Message types

The following message supports the `connectToSCU` extension parameter:

- **`Continue`** from an **`Info_Analyzed`** query

Usage

Optional

Range of values

Presence or absence

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Table CAINXDFT does not provision a default for `connectToSCU`.

ExtensionParameter classOfSvc

Extension parameter definition

This extension parameter provides an index into table MULTICOS for class of service (COS) screening. When this parameter is returned (or provisioned in table CAINXDFT), the UCS DMS-250 switch performs COS screening.

Note: The UCS DMS-250 switch can perform COS screening before querying the SCP.

The following types of screening are provided:

- Call type – capability to allow/disallow calls based on the call type (international, OFFNET, or ONNET). If the ***`CalledPartyID`*** parameter is provided, the new call type (set by NOA) is used for screening.
- Destination number – capability to allow/disallow calls based upon the destination number. If the ***`CalledPartyID`*** parameter is provided, the new number is used for screening.
- Time of day – capability to allow/disallow calls based on the time of day. CAIN does not affect this type of screening.

If screening blocks the call, the appropriate COS treatment is applied. Otherwise, CAIN continues to process the **`Analyze_Route`** message and performs the appropriate routing.

Interaction

The parameter value indexes table MULTICOS for class of service screening. Refer to *UCS DMS-250 CSP Translations Reference Manual* for more information.

When COS screening fails, the action specified by COS screening is performed and the **`Analyze_Route`** routing parameters are discarded. In this case, the route index determined through in-switch translations is used to route the call.

When successful COS screening occurs, the CAIN framework continues to process the **`Analyze_Route`** message in order to identify a route index.

Message types

The following message supports the `classOfSvc` extension parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

0–2047

CDR population

Table 12-66 shows the fields populated in the CDR when the `classOfSvc` extension parameter is received.

Table 12-66
classOfSvc CDR population

CDR field	Population
COSINDEX	CLASS OF SERVICE INDEX. This field identifies the index into table MULTICOS that was used to perform class of service screening. This field is empty when COS screening is successful after processing all the indexes provided in table MULTICOS. When screening fails, the index responsible for the failure is captured.
MLTCOSID	MULTI-CLASS OF SERVICE IDENTIFIER. This field contains the index, within table MULTICOS, used to perform multiple COS screening.

Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.

Application errors

None

Associated logs

None

Restrictions

- Class of service override is not supported.
- When the index is not datafilled in table MULTICOS, COSX treatment is applied; logs OCC222 and TRK138 are generated.
- A COS value of zero is ignored.

ExtensionParameter
classOfSvc (end)

Datafill requirements

Provision a default classOfSvc parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default class of service using the following format:

>ADD caingrp COS multicos_index
where

caingrp	is the CAIN group requiring a default call type.
multicos_index	is the index into table MULTICOS, which defines COS screening (0–2047).

Sample entry: **>ADD caingrp1 COS 5**

A default class of service parameter has been provisioned.

ExtensionParameter edpBuffer

Extension parameter definition

This extension parameter indicates, by its presence, that digits are to be buffered for conversation when an EDP-Request message is sent to the SCP. This extension parameter is the EDP equivalent to the BUFFER option the trigger tables provide for triggers.

Message types

The following message supports the `edpBuffer` extension parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

Presence or absence

CDR population

None

Application errors

None

Associated logs

None

Restrictions

Digit buffering is not supported for the ***O_No_Answer*** EDP.

Datafill requirements

Provision a default edpBuffer parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single CAIN group. Each CAIN group can have one set of defaults.

2 Provision EDP buffering using the following format:

>ADD caingrp EDPBUFFR
where

caingrp	is the CAIN group requiring EDP buffering.

ExtensionParameter
edpBuffer (end)

Sample entry: **>ADD caingrp1 edpbuffr**

EDP buffering is provisioned.

ExtensionParameter
netinfo

Extension parameter definition

This extension parameter contains the new business group information to be sent in the outgoing IAM for multiswitch business group (MBG) calls.

Note: The MBG calls discussed here refer only to calls received from a Nortel Networks SL-100 switch.

Message types

The following messages support the `netinfo` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

PS (Party Selector):
0000–1111
LPII (Line Privileges Information Indicator):
0–1
BGID (Business Group Identifier):
0–1
Attst (Attendant Status):
0–1
EXTNETID (External Network ID):
0000000000000000–1111111111111111
NETCGID (Network Customer Group ID):
0000000000000000–1111111111111111
NCOS (Network Class of Service):
0000000000000000–1111111111111111

CDR population

None

Application errors

None

Associated logs

None

ExtensionParameter
netinfo (end)

Restrictions

None

Datafill requirements

Provision a default netinfo extension parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default netinfo extension parameter value using the following format:

>ADD caingrp netinfo extnetid_value
where

caingrp	is the CAIN group requiring default netinfo values.
extnetid_value	is the default external network ID value.
netcgid_value	is the default network customer ID value.
ncos_value	is the default network class of service value.

Sample entry: **>ADD caingrp1 netinfo 5 100 25 $**

A default netinfo extension parameter has been provisioned

Extension parameter definition

This extension parameter specifies the action to take based on the routing criteria for the ***Network_Busy*** EDP.

Message types

The following message supports the `networkBusyActions` extension parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

The `networkBusyActions` extension parameter contains three fields: RTEAVAIL (route available), RTESDONE (routes done), and TERMRTE_GNCT (terminating route generalized no circuit).

The value of the first field, RTEAVAIL, is the ***Network_Busy*** EDP equivalent to the RTEAVAIL trigger criteria in table NETBUSY for the *Network_Busy* trigger, which specifies the action to take if additional routes are available when the ***Network_Busy*** EDP is encountered. The possible values for RTEAVAIL are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***Network_Busy*** EDP.
- NEXTRTE – indicates the next route choice should be used.

The second field, RTESDONE, is the ***Network_Busy*** EDP equivalent to the RTESDONE trigger criteria in table NETBUSY for the *Network_Busy* trigger, which specifies the action to take if there are no additional routes are available when the ***Network_Busy*** EDP is encountered. The possible values for RTESDONE are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***Network_Busy*** EDP.

ExtensionParameter networkBusyActions (continued)

The third field, TERMRTE_GNCT, is the ***Network_Busy*** EDP equivalent to the TERMRTE_GNCT condition for the *Network_Busy* trigger, which specifies the action to take when the route choice determined by direct termination routing through table TERMRTE is busy. The possible values for TERMRTE_GNCT are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***Network_Busy*** EDP.
- NEXTRTE – indicates the next in-switch route choice should be used.
- NEXTCNRTE – indicates the next CAIN route choice should be used.

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default networkBusyActions parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single CAIN group. Each CAIN group can have one set of defaults.

2 Add a default networkBusyActions parameter using the following format:

>ADD caingrp NETBACT rteavail_action rtesdone_action termrtegnct_action
where

caingrp	is the CAIN group requiring a default call type.
rteavail_action	is the ***Network_Busy*** EDP action to take when an additional route(s) is available for the call.

ExtensionParameter networkBusyActions (end)

rtesdone_action is the ***Network_Busy*** EDP action to take when all possible routes have been attempted for the call.

termrtegnct_action is the ***Network_Busy*** EDP action to take when the route choice determined by direct termination routing through table TERMRTE is busy.

Sample entry: **>ADD caingrp1 NETBACT REQUEST IGNORE REQUEST**

A default networkBusyActions parameter has been provisioned.

ExtensionParameter oCalledPartyBusyActions

Extension parameter definition

This extension parameter specifies the action to take based on the routing criteria for the ***O_Called_Party_Busy*** EDP.

Message types

The following message supports the `oCalledPartyBusyActions` extension parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

The `oCalledPartyBusyActions` extension parameter contains two fields: RTEAVAIL (route available), and RTESDONE (routes done).

The value of the first field, RTEAVAIL, is the ***O_Called_Party_Busy*** EDP equivalent to the RTEAVAIL trigger criteria in table OCLDBUSY for the *O_Called_Party_Busy* trigger, which specifies the action to take if additional routes are available when the ***O_Called_Party_Busy*** EDP is encountered. The possible values for RTEAVAIL are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***O_Called_Party_Busy*** EDP.
- NEXTRTE – indicates the next in-switch route choice should be used.
- NEXTCNRTE – indicates the next CAIN route choice should be used.

The second field, RTESDONE, is the ***O_Called_Party_Busy*** EDP equivalent to the RTESDONE trigger criteria in table OCLDBUSY for the *O_Called_Party_Busy* trigger, which specifies the action to take if no additional routes are available when the ***O_Called_Party_Busy*** EDP is encountered. The possible values for RTESDONE are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***O_Called_Party_Busy*** EDP.

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default oCalledPartyBusyActions parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single CAIN group. Each CAIN group can have one set of defaults.

2 Add a default oCalledPartyBusyActions parameter using the following format:

>ADD caingrp OCLDBACT rteavail_action rtesdone_action
where

caingrp is the CAIN group requiring a default call type.
rteavail_action is the ***O_Called_Party_Busy*** EDP action to take when an additional route(s) is available for the call.
rtesdone_action is the ***O_Called_Party_Busy*** EDP action to take when all possible routes have been attempted for the call.

Sample entry: **>ADD caingrp1 OCLDBACT IGNORE REQUEST**

A default oCalledPartyBusyActions parameter has been provisioned.

Extension parameter definition

This extension parameter specifies the action to take based on the routing criteria for the ***O_No_Answer*** EDP.

Message types

The following message supports the `oNoAnswerActions` extension parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

The `oNoAnswerActions` extension parameter contains two fields: RTEAVAIL (route available), and RTESDONE (routes done).

The value of the first field, RTEAVAIL, is the ***O_No_Answer*** EDP equivalent to the RTEAVAIL trigger criteria in table ONOANSWR for the *O_No_Answer* trigger, which specifies the action to take if additional routes are available when the ***O_No_Answer*** EDP is encountered. The possible values for RTEAVAIL are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***O_No_Answer*** EDP.
- NEXTRTE – indicates the next in-switch route choice should be used.
- NEXTCNRTE – indicates the next CAIN route choice should be used.

The second field, RTESDONE, is the ***O_No_Answer*** EDP equivalent to the RTESDONE trigger criteria in table ONOANSWR for the *O_No_Answer* trigger, which specifies the action to take if no additional routes are available when the ***O_No_Answer*** EDP is encountered. The possible values for RTESDONE are:

- REQUEST – indicates the switch should send an EDP-Request message to the SCP.
- IGNORE – indicates no action should be taken at the ***O_No_Answer*** EDP.

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default oNoAnswerActions parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single CAIN group. Each CAIN group can have one set of defaults.

2 Add a default oNoAnswerActions parameter using the following format:

>ADD caingrp ONOANACT rteavail_action rtesdone_action
where

caingrp	is the CAIN group requiring a default call type.
rteavail_action	is the ***O_No_Answer*** EDP action to take when an additional route(s) is available for the call.
rtesdone_action	is the ***O_No_Answer*** EDP action to take when all possible routes have been attempted for the call.

Sample entry: **>ADD caingrp1 ONOANACT REQUEST REQUEST**

A default oNoAnswerActions parameter has been provisioned.

ExtensionParameter overflowRoutingNoSTS (continued)

Extension parameter definition

This extension parameter contains the STS value associated with the ***GenericAddressList*** parameter's returned `OverflowRoutingNo`. The STS returned by the SCP (or datafilled in table CAINXDFT) replaces the STS defined at the time of the query.

Refer to Chapter 10, "Incoming messages," for more information on how the `overflowRoutingNoSTS` extension parameter is used in routing.

Figure 12-2 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter.

Figure 12-2
Precedence order for STS extension parameters

STS Extension Parameters / Route Parameters Returned by the SCP	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	table CAINXDFT					Pre Query STS
						PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
`OverflowRoutingNo`				3	1				4	2	5

LEGEND

`PRISTS (primaryTrunkGroupSTS)`	– STS to be used with the ***PrimaryTrunkGroup*** parameter
`ALTSTS (alternateTrunkGroupSTS)`	– STS to be used with the ***AlternateTrunkGroup*** parameter
`SALTSTS (secondAlternateTrunkGroupSTS)`	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
`STS (servTranslationScheme)`	– STS to be used with the ***CalledPartyID*** parameter
`OVFLSTS (overflowRoutingNo)`	– STS to be used with the `OverflowRoutingNo` parameter

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

Message types

The following messages support the `overflowRoutingNoSTS` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

0–999

CDR population

Table 12-67 shows the fields populated in the CDR when the `overflowRoutingNoSTS` extension parameter is used in conjunction with with the ***`GenericAddressList`*** parameter's returned `OverflowRoutingNo` to route the call.

Table 12-67
overflowRoutingNoSTS CDR population

CDR field	Population
OPART	This field contains the originating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
TPART	This field contains the terminating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-68 lists the application errors that can occur when an `overflowRoutingNoSTS` extension parameter is received.

ExtensionParameter overflowRoutingNoSTS (end)

Table 12-68
overflowRoutingNoSTS application errors

Error type	Log generated	Reported to SCP?	Error action performed
STS not datafilled in table HNPACONT	CAIN100	No	Ignore the STS, and use the default from table CAINPARM when available. If unavailable, use the STS available before the query.
STS not datafilled in table STS2ACDB, STS2FTDB, STS2PXDB, STS2CTDB	OCC203	No	The call is sent to treatment. ***Note:*** This error applies only to Global IMTs.

Associated logs

CAIN100, OCC203

Restrictions

None

Datafill requirements

Provision a default overflowRoutingNoSTS parameter

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single CAIN group. Each CAIN group can have one set of defaults.

2 Add a default overflowRoutingNoSTS parameter value using the following format:

>ADD caingrp OVFLSTS sts_value
where

caingrp	is the CAIN group requiring default OVFLSTS values.
sts_value	is the default STS value.

Sample entry: **>ADD caingrp1 ovflsts 761 $**

A default overflowRoutingNoSTS parameter has been provisioned**.**

Extension parameter definition

The `pinDigits` extension parameter holds the collected PIN code, when available, or a PIN collected at *O_Feature_Requested*. It also contains digits collected during conversational digit collection.

Message types

The following messages support the `pinDigits` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

Maximum of 24 digits

Supported values

The `pinDigits` extension parameter should be encoded as AINDigits, with an NOA of UNKNOWN and a Numbering Plan (NumPlan) of ISDN.

CDR population

The digits contained in this extension parameter populate the PINDIGS field of the CDR.

Note: If any ***`AMADigitsDialedWC`*** parameters are received later in the call that are to populate the same CDR field (PINDIGS), the ***`AMADigitsDialedWC`*** parameters have precedence over the digits sent in this extension parameter.

Restrictions

The digits used to populate this extension parameter may be used by call processing later in the call for future queries. They are not used to check for subscription.

ExtensionParameter pretranslatorName (end)

Extension parameter definition

This extension parameter contains an index into table CNPREXLA. Table CNPREXLA provides a new pretranslator to be used by the switch, through table STDPRTCT, in translations for the rest of the call including reorigination.

This extension parameter helps minimize ambiguous dialing. Using the pretranslator name to resolve ambiguous dialing applies only when the address is the first collectible requested by the SCP. Any misdialing errors or other dialing ambiguities continue to be handled by the SCP.

Message types

The following messages support the `pretranslatorName` extension parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

0–255

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Table CAINXDFT does not provision a default for `pretranslatorName`.

ExtensionParameter primaryTrunkGroupSTS

Extension parameter definition

This extension parameter contains the serving translation scheme (STS) value associated with the returned ***PrimaryTrunkGroup*** parameter. The STS returned by the SCP (or datafilled in table CAINXDFT) replaces the STS defined at the time of the query when the corresponding parameter is used for routing the call.

Refer to Chapter 10, “Incoming CAIN messages,” for more information on how the `primaryTrunkGroupSTS` extension parameter is used in routing.

Figure 12-3 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter.

Figure 12-3
Precedence order for STS extension parameters

STS Extension Parameters / **Route Parameters Returned by the SCP**						**table CAINXDFT**					**Pre Query STS**
	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
OverflowRoutingNo				3	1				4	2	5

LEGEND

PRISTS (primaryTrunkGroupSTS)	– STS to be used with the ***PrimaryTrunkGroup*** parameter
ALTSTS (alternateTrunkGroupSTS)	– STS to be used with the ***AlternateTrunkGroup*** parameter
SALTSTS (secondAlternateTrunkGroupSTS)	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
STS (servTranslationScheme)	– STS to be used with the ***CalledPartyID*** parameter
OVFLSTS (overflowRoutingNo)	– STS to be used with the OverflowRoutingNo parameter

ExtensionParameter primaryTrunkGroupSTS (continued)

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

Message types

The following messages support the `primaryTrunkGroupSTS` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

0–999

CDR population

Table 12-69 shows the fields populated in the CDR when the ***`PrimaryTrunkGroup`*** parameter is used and the `primaryTrunkGroupSTS` extension parameter is received.

Table 12-69
primaryTrunkGroupSTS CDR population

CDR field	Population
OPART	This field contains the originating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
TPART	The field contains the terminating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-70 lists the application errors that can occur in regard to the `primaryTrunkGroupSTS` extension parameter.

Table 12-70
primaryTrunkGroupSTS application errors

Error type	Log generated	Reported to SCP?	Error action performed
STS not datafilled in table HNPACONT	CAIN100	No	Ignore the STS, and use the default from table CAINPARM when available. If unavailable, use the STS available before the query.
STS not datafilled in table STS2ACDB, STS2FTDB, STS2PXDB, STS2CTDB	OCC203	No	The call is sent to treatment. ***Note:*** This error applies only to Global IMTs.

Associated logs

CAIN100, OCC203

Restrictions

None

Datafill requirements

Provision a default primaryTrunkGroupSTS

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default primaryTrunkGroupSTS value using the following format:

>ADD caingrp PRISTS sts_value
where

caingrp	is the CAIN group requiring default STS values.
sts_value	is the default STS value.

Sample entry: **>ADD caingrp1 prists 761 $**

A default primaryTrunkGroupSTS has been provisioned

ExtensionParameter reorigAllowed

Extension parameter definition

This extension parameter, when present, contains an integer indicating the number of successive reoriginations allowed, 0 indicates that reorigination is not allowed.

Note: Call reorigination is still subject to in-switch feature and datafill restrictions which may override this parameter.

Message types

The following message supports the `reorigAllowed` extension parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

0–99

Note: Zero (0) indicates that reorigination is prohibited. Reorigination is blocked once the returned maximum is reached.

CDR population

None

Application errors

None

Associated logs

None

Restrictions

This parameter interacts with existing in-switch reorigination controls. Therefore, reorigination can be blocked even though the maximum number of reoriginations has not been reached. For example:

- A call originates on an agent where reorigination is disabled. Later in the call, an SCP response is received containing the `reorigAllowed` parameter. Regardless of the parameter value, reorigination is not allowed for this call.

- A call originates on an hotline-provisioned agent. Later in the call, an SCP response is received containing the `reorigAllowed` extension parameter. Since reorigination is not supported for hotline calls, no reorigination is allowed, regardless of the parameter value.

Datafill requirements

Table CAINXDFT does not provision a default for `reorigAllowed`.

ExtensionParameter satRestriction

Extension parameter definition

This extension parameter, when present, indicates the call should not terminate to a satellite-based trunk. This parameter replaces the satellite restriction value determined before the SCP query.

Message types

The following message supports the `satRestriction` extension parameter:

- **`Analyze_Route`**

Usage

Optional

Range of values

Presence or Absence

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default satRestriction

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Provision satellite restriction using the following format:

>ADD caingrp SATREST
where

caingrp	is the CAIN group requiring satellite restriction.

Sample entry: **>ADD caingrp1 SATREST**

Satellite restriction is provisioned.

ExtensionParameter secondAlternateTrunkGroupSTS

Extension parameter definition

This extension parameter contains the serving translation scheme (STS) value associated with the returned ***SecondAlternateTrunkGroup*** parameter. The STS returned by the SCP (or datafilled in table CAINXDFT) replaces the STS defined at the time of the query when the corresponding parameter is used for routing.

Refer to Chapter 10, "Incoming CAIN messages," for more information on how the `secondAlternateTrunkGroupSTS` extension parameter is used in routing.

Figure 12-4 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter.

Figure 12-4
Precedence order for STS extension parameters

STS Extension Parameters / Route Parameters Returned by the SCP	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	table CAINXDFT					Pre Query STS
						PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
OverflowRoutingNo				3	1				4	2	5

LEGEND

PRISTS (primaryTrunkGroupSTS)	– STS to be used with the ***PrimaryTrunkGroup*** parameter
ALTSTS (alternateTrunkGroupSTS)	– STS to be used with the ***AlternateTrunkGroup*** parameter
SALTSTS (secondAlternateTrunkGroupSTS)	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
STS (servTranslationScheme)	– STS to be used with the ***CalledPartyID*** parameter
OVFLSTS (overflowRoutingNo)	– STS to be used with the OverflowRoutingNo parameter

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

Message types

The following messages support the `secondAlternateTrunkGroupSTS` extension parameter:

- **`Analyze_Route`**
- **`Originate_Call`**

Usage

Optional

Range of values

0–999

CDR population

Table 12-71 shows the fields populated in the CDR when the ***`SecondAlternateTrunkGroup`*** parameter is used and the `secondAlternateTrunkGroupSTS` extension parameter is received.

Table 12-71
secondAlternateTrunkGroupSTS CDR population

CDR field	Population
OPART	This field contains the originating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
TPART	This field contains the terminating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-72 lists the application errors that can occur in regard to the `secondAlternateTrunkGroupSTS` extension parameter.

ExtensionParameter secondAlternateTrunkGroupSTS (end)

Table 12-72
secondAlternateTrunkGroupSTS application errors

Error type	Log generated	Reported to SCP?	Error action performed
STS not datafilled in table HNPACONT	CAIN100	No	Ignore the STS, and use the default from table CAINPARM when available. If unavailable, use the STS available before the query.
STS not datafilled in table STS2ACDB, STS2FTDB, STS2PXDB, STS2CTDB	OCC203	No	The call is sent to treatment. ***Note:*** This error only applies to Global IMTs.

Associated logs

CAIN100, OCC203

Restrictions

None

Datafill requirements

Provision a default secondAlternateTrunkGroupSTS

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default secondAlternateTrunkGroupSTS value using the following format:

>ADD caingrp SALTSTS sts_value
where

caingrp	is the CAIN group requiring default STS values.
sts_value	is the default STS value.

Sample entry: **>ADD caingrp1 saltsts 761 $**

A default secondAlternateTrunkGroupSTS has been provisioned

Extension parameter definition

This extension parameter contains a serving translation scheme (STS). The STS returned by the SCP (or datafilled in table CAINXDFT) replaces the STS defined at the time of the query. The STS can be overridden by other STS extension parameters.

Figure 12-5 illustrates the order of precedence used to obtain the STS for each CAIN routing parameter.

Figure 12-5
Precedence order for STS extension parameters

STS Extension Parameters / Route Parameters Returned by the SCP	PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	table CAINXDFT					Pre Query STS
						PRISTS	ALTSTS	SALTSTS	STS	OVLFSTS	
PrimaryTrunkGroup	1			3		2			4		5
AlternateTrunkGroup		1		3			2		4		5
SecondAlternateTrunkGroup			1	3				2	4		5
CalledPartyID				1					2		3
`OverflowRoutingNo`				3	1				4	2	5

LEGEND

`PRISTS (primaryTrunkGroupSTS)`	– STS to be used with the ***PrimaryTrunkGroup*** parameter
`ALTSTS (alternateTrunkGroupSTS)`	– STS to be used with the ***AlternateTrunkGroup*** parameter
`SALTSTS (secondAlternateTrunkGroupSTS)`	– STS to be used with the ***SecondAlternateTrunkGroup*** parameter
`STS (servTranslationScheme)`	– STS to be used with the ***CalledPartyID*** parameter
`OVFLSTS (overflowRoutingNo)`	– STS to be used with the `OverflowRoutingNo` parameter

Note: The `univIdx` extension parameter is used to specify the translations scheme for SS7 Global-IMT agents.

ExtensionParameter servTranslationScheme (continued)

Message types

The following messages support the `servTranslationScheme` extension parameter:

- **`Analyze_Route`**
- **`Send_To_Resource`** (only for STR-Connections)
- **`Connect_To_Resource`** (only for CTR-Connections)
- **`Originate_Call`**

Usage

Optional

Range of values

0–999

CDR population

Table 12-73 shows the fields populated in the CDR when the `servTranslationScheme` extension parameter is used.

Table 12-73
servTranslationScheme CDR population

CDR field	Population
OPART	This field contains the originating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
TPART	This field contains the terminating partition associated with the STS. Table STSTOPAR converts the STS into an OPART and TPART.
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Application errors

Table 12-74 lists the application errors that can occur in regard to the `servTranslationScheme` extension parameter.

Table 12-74
servTranslationScheme application errors

Error type	Log generated	Reported to SCP?	Error action performed
STS not datafilled in table HNPACONT	CAIN100	No	Ignore the STS, and use the default from table CAINPARM when available. If unavailable, use the STS available before the query.
STS not datafilled in table STS2ACDB, STS2FTDB, STS2PXDB, STS2CTDB	OCC203	No	The call is sent to treatment. ***Note:*** This error only applies to Global IMTs.

Associated logs

CAIN100, OCC203

Restrictions

None

Datafill requirements

Provision a default STS

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default STS value using the following format:

>ADD caingrp STS sts_value
where

caingrp	is the CAIN group requiring default STS values.
sts_value	is the default STS value.

Sample entry: **>ADD caingrp1 sts 761 $**

A default STS has been provisioned

ExtensionParameter shfelegs

Extension parameter definition

The `shfelegs` extension parameter indicates which leg or legs the switch monitors for a switch hook flash.

Note 1: The absence of this extension parameter, either in the message or in table CAINXDFT, indicates that all legs are capable of using the ***Switch_Hook_Flash*** EDP. Currently, the originator cannot produce a *Switch_Hook_Flash* event.

Message types

The following message supports the `shfelegs` extension parameter:

- **`Request_Report_BCM_Event`**

Usage

Optional

Range of values

Leg0 or Leg1

CDR population

None

Application errors

None

Associated logs

None

Restrictions

In certain call configurations, one or more legs are not monitored for a switch hook flash. Therefore, any values returned in the `shfelegs` extension parameter may be ignored, depending on which call configuration the call is to enter next. If all of the legs included in the extension parameter are to be ignored, then the request to arm the *Switch_Hook_Flash* event is also ignored. If this is the only EDP that is armed in the **`Request_Report_BCM_Event`** message, then a **`Close`** message is sent to the SCP. Table 12-75 lists the situations where `shfelegs` extension parameter settings are ignored.

ExtensionParameter shfelegs (end)

Table 12-75
shfelegs extension parameter use

CC transitioned to:	set to Leg 0	set to Leg 1
CC2	ignored	enables leg
CC4	enables leg	ignored
CC6	enables leg	ignored
CC7	enables leg	ignored
CC10	enables leg	ignored
CC11	ignored	ignored

Datafill requirements

Provision a default shfelegs

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default shfelegs parameter using the following format:

>ADD caingrp SHFELEGS shfeleg_type
where

caingrp	is the CAIN group requiring a default call leg(s) to for the switch to monitor.
shfeleg_type	is the call legid(s) to monitor (Leg0, Leg1).

Sample entry: **>ADD caingrp1 SHFELEGS Leg0**

A default `shfelegs` parameter has been provisioned.

ExtensionParameter strConnectionType

Extension parameter definition

This extension parameter indicates what type of connection protocol (IPI) is to be used to establish communication between the SCP, switch, and an IP resource.

Message types

The following messages support the `strConnectionType` extension parameter:

- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

NONE, CONNECT_ONLY, CONNECT_1129_STYLE

CDR population

None

Application errors

None

Associated logs

None

Restrictions

None

Datafill requirements

Provision a default strConnectionType

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

ExtensionParameter strConnectionType (end)

2 Add a default strConnectionType value using the following format:

>ADD caingrp STRCONTP str_conn_type
where

caingrp	is the CAIN group requiring a default STR-Connection type.
str_conn_type	is the type of connection protocol to use when establishing a connection with an IP (NONE, CONNECT_ONLY, CONNECT_1129_STYLE).

Sample entry: **>ADD caingrp1 STRCONTP connect_1129_style**

A default strConnectionType parameter has been provisioned.

ExtensionParameter treatment

Extension parameter definition

This extension parameter, when present, indicates the treatment to be set as a result of SCP service logic. The switch applies this treatment in place of AIND. When the switch receives a **`Send_To_Resource`** or **`Connect_To_Resource`** message that contains the `treatment` extension parameter, the resources identified in the **`Send_To_Resource`** or **`Connect_To_Resource`** message are played before the treatment resource.

Message types

The following messages support the `treatment` extension parameter:

- **`Disconnect`**
- **`Send_To_Resource`**
- **`Connect_To_Resource`**

Usage

Optional

Range of values

0 to 255

CDR population

Table 12-76 shows the fields populated in the CDR when the `treatment` extension parameter is used.

Table 12-76
treatment CDR population

CDR field	Population
TRTMTCD	This field contains the treatment code returned by the SCP (if used).
Note: For more information on CDR fields, refer to the *UCS DMS-250 Billing Records Application Guide*.	

Note: When the CAIN parameter TRTMTCD_COMPCODE_ZAPPED_ZERO in table CAINPARM is 'Y', the TRTMTCD and COMPCODE fields in the CDR are zapped to Zero for TBT (Take-Back & Transfer) calls terminated by the DISCONNECT message.

Application errors

Table 12-77 lists the application errors that can occur in regard to the `treatment` extension parameter.

Table 12-77
treatment application errors

Error type	Log generated	Reported to SCP?	Error action performed
Treatment not valid or out of range	CAIN100	No	AIND treatment is applied.

Associated logs

CAIN100

Restrictions

Only valid datafilled treatments should be returned in an SCP response.

Datafill requirements

Table CAINXDFT does not provision a default for `treatment`.

ExtensionParameter univIdx

Extension parameter definition

This extension parameter contains the universal translation scheme to be used for the call.

Message types

The following messages support the `univIdx` extension parameter:

- **`Analyze_Route`**
- **`Send_To_Resource`**

Usage

Optional

Range of values

NIL, AC, PX, CT, FA, OFC, DN, AM, FT, CC, NSC, CTY, NN, VPN

CDR population

None

Application errors

None

Associated logs

None

Restrictions

This parameter only applies to SS7 Global-IMTs.

Datafill requirements

Provision a default universal translation index

At the CI prompt

1 Enter table CAINXDFT.

Note: This procedure defines one default for a single tuple. Each CAIN group can have one set of defaults.

2 Add a default univIdx value using the following format:

>ADD caingrp univIdx_value
where

caingrp	is the CAIN group requiring default univIdx values.
univIdx_value	is the default univIdx value.

Sample entry: **>ADD caingrp1 univIdx FT $**

A default univIdx has been provisioned

Incoming IN/1 message parameters

Table 13-1 lists the parameters that may be sent from the SCP within a TCAP response IN/1 message.

Table 13-1
Incoming IN/1 message parameters

Parameter	Usage
AutomaticCallGapIndicators	Required
BillingIndicators	Required
Digits (CalledPartyNumber)	Required
Digits (CallingPartyNumber)	Required
Digits (CallerInteraction)	Required
Digits (RoutingNumber)	Required
Digits (LATA)	Required
Digits (Carrier)	Required
EchoData	Required
ErrorCode	Required
ProblemCode	Required
ProblemData	Required
StandardAnnouncement	Required

Fatal application errors

For more information on fatal application errors, refer to Chapter 1, “TCAP messaging,” Table 1-3, or each specific parameter within this chapter.

Nonfatal application errors

For more information on nonfatal application errors, refer to Chapter 1, “TCAP messaging,” Table 1-4, or each specific parameter within this chapter.

Associated logs

CAIN101, CAIN200, CAIN201

Associated OMs

TFREE533, VAMPACG

AutomaticCallGapIndicators

Parameter definition

The ***`AutomaticCallGapIndicators`*** parameter provides the switch with specific ACG control information.

Message types

The following message supports the ***`AutomaticCallGapIndicators`*** parameter:

- **`ACG`**

Usage

Required

Range of values

Control Cause Indicator field:

- vacant code
- out of band
- database overload
- destination mass calling
- SMS initiated

Duration field:

- 1 second
- 2 seconds
- 4 seconds
- 8 seconds
- 16 seconds
- 32 seconds
- 64 seconds
- 128 seconds
- 256 seconds
- 512 seconds
- 1024 seconds
- 2048 seconds
- infinity

AutomaticCallGapIndicators (end)

Gap field:

- 0 seconds
- 3 seconds
- 4 seconds
- 6 seconds
- 8 seconds
- 11 seconds
- 16 seconds
- 22 seconds
- 30 seconds
- 42 seconds
- 58 seconds
- 81 seconds
- 112 seconds
- 156 seconds
- 217 seconds
- 300 seconds

CDR population

None

Restrictions

None

Parameter definition

The ***`BillingIndicators`*** parameter contains an AMA call type code and a service feature identification code.

Message types

The following message supports the ***`BillingIndicators`*** parameter:

- **`Connect`**

Usage

Required

Range of values

AMA Call Type field:

- a three digit call type code (141 or 142)

Service Feature Identification field:

- a three digit code

CDR population

The AMA call type code and a service feature identification code are populated in the CDR fields CAINCT and SLPID. The CAINCT field is populated with a value of 4 when routing is performed by the Local Exchange Carrier (LEC).

Restrictions

None

Digits

Parameter definition

The ***`Digits`*** parameter contains any digits that the SCP passes to the switch.

Message types

The following messages support the ***`Digits`*** parameter:

- **`ACG`**
- **`Connect`**

Usage

Required

Range of values

Type of Digits field:

- called party number (dialed)
- calling party number (ANI)
- caller interaction
- routing number
- LATA
- carrier

Nature of Number field:

- national (NATL)
- international (INTL)

Numbering Plan field:

- unknown (UNK)
- ISDN
- telephony numbering plan (TELE)

Encoding Scheme field:

- Binary Coded Decimal (BCD)

Number of Digits field:

- a string of digits specifying the number of digits contained in the Digits field

Digit field:

- a string of digits of type 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, #.

CDR population

Carrier digits received are populated in the CIC CDR field. Routing digits are populated in the CALLEDNO CDR field.

Restrictions

None

EchoData (end)

Parameter definition

The UCS DMS-250 switch uses this parameter to match the SCP **`Termination`** message with the switch **`Termination_Information`** message.

Message types

The following message supports the ***`EchoData`*** parameter:

- **`Termination`**

Usage

Required

Range of values

This parameter contains an unique TCAP transaction identifier, represented by a string of 12 hexadecimal digits

CDR population

None

Restrictions

None

Parameter definition

The ***ErrorCode*** parameter contains the reason why the SCP is sending a **`Return_Error`** message to the switch.

Message types

The following message supports the ***ErrorCode*** parameter:

- **`Return_Error`**

Usage

Required

Range of values

Error Type field:

- unexpected component sequence
- unexpected data value
- unavailable resource
- data unavailable

CDR population

None

Restrictions

None

ProblemCode

Parameter definition

The ***ProblemCode*** parameter identifies the reason the SCP is sending a **`Reject`** message.

Message types

The following message supports the ***ProblemCode*** parameter:

- **`Reject`**

Usage

Required

Range of values

Problem Type field:

- General
- INVOKE
- RETURN RESULT
- RETURN ERROR
- Transaction Portion

Specifier field, for:

General problem type family

— Unrecognized Component

— Incorrect Component Portion

— Badly Structured Component

— Portion

INVOKE problem type family

— Duplicate INVOKE ID

— Unrecognized Operation Code

— Incorrect Parameter

— Unrecognized Correlation ID

RETURN RESULT family

— Unrecognized Correlation ID

— Unexpected RETURN RESULT

— Incorrect Parameter

RETURN ERROR family

— Unrecognized Correlation ID

— Unexpected RETURN ERROR

— Unrecognized Error

— Unexpected Error

— Incorrect Parameter

Transaction Portion family

— Unrecognized Package Type

— Incorrect Transaction Portion

— Badly Structured Transaction Portion

— Unrecognized Transaction ID

— Permission to Release Problem

— Resource Unavailable

CDR population

None

Restrictions

None

ProblemData (end)

Parameter definition

The switch sends the ***`ProblemData`*** parameter when there is an error indication caused by erroneous contents within a data element.

Message types

The following message supports the ***`ProblemData`*** parameter:

- **`Return_Error`**

Usage

Required

Range of values

This parameter contains the data element identifier, the data element length, and the erroneous data element.

CDR population

None

Restrictions

None

Parameter definition

The ***StandardAnnouncement*** parameter specifies the type of standard announcement to be played.

Message types

The following message supports the ***StandardAnnouncement*** parameter:

- **`Play_Announcement`**

Usage

Required

Range of values

Announcement Type field:

- out of band
- vacant code
- disconnected number
- reorder (120 IPM)
- busy (60 IPM)
- no circuit available
- reorder (announcement)
- audible ring

CDR population

None

Restrictions

None

Datafill requirements

Note: Refer to *Volume 1, Chapter 2, step 15* for provisioning requirements.

Digital Switching Systems

UCS DMS-250

NetworkBuilder Application Guide,
Volume 3 of 5

Product Documentation–Dept 3423
Nortel Networks
P.O. Box 13010
RTP, NC 27709–3010
1–877-662-5669

Publication number: 297-2621–370
Product release: UCS17
Document release: Standard 10.01
Date: July 2002
Printed in the United States of America

NORTEL NETWORKS™

How the world shares ideas.