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 5 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 5 of 5

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

Printed in the United States of America

Contents

Volume 1 of 5

Volume 2 of 5

Figures

Volume 3 of 5

Volume 4 of 5

Volume 5 of 5

NetworkBuilder tools

The NetworkBuilder platform offers several tools for troubleshooting and tracking purposes.

The following tools are available:

- ACGCNTRL
- CAINSCPT
- CAINTEST
- SCP simulator
- SOC

  Note: Although the SOC command set is not a CAIN tool, it is included here because it is required to enable CAIN.

- TRAVER

  Note: AXXESS agents use the FLEXSIM tool rather than TRAVER. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information on the FLEXSIM tool for AXXESS agents.

- VPTRACE

ACGCNTRL

The ACGCNTRL command displays and updates ACG control lists. CAIN applications use these ACG control lists with the Variable AIN Messaging (VAMP) framework. The control lists reduce the number of outgoing queries from the switch to the Service Control Point (SCP) by blocking specific queries.

ACGCNTRL provides the ability to

- add ACG controls to the Service Management System (SMS) Originated Control Code (SOCC) control list
- gap outgoing requests to all SCP destinations
- list specified ACG controls for a given VAMP application

- delete an individual ACG control for a given application
- remove all the controls in a control list for a given VAMP application1

Note: For additional information on ACGCNTRL commands, refer to the *UCS DMS-250 Commands Reference Manual*.

CAINSCPT

CAINSCPT supports the CAIN SCP Simulator.

When the CAINSCPT tool is activated, the user is able to perform the following tasks:

- idle the specified transaction identifiers (trid)[s]
- mark the specified trid(s) in use
- display information associated with the given trids
- detail information associated with the simulator's timeout wheel

Note: For additional information on CAINSCPT, refer to the *UCS DMS-250 Commands Reference Manual*.

CAINTEST

CAINTEST is the CAIN message query test tool. You can create and send TCAP queries to the SCP. The SCP responds as if an actual call were being made and does not recognize any difference between a CAINTEST-generated message and a call-generated message. You can display SCP queries and responses for maintenance and verification purposes.

CAINTEST provides the ability to

- generate queries for testing SCP interaction
- display SCP or SCP simulator responses
- automatically or manually enter into conversation with the SCP
- display, set, and clear message parameters
- list parameters for any particular switch-generated message
- override the CAIN_T1_TIMEOUT parameter for a particular test query
- display command syntax
- support the maintenance and verification of the SCP, or SCP simulator

Note 1: CAINTEST commands function differently for Local Number Portability agents. Refer to *UCS DMS-250 Local Number Portability Application Guide*.

Note 2: For additional information on CAINTEST commands, refer to the *UCS DMS-250 Commands Reference Manual*.

SCP simulator

Use the simulator to test CAIN functionality. You can datafill the simulator to provide CAIN testing, as well as robustness testing. Based on datafill, the simulator provides the following SCP services:

- accepts and decodes TCAP messages
- detects and reports protocol errors

  Note: When the error does not require an SCP response, the simulator may generate a CAIN901 log. (CAIN900 series logs are enabled and disabled by parameter CAIN900_LOGS_ENABLED table CAINPARM).

- checks incoming message parameters against the database you have built using tables CAINKEY and CAINMTCH
- determines if additional data is required to process the call
- determines appropriate response messages, including error cases
- encodes response messages in TCAP message format, using tables CAINRESP and CAINMTCH
- sends the TCAP message to the switch
- continues to collect data until datafill requirements are met
- validates (using table CAINCONV) data gathered during TCAP conversational digit collection and sends an appropriate message
- supports event processing

Robustness testing

Perform robustness testing by datafilling errors that can occur in communication between the CAIN framework and the SCP. You can datafill (in table CAINRESPP) the following information for robustness testing:

- nonrestricted messages with normally invalid package and component types
- messages with invalid parameters
- parameters with invalid data

SOC

Nortel Networks uses Software Optionality Control (SOC) to define and deliver software in product computing-module loads (PCL). Nortel Networks categorizes all functionality in a PCL as either base or optional. Base functionality is available for immediate use. Optional functionality is

grouped into commercial units called SOC options, which can be purchased by operating companies. SOC options correspond to functional groups and are controlled by Nortel Networks-supplied passwords.

SOC is the tool for managing options in a PCL. These options reside in the software. When an operating company purchases an option, SOC allows the company to monitor and control its use. You can order, activate, and use these options without a software reload or restart.

Note: For additional information on SOC commands, refer to the *UCS DMS-250 Commands Reference Manual*.

TRAVER

The TRAVER (translation verification) tool simulates a call from a user specified originating trunk to a user-specified address. TRAVER examines and displays translation and routing data for a single call leg.

TRAVER performs the following functions:

- verifies the translation tables
- aids in debugging and analyzing translation and routing datafill.
- helps determine reasons for unexpected results and changes required to achieve the expected results.

TRAVER is capable of displaying the following information:

- tables used to translate and route a call
- treatment
- CAIN subscription method and group
- tuple (from the appropriate trigger table) where trigger criteria was met
- limited messaging parameters

Note: For additional information on TRAVER, refer to the *UCS DMS-250 Commands Reference Manual*.

VPTRACE

VPTRACE allows you to enable or disable the tracing of VAMP messages (logged in VAMP 90x logs). If called with no parameter, the command displays the current status of tracing (enabled or disabled).

Note: For additional information on VPTRACE, refer to the *UCS DMS-250 Commands Reference Manual*.

CAINSCPT

CAINSCPT supports the CAIN SCP Simulator.

CAINSCPT performs the following functions:

- idles the specified transactions identifiers (trid)[s]
- marks the specified trid(s) in use
- displays information associated with the given trids
- details information associated with the simulator's timeout wheel

Commands

When the CAINSCPT tool is activated, the user can select different commands to perform different tasks. Table 2-1 provides a summary of these commands and their definitions.

Table 2-1
CAINSCPT commands

Command	Parameter	Definition
TRIDIDLE (Note)	all trid **trid_number** range **first_trid second_trid**	This command idles the specified trid(s).
TRIDUSE (Note)	all trid **trid_number** range **first_trid second_trid**	This command places the specified trid(s) in use. The purpose of this function is for testing and debugging.
Note: If the first entry is greater than the second entry they are switched internally so that the first entry is always less than or equal to the second entry.		
—continued—		

Table 2-1
CAINSCPT commands

Command	Parameter	Definition
TRIDINFO (Note)	all trid **trid_number** range **first_trid second_trid**	This command displays information associated with the given trid(s). If the trid(s) are idle, only that information is printed. If the trid(s) are in use, its state as well as all of the associated information that is kept with the trid(s) is displayed.
TIMERWHEEL (Note)	all slot **slot_number** range **first_slot second_slot**	This command provides detailed information associated with the simulator's timeout wheel. The command also indicates the **currslot** (current slot) being processed.
TIMERWHEEL (Note)	**currslot**	This command indicates the **currslot** (current slot) being processed.
Note: If the first entry is greater than the second entry they are switched internally so that the first entry is always less than or equal to the second entry.		
—end—		

Using CAINSCPT

The following examples show how you can use CAINSCPT commands.

Idle a trid

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function trid_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

trid_number is the specified trid to perform the function on (all, trid #, range #). If the first parameter in the range is larger than the second number, they are reversed.

Sample entry: **>CAINSCPT trididle trid 10**

Example of a MAP response:

```
TRID 10 IS ALREADY IN AN IDLE STATE.
```

OR

```
CLEARING TRID 10
```

Marking trids in use

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function trid_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

trid_number is the specified trid to perform the function on (all, trid #, range #). If the first parameter in the range is larger than the second number, they are reversed.

Sample entry: **>CAINSCPT triduse trid 10**

Example of a MAP response:

```
TRID 10 WAS PUT IN USE BY ANOTHER PROCESS.
```

OR

```
TRID 10 IS ALREADY IN USE.
```

OR

```
BUSYING TRID 10
```

Displaying information on trids for EDP messaging

EDP messaging example:

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function trid_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

trid_number is the specified trid to perform the function on (all, trid #, range #). If the first parameter in the range is larger than the second number, they are reversed.

Sample entry: **>CAINSCPT tridinfo trid 7**

Example of a MAP response:

```
TRID 7 IS IN USE
   INVOKE ID: 18
   TIMEOUT SLOT: 20
   TIMEOUT MINUTES REMAINING: 2
   THIS COULD POSSIBLY BE A LOST RESOURCE.
   NUM OF CORRELATION IDS: 0
   EDP CONVERSATION MODE
```

Displaying information on trids for STR messaging

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function trid_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

trid_number is the specified trid to perform the function on (all, trid #, range #). If the first parameter in the range is larger than the second number, they are reversed.

Sample entry: **>CAINSCPT tridinfo range 0 4**

Example of a MAP response:

```
TRID 0 IS IDLE
TRID 1 IS IN USE
   INVOKE ID: 16
   TIMEOUT SLOT: 20
   TIMEOUT MINUTES REMAINING: 0
   SECONDS UNTIL TIMEOUT: 10
   NUM OF CORRELATION IDs: 1
   CAINMTCH IDX: 102
   PLAYLIST IDX: 1
TRID 2 IS IDLE
TRID 3 IS IDLE
TRID 4 IS IDLE
```

Identifying trids in a particular slot of the timer wheel

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function slot_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

slot_number is the specified slot to perform the function on.

Sample entry: **>CAINSCPT timerwheel slot 5**

Example of a MAP response:

```
SLOT 5
```

Identifying the current slot of the timer wheel

At the CI prompt

1 From the MAPCI level enter:

>CAINSCPT function slot_number
and pressing the Enter key. ↵

where

function specifies which function (command) of CAINSCPT to perform.

slot_number is the specified trid to perform the function on.

Example of a MAP response:

```
CURRENT TIMER WHEEL:  90
```

Sample entry: **>CAINSCPT timerwheel currslot**

CAINTEST

CAUTION

Avoid CAINTEST during high traffic periods

CAINTEST sends actual messages that contribute to network traffic to the SCP. Avoid using this command during high traffic periods because of the low CPU priority of command interpreter (CI) commands.

Commands

When the CAINTEST tool is activated, the user is able to select different commands to perform different tasks. Table 3-1 provides a summary of these commands and their definitions:

ATTENTION

CAINTEST requires the CAIN0400 SOC option. Refer to the *UCS DMS-250 Software Optionality Control User's Manual* for more information on UCS DMS-250 SOC.

Table 3-1
CAINTEST commands

Command	Parameter	Definition
CLRPARM	CLRPARM STD parameter | EXT parameter | ALL	This command clears values of one or all parameters associated with the active message.
HELP	HELP fields	This command provides assistance on use of CAINTEST tool
LISTPARM	LISTPARM fields	This command lists all valid parameters for a specific message.
QUIT	QUIT	This command exits the CAINTEST tool and returns user to the CI prompt.
RESPORD	RESPORD order#	This command sets the order to which CAINTEST sends a response to the SCP.
SEND	SEND QUERY query# #_of_times | RESPONSE response#	This command sends the CAINTEST-defined CAIN query messages to the SCP or SCP simulator.
SETAPPL	SETAPPL applications	This command selects the application to be tested (CAIN02, IN1) .
SETFAM	SETFAM family	This command sets the AIN0.2 family (DMS-250) of extension parameters.
SETMSG	SETMSG messages	This command sets type of message to send to SCP or SCP simulator.
SETPARM	SETPARM STD parameter parm_value | EXT parameter parm_value	This command sets parameter values for the selected message.
SETQUERY	SETQUERY query#	This command sets the query number (this becomes the active query message).
SETRESP	SETRESP response#	This command populates the response message to a conversational package. This becomes the active response message.
SETTRANS	SETTRANS message protocol gt_name gt_string	This command sets the transport medium.
Note: Refer to *UCS DMS-250 Commands Reference Manual* for instructions on using CAINTEST commands.		

—continued—

Table 3-1
CAINTEST commands (continued)

Command	Parameter	Definition
SHOWFLDS	SHOWFLDS QUERY query_# | RESPONSE response_# | ALL	This command shows current parameter settings.
TIMEOUT	TIMEOUT time value	This command allows user to override the CAIN_T1_TIMEOUT value defined in table CAINPARM.
Note: Refer to *UCS DMS-250 Commands Reference Manual* for instructions on using CAINTEST commands.		
—end—		

Messaging

By using CAINTEST, you can instruct the switch to build messages and query the SCP without making a CAIN call. The switch uses the following modes of TCAP communication:

- query-response
- **`Send_To_Resource`** or **`Connect_To_Resource`** conversation
- EDP conversation

In a query-response scenario, the switch builds a query message and waits for a response from the SCP. The switch sends queries to the SCP in query packages and receives responses from the SCP in response packages.

`Send_To_Resource` and **`Connect_To_Resource`** conversational scenarios occur when the SCP requires additional information from the subscriber. The switch queries the SCP; the SCP, in turn, requests that the switch gather the required data (through a **`Send_To_Resource`** or a **`Connect_To_Resource`** message), beginning a conversation between the SCP and the switch. The SCP then waits for a reply from the switch. The conversation ends when the switch receives a response message in a response package.

EDP conversational scenarios occur when the SCP returns a **`Request_Report_BCM_Event`** component (arming EDPs) along with an

Analyze_Route, a **Continue**, or a **Collect_Information** message, in a conversational TCAP package.

Note: CAINTEST does not enforce mandatory TCAP parameter requirements; however, the CAIN TCAP encoders do check for mandatory parameters.

Query-Response TCAP

You have two options for using the **Send_To_Resource** or the **Connect_To_Resource** conversational portions of CAINTEST:

- You can specify all conversational response messages before sending the query. When the SCP returns a **Send_To_Resource** or **Connect_To_Resource** message, CAINTEST automatically responds with the message you specified.
- You can manually build the conversational response message when CAINTEST receives a **Send_To_Resource** or **Connect_To_Resource** message from the SCP.

EDP conversational TCAP

Using CAINTEST, you can define an EDP Notification, Request, or Close message for the EDP conversational messages in response to a conversational TCAP package containing an **Analyze_Route**, a **Continue**, or a **Collect_Information** message and a **Request_Report_BCM_Event** component arming EDPs:

You have two options for using the EDP conversational portions of CAINTEST:

- You can specify all conversational response messages before sending the query. When the SCP returns an **Analyze_Route**, **Continue**, or **Collect_Information** message containing a **Request_Report_BCM_Event** component, CAINTEST automatically responds with the message you specified.
- You can manually build the conversational response message when CAINTEST receives an **Analyze_Route**, **Continue**, or **Collect_Information** message containing a **Request_Report_BCM_Event** component from the SCP.

Restrictions

The following restrictions apply to CAINTEST:

- CAINTEST supports the messages and parameters supported in the UCS09 CAIN software release.
- More than one user can use CAINTEST. The number of MAP sessions that can be activated at one time limits the number of possible users.

- When the switch sends a test query (single or multiple messages) to the SCP, the display session must end before initiating another test query (single or multiple messages).
- CAINTEST can send multiple messages (up to 50) in one execution of the CAINTEST command.
- CAINTEST does not report application errors to the SCP, including CAIN_T1_TIMEOUT.
- CAINTEST can send only one reply message to a conversational package.

  Note: If you send multiple reply messages in response to a conversational **`Send_To_Resource`** or **`Connect_To_Resource`** message, any queued conversation packages timeout before you can read them. Therefore, you are restricted to one conversational response for every conversational **`Send_To_Resource`** or **`Connect_To_Resource`** message received from the SCP.

- If you did not build enough conversational response messages before sending the query, CAINTEST defaults to sending a **`Resource_Clear`** or **`CTR_Clear`** message. CAINTEST then prompts you for parameter data before sending the response.

Using CAINTEST

Use Procedure 3-1 to define and send a correctly formatted message to the SCP or SCP simulator. Refer to the "CAINTEST/SCP simulator interaction," section for a full example of CAINTEST.

Procedure 3-1
Using CAINTEST without conversation

At the CI prompt

1 Specify whether the test message should be sent to the SCP or the SCP simulator by datafilling table C7GTT appropriately.

Note: To send the message to the SCP simulator, field RESULT should contain SSNONLY.

2 From the MAPCI level, enter CAINTEST by typing:

>CAINTEST
and pressing the Enter key. ↵

3 OPTIONAL (default is CAIN02): Set the application to be tested using the following format:

>SETAPPL appl_name
where

appl_name is the application to be tested (CAIN02, IN1).

Sample entry: **>SETAPPL cain02**

4 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).
gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_clid_gt 9726841000**

5 OPTIONAL (default is DMS250): Set the family using the following format:

>SETFAM family
where

family is the system (DMS250).

Sample entry: **>SETFAM dms250**

6 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG info_analyzed**

7 OPTIONAL: Set the T1 timeout:

>TIMEOUT time value
where

time value is the time (in seconds) CAIN call processing waits for an SCP response before a time-out error occurs (default is 2).

Sample entry: **>TIMEOUT 30**

8 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD trigcrit cust_int**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

9 OPTIONAL: Set extension parameter values using the following format:

>SETPARM EXT parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM EXT adin 70**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid parameter options.

10 Send the message to the SCP or the SCP simulator (defined in step 1) by typing:

>SEND QUERY query# #_of_times
where

query # number of the query to be sent (0 to 3).
#_of_times is the number of times CAINTEST sends the message.

Sample entry: **>SEND query 1 3**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid rules.

Example of a MAP response:

```
Message sent, waiting for reply.

Message sent, waiting for reply.

Message sent, waiting for reply.

TIME --> Received response in 0.79 sec
ANALYZE_ROUTE received
Continue call under the direction of the SCP.
        PriTrk          : opls=N swid=11 trkgrp=220
Mailbox deallocated.

TIME --> Received response in 0.79 sec
ANALYZE_ROUTE received
Continue call under the direction of the SCP.
        PriTrk          : opls=N swid=11 trkgrp=220
Mailbox deallocated.

TIME --> Received response in 0.79 sec
ANALYZE_ROUTE received
Continue call under the direction of the SCP.
        PriTrk          : opls=N swid=11 trkgrp=220
Mailbox deallocated.
```

11 To exit CAINTEST, type

>QUIT

Use Procedure 3-2 to define and send a message to the SCP or SCP simulator with the Send_To_Resource conversation.

Procedure 3-2
Using CAINTEST with Send_To_Resource conversation

At the CI prompt

1 Specify whether the test message should be sent to the SCP or the SCP simulator by datafilling table C7GTT appropriately.

Note: To send the message to the in-switch SCP simulator, field RESULT should contain SSNONLY.

2 From the MAPCI level, enter CAINTEST by typing:

>CAINTEST
and pressing the Enter key. ↵

3 OPTIONAL (default is CAIN02): Set the application to be tested using the following format:

>SETAPPL appl_name
where

appl_name is the application to be tested (CAIN02, IN1).

Sample entry: **>SETAPPL cain02**

4 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).
gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_clid_gt 9726841000**

5 OPTIONAL (default is DMS250): Set the family using the following format:

>SETFAM family
where

family is the system (DMS250).

Sample entry: **>SETFAM dms250**

6 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG info_analyzed**

7 OPTIONAL: Set the T1 timeout:

>TIMEOUT time value
where

time value is the time (in seconds) CAIN call processing waits for an SCP response before a time-out error occurs (default is 2).

Sample entry: **>TIMEOUT 30**

8 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD trigcrit cust_int**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

9 OPTIONAL: Set extension parameter values using the following format:

>SETPARM EXT parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM EXT adin 70**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid parameter options.

10 OPTIONAL: Set the order in which CAINTEST sends a response to the SCP using the following format:

>RESPORD order#
where

order# is a set of provisioned responses (3 or less).

Sample entry: **>RESPORD 1**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid parameter options.

11 OPTIONAL: Set the response number using the following format:

>SETRESP response#
where

response# is the number of the response (1 to 3).

Sample entry: **>SETRESP 1**

12 OPTIONAL: Set the response message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG resource_clear**

13 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD clearcse normal**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

14 Send the messages to the SCP or the SCP simulator (defined in step 1) by typing:

>SEND QUERY query# #_of_times
where

query # number of the query to be sent (0 to 3).
#_of_times is the number of times CAINTEST sends the message.
response # number of the response to be sent (0 to 3).

Sample entry: **>SEND QUERY 1**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid rules.

Example of a MAP response:

```
Message sent, waiting for reply.
TIME --> Received response in 0.167 sec
SEND_TO_RESOURCE received
Conversation mode between SSP and SCP.
        RsrcType: COLL_DIGITS
        STRPARM : Action:DIGS_BLK  Size:Fixed Number:0
           INTER1: 14
```

Note: Response is automatically sent due to SETRESP command.

```
       RESPONSE     1

Application     : cain02
Transport       : tcap_sccp     cain_clid_gt    801
Timeout         : 1 sec
Family          : DMS250
Message         : resource_clear
        ClearCse: NORMAL
        CollDigs: UNKNOWN   1234567

Message sent, waiting for reply
Mailbox deallocated.
TIME --> Received response in 5.9 sec
Analyze_route -- Continue call under the direction of the
SCP
PARMS:          CHGNO:      NATL   2143562784
                CHGNO:      NATL   2143562784
                CLDNO:      NATL   9725323773
                OPULSNO:    NATL   9725323773
                ALTTRK:     OPLS=Y  SWID=111 TRKGRP=532
                PRIRTK:     OPLS=N  SWID=083 TRKGRP=2236
```

15 To exit CAINTEST, type

>QUIT

Use Procedure 3-3 to define and send a message to the SCP or simulator with the EDP conversation.

Procedure 3-3
Using CAINTEST with EDP conversation

At the CI prompt

1 From the MAPCI level, enter CAINTEST by typing:

>CAINTEST
and pressing the Enter key. ↵

2 OPTIONAL (default is CAIN02): Set the application to be tested using the following format:

>SETAPPL appl_name
where

appl_name is the application to be tested (CAIN02, IN1).

Sample entry: **>SETAPPL cain02**

3 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).
gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_addr_gt 801**

4 OPTIONAL (default is DMS250): Set the family using the following format:

>SETFAM family
where

family is the system (DMS250).

Sample entry: **>SETFAM dms250**

5 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG info_analyzed**

6 OPTIONAL: Set the T1 timeout:

>TIMEOUT time value
where

time value is the time (in seconds) CAIN call processing waits for an SCP response before a time-out error occurs (default is 2).

Sample entry: **>TIMEOUT 5**

7 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD userid cain dal273twdtls 11**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

8 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD bearcap speech**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

9 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD trigcrit cust_int**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

10 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD cldno natl 2731200**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

11 OPTIONAL: Set the response number using the following format:

>SETRESP response#
where

response# is the number of the response (1 to 3).

Sample entry: **>SETRESP 1**

12 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).
gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_addr_gt 801**

13 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG o_term_seized**

14 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD userid cain dal273twdtls 11**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

15 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD bearcap speech**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

16 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD notifind y**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

17 OPTIONAL: Set extension parameter values using the following format:

>SETPARM EXT parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM EXT termtrk dal dal220twdtgs 11**

Note: Refer to *UCS DMS-250 Commands Reference Manual* for valid parameter options.

18 OPTIONAL: Set the response number using the following format:

>SETRESP response#
where

response# is the number of the response (1 to 3).

Sample entry: **>SETRESP 2**

19 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).

gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_addr_gt 801**

20 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG o_answer**

21 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD userid cain dal273twdtls 11**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

22 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD bearcap speech**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

23 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD notifind y**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

24 OPTIONAL: Set the response number using the following format:

>SETRESP response#
where

response# is the number of the response (1 to 3).

Sample entry: **>SETRESP 3**

25 REQUIRED: Set the transport medium using the following format:

>SETTRANS msg_prtcl gt_name gt_value
where

msg_prtcl is the transport medium (TCAP_SCCP).
gt_name is the global title name (CAIN_CLID_GT, CAIN_ADDR_GT, CAIN_FEAT_GT, CAIN_OFCD_GT, or E800BELLCORE).
gt_value is the global title value.

Sample entry: **>SETTRANS tcap_sccp cain_addr_gt 801**

26 REQUIRED: Set the query message type using the following format:

>SETMSG messages
where

messages is the type of message to be sent. Refer to *UCS DMS-250 Commands Reference Manual* to obtain a current list of CAINTEST message values.

Sample entry: **>SETMSG Network_Busy**

27 OPTIONAL: Set the T1 timeout:

>TIMEOUT time value
where

time value is the time (in seconds) CAIN call processing waits for an SCP response before a time-out error occurs (default is 2).

Sample entry: **>TIMEOUT 5**

28 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD userid cain dal273twdtls 11**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

29 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD bearcap speech**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

30 OPTIONAL: Set parameter values using the following format:

>SETPARM STD parameter parm_value
where

parameter is the parameter name.
parm_value is the value of the parameter.

Sample entry: **>SETPARM STD notifind n**

Note: All required parameters must be specified (using the SETPARM command) in order for the message to be sent to the SCP without errors. Refer to *UCS DMS-250 Commands Reference Manual* for more information on CAINTEST commands and for valid parameter options.

31 >SHOWFLDS

Example of a MAP response:

```
       RESPONSE     1

Application     : cain02
Transport       : tcap_sccp     cain_addr_gt    801
Timeout         : 30 sec
Family          : DMS250
Message         : o_term_seized
     UserID     : cain    dal273twdt
     bearcap    : speech
     notifind   : y
```

32 Send the messages to the SCP or the SCP simulator by typing:

>SEND QUERY 1

Example of a MAP response:

```
Message sent, waiting for reply.
TIME --> Received response in 0.26 sec
```

```
Conversation mode between SSP and SCP.
ANALYZE_ROUTE received
     CHGNO: AUTH 6113311
     CLDNO: NATL 2201234
    PRITRK: OPLS=N   SWID=11   TRKGRP=220
     CHGNO: AUTH 6113311
Request_Report_BCM_Event received.
    EDPREQ: ONOANSWR   OCLDBUSY   NETBUSY
  EDPNOTIF: OANSWR     OTERMSZ
  ONOANSWT: 120
EDPS armed, EDP conversation started.
Valid responses are EDP-Request and EDP-Notification
messages armed by EDPREQ and EDPNOTIF parameters of the
REQUEST_REPORT_BCM_EVENT message respectively.
Close messages can also be sent to indicate EDP is completed.
The response order is not present.
You must manually build the response message.
Please enter in parms for a response message.
```

33 Send the response **O_Term_Seized** EDP-Notification message from CAINTEST.

>SEND RESPONSE 1

Example of a MAP response:

```
Message sent.
No reply expected
```

34 Send the response **O_Answer** EDP-Notification message from CAINTEST.

>SEND RESPONSE 2

Example of a MAP response:

```
Message sent.
No reply expected.
```

35 Send the response **Network_Busy** EDP-Request message from CAINTEST.

>SEND RESPONSE 3

Example of a MAP response:

```
Message sent.
EDPs disarmed, waiting for a reply.
TIME --> Received response in 0.146 sec
DISCONNECT received.
Message from the SCP to discontinue the call.
Mailbox deallocated.
```

36 To exit CAINTEST, type

>QUIT

SCP simulator

NetworkBuilder includes an SCP simulator used to test CAIN functionality. The previous chapters provide the necessary resources to manufacture testing scenarios. Simulator logic is provided through datafill.

ATTENTION

The SCP simulator requires the CAIN0300 SOC option. The switch generates a CAIN102 log when the switch attempts to use the SCP simulator without the CAIN0300 SOC option. Attempts to use the SCP simulator without the SOC option results in a T1 timeout. Refer to Chapter 5, "NetworkBuilder SOC functionality" in this volume for more information on SOC options.

CAUTION

Avoid using the simulator during high-traffic periods

Using the SCP simulator reduces call processing capacity of the switch.

Limitations and restrictions

The SCP simulator does not perform the following functions:

- provide service logic
- handle more than a total of four conversational **`Send_To_Resource`** or **`Connect_To_Resource`** messages
- perform audits; resources can be lost
- allow more than 100 open transactions at a given time

Simulator query processing

The simulator performs the following functions after receiving the query:

1 Decodes the query message.

2 Evaluates the parameters for errors.

3 Checks for incoming CAIN or IN/1 error messages.

Note 1: When the simulator detects a message that does not require an SCP response, it generates a CAIN901 log to indicate the message was received.

Note 2: CAIN900 series logs can be enabled or disabled in table CAINPARM, parameter CAIN900_LOGS_ENABLED.

4 Determines the appropriate response.

Note: The simulator does not include service logic. Table control is used to reflect incoming and outgoing messages. Query message parameters are matched against the datafill. The first set of datafill that matches the query parameters is used to form the appropriate response.

5 Generates a CAIN900 log.

Note 1: This log displays the datafill matching the message criteria used to form the SCP response.

Note 2: CAIN900 series logs can be enabled or disabled in table CAINPARM, parameter CAIN900_LOGS_ENABLED.

CAUTION
CAIN900 logs degrade performance
Enabled CAIN900 logs degrade the performance of the simulator.

Figure 4-1 shows an example of the SCP simulator query processing for CAIN. Figure 4-2 shows an example of the SCP simulator query processing for the IN/1.

Figure 4-1
SCP simulator query processing for CAIN

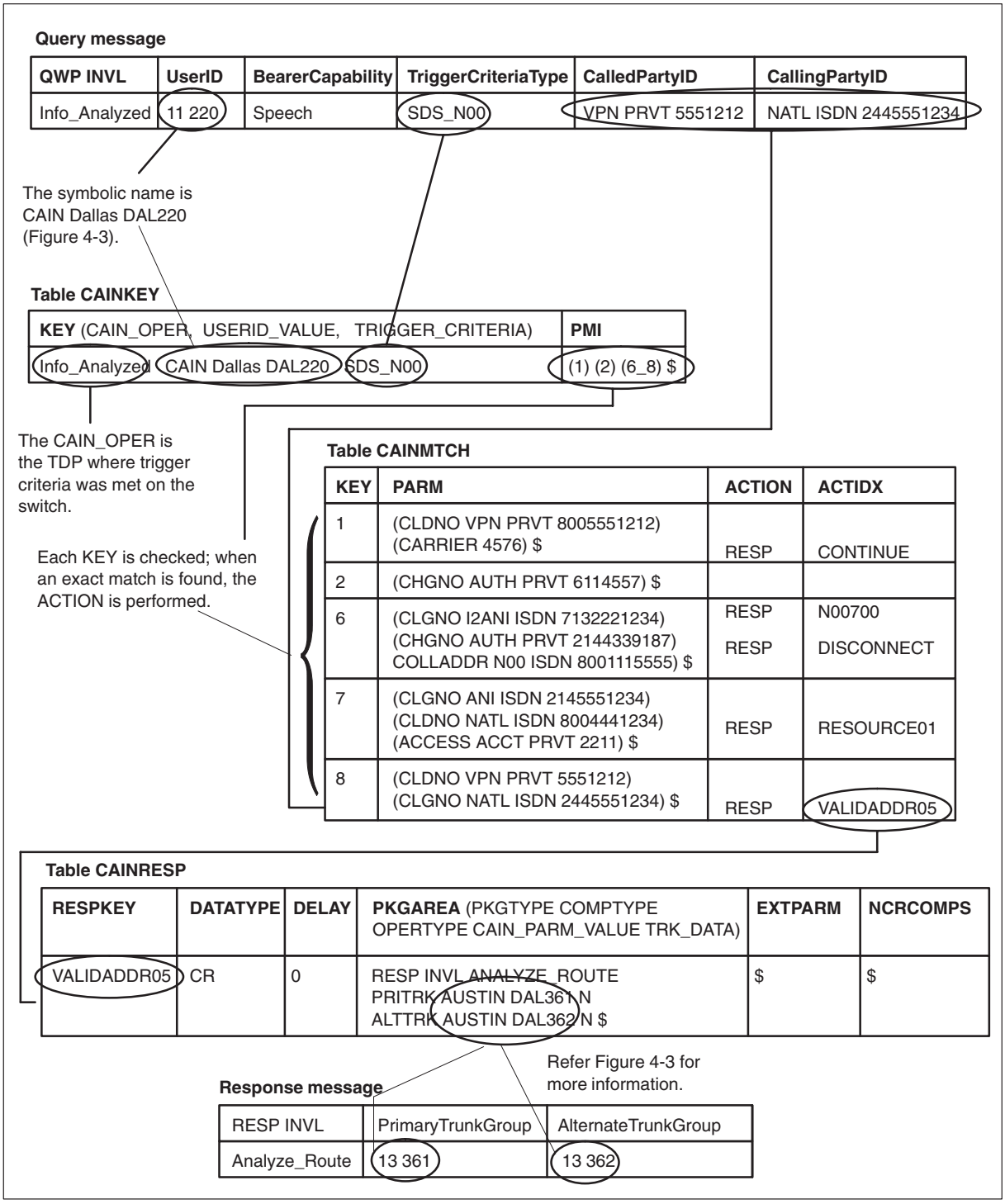
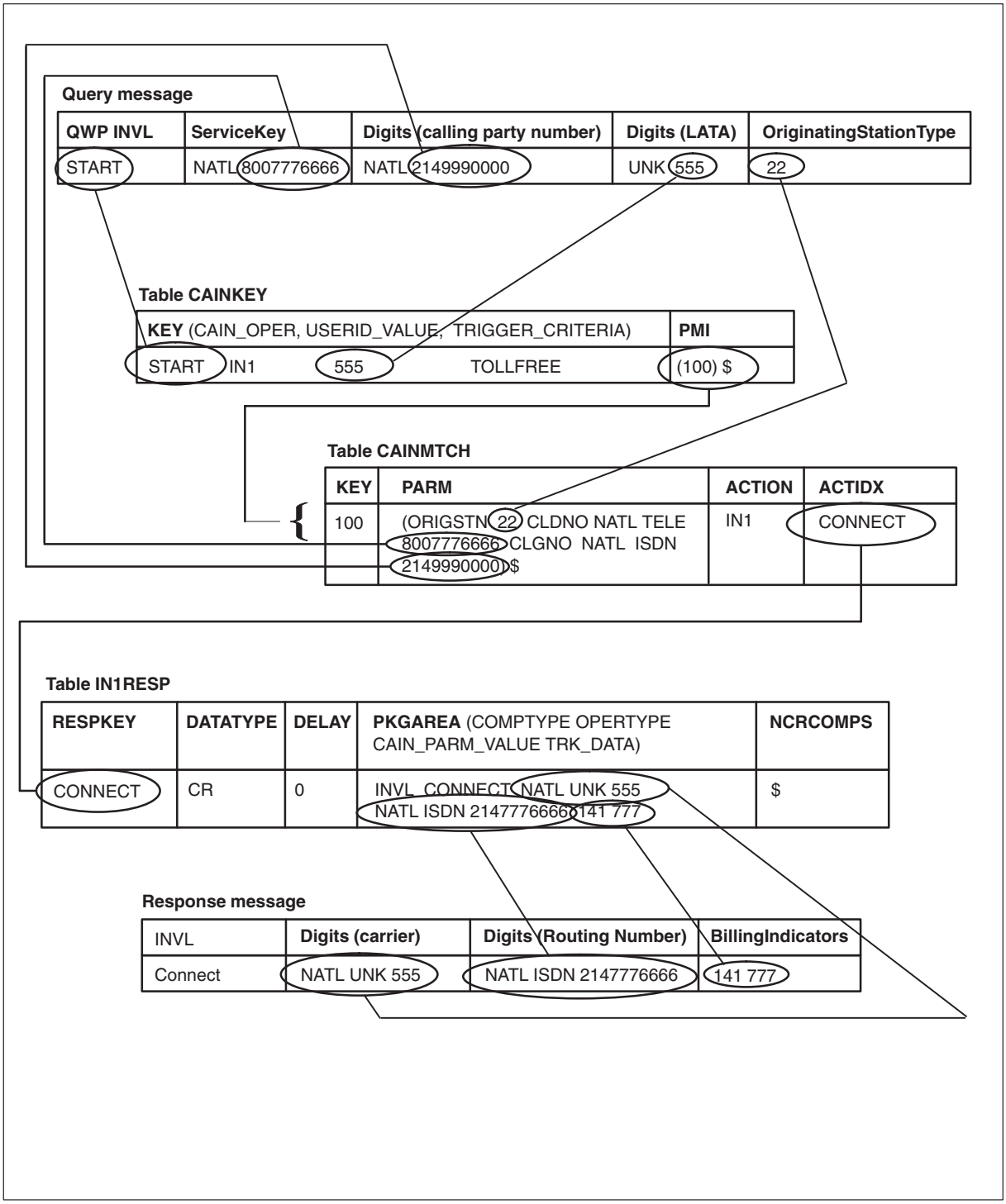


Figure 4-2
SCP simulator query processing for IN/1



Datafill requirements

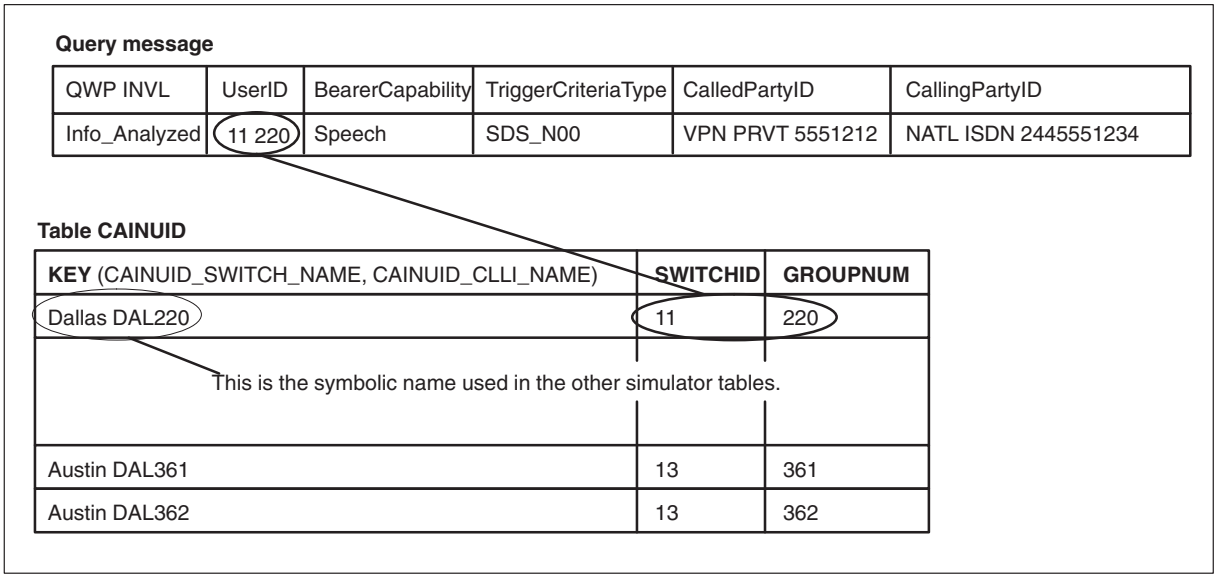
The SCP simulator consists of the following tables and minimal service logic:

- Table CAINUID (CAIN User Identification) – provides symbolic names for the ***UserID*** parameter and other trunk groups and switch identities used in the simulator. Figure 4-3 shows the relationship between the ***UserID*** parameter and table CAINUID.

  Note 1: The symbolic names are used when setting up the other simulator tables.

  Note 2: The CAIN_PROTOCOL_VERSION parameter in the table CAINPARM controls the ***UserID*** parameter population. When the CAIN_PROTOCOL_VERSION parameter is set to V2 or lower, the ***UserID*** parameter is populated as shown throughout this section. When the CAIN_ PROTOCOL_VERSION parameter is set to V3 or higher, the SwitchID is not included in the ***UserID*** parameter.

Figure 4-3
Relationship between the UserID parameter and table CAINUID



- Table CAINKEY (CAIN Key) – links the request message (operation_type), ***UserID***, and ***TriggerCriteriaType*** query parameters to a list of indexes into table CAINMTCH.

  Note: The ***UserID*** parameter supports four ***UserID*** formats. When the message is sent as a result of the *Office_Code* trigger or when the CAIN_PROTOCOL_VERSION is set to V3 or higher, the ***UserID*** parameter takes the TRK format.

— CAIN – identifies the key format as the original CAIN format, and that table CAINUID is used to determine the index into table CAINMATCH.

```
operation_type CAIN swid trkgrpnum trigger_criteria
```

— DN – identifies the key format that has a 10-digit number instead of a switch ID and trunk group number, and does not need to index table CAINUID.

```
operation_type DN (ten-digit number) trigger_criteria
```

— TRK – identifies the key format that has a TrunkGroupID and does not need to index table CAINUID.

```
operation_type TRK {0-0891} trigger_criteria
```

— IN1 – identifies the key format that has a 3-digit LATA and does not index table CAINUID.

```
operation_type IN1 (three-digit number)
trigger_criteria
```

— PRIFVAC – identifies the key format that has the private facility group encoding and does not index CAINUID.

```
operation_type PRIVFAC (0-0891) trigger_criteria
```

Note: Specify the value EDPREQ in the trigger criteria field to match on EDP-Requests.

- Table CAINMTCH (CAIN Match) – contains parameter sequences that are evaluated against the query parameters. Figure 4-4 provides examples that show how matching occurs.

Figure 4-4
CAINMTCH examples

Query message

QWP INVL	UserID	BearerCapability	TriggerCriteriaType	CalledPartyID	CallingPartyID
Info_Analyzed	11 220	Speech	SDS_N00	VPN PRVT 5551212	NATL ISDN 2145551234

Table CAINMTCH

KEY	PARMS	ACTION	ACTIDX
1	(CLDNO VPN PRVT 5551212) (CLGNO NATL ISDN 2145551234) $	RESP	RESPONSE1
2	(CLDNO NATL ISDN 2145557896) $	RESP	RESPONSE2
3	$	RESP	RESPONSE2
5	(CLDNO VPN PRVT 5553333) $	RESP	RESPONSE4
7	(CLDNO VPN PRVT 555) $	RESP	RESPONSE4

In this example, the query matches on KEY 1, 2, 3, and 7. The following describes the reasons for matching:

KEY 1: Both parameters (***CalledPartyID*** and ***CallingPartyID***) match in the query and table CAINMTCH.

KEY 2: One query parameter (***CalledPartyID***) matches the only CAINMTCH parameter.

KEY 3: When a CAINMTCH tuple is empty, all queries are considered a match.

KEY 7: When a CAINMTCH parameter contains fewer digits than the query parameter, but the available digits are the same, a match is made.

The query does not match on KEY 5; the parameter values do not match.

Note: If table CAINMTCH is provisioned as in this figure, the simulator would match on Key 1 and RESPONSE1 and table CAINRESP would be accessed.

Note 1: The matching criteria does not consider numbering plans.

Note 2: If no match is found in CAINKEY or CAINMTCH, an application error is returned and a CAIN901 log is generated.

- Table CAINRESP (CAIN Response) – contains information used to build the CAIN response messages.
- Table IN1RESP (IN/1 Response) – contains information used to build the IN/1 response messages.
- Table CAINREXT (CAIN Response Extension Parameters) – contains extension parameters used in building the CAIN response messages.

- Table CAINCONV (CAIN Conversation) – stores digit streams to check against collected digits and identifies the response. Figure 4-5 shows an example using table CAINCONV. Figure 4-6 shows an example of the message flow between the switch and the SCP simulator.

Figure 4-5
SCP simulator query processing example with table CAINCONV

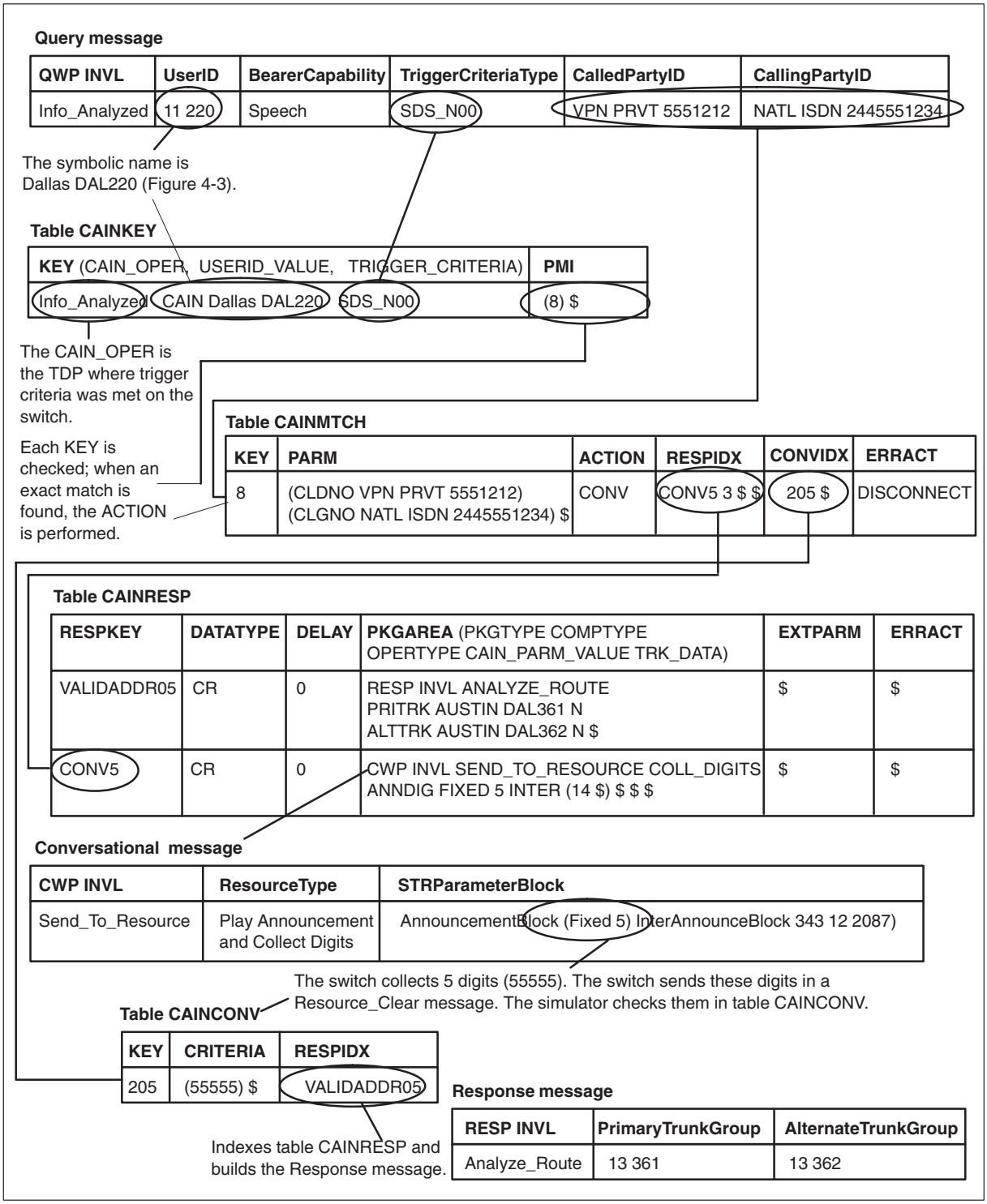
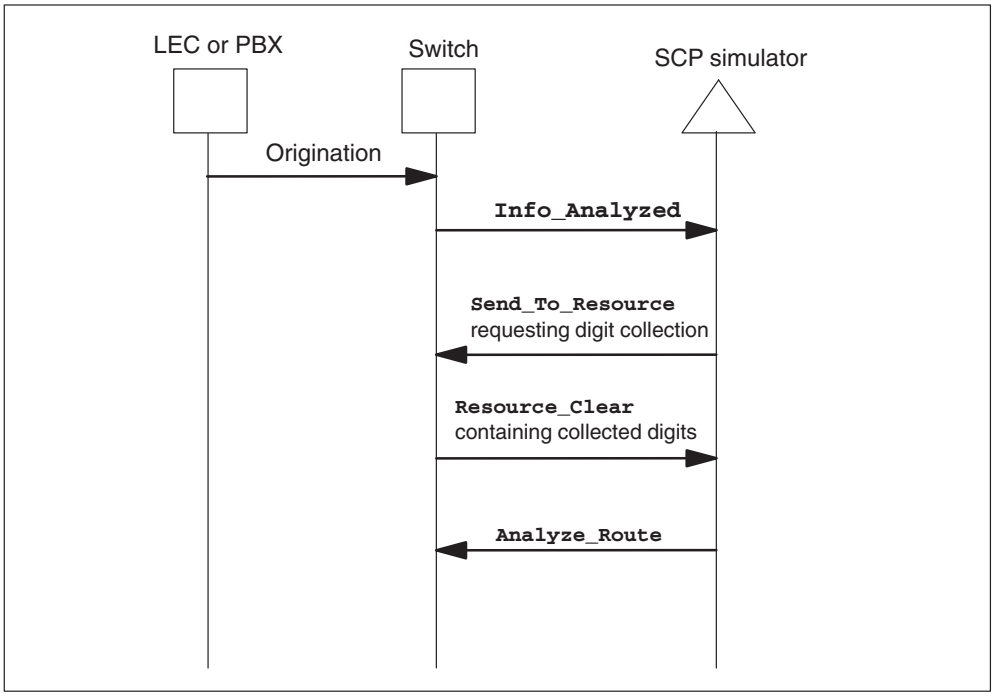


Figure 4-6
Send_To_Resource conversation with the SCP simulator



CAINRESP and IN1RESP prerequisite knowledge

Tables CAINRESP and IN1RESP datafill requires knowledge of TCAP messaging.

Responses datafilled in tables CAINRESP and IN1RESP requires the following information:

- data type
- package type (Note 1)
- component type (Note 2)
- operation type (Note 2)

Note 1: Package types are not used in table IN1RESP.

Note 2: An Abort response does not require a component or operation type.

Data types

Data type refers to the type of component generated by the SCP simulator. The supported data types are: call-related (CR) and non-call-related (NCR).

Package types

Package type refers to the type of package generated by the SCP simulator. Table 4-1 shows the supported packages. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Table 4-1
CAINRESP supported packages

Package type	Definition
QWP (Note)	Query with permission
CWP	Conversation with permission
RESP	Response
UNI (Note)	Unidirectional
Abort_P	Abort package

Note: QWP and UNI are supported for robustness testing.

Component types

The component type specifies the TCAP component portion of the response message. Table 4-2 shows supported component types. Refer to the *UCS DMS-250 NetworkBuilder AIN 0.2 TCAP Protocol Definition* for more information.

Table 4-2
CAINRESP and IN1RESP support component types (continued)

Component type	Definition
INVL	Invoke last
INVNL (Note 1)	Invoke not last
RERR	Return error
REJ	Reject
RRL (Note 2)	Return result last
RRNL (Notes 1 and 2)	Return result not last

Note 1: INVNL and RRNL are supported for robustness testing.

Note 2: RRL and RRNL are supported only for CAINRESP.

Operation types

The operation type specifies the operation or message to return to the switch. Table 4-3 shows supported operation types for table CAINRESP.

Table 4-3
CAINRESP supported operation types

Operation type	Valid DATA-TYPE	Valid PKGTYPE	Valid COMPTYPE	Definition
ACG	NCR	Not datafilled	INVL	Automatic code gapping
ACG_NR	(Note)	(Note)	(Note)	Automatic code gapping – nonrestricted
ACGGLOBAL_CTRL_ RESTORE	NCR	Not datafilled	INVL	Automatic code gapping global control restore
ACGGLOBAL_CTRL_ RESTORE_NR	(Note)	(Note)	(Note)	Automatic code gapping global control restore – nonrestricted
ACKNOWLEDGE	CR	RESP	INVL	Acknowledge
ACKNOWLEDGE_NR	(Note)	(Note)	(Note)	Acknowledge – nonrestricted
ANALYZE_ROUTE	CR	CWP RESP	INVL	Analyze route
ANALYZE_ROUTE_NR	(Note)	(Note)	(Note)	Analyze route – nonrestricted
APPL_ERROR	CR	CWP RESP	RERR	Application error
APPL_ERROR_NR	(Note)	(Note)	(Note)	Application error – nonrestricted
AUTHORIZE_TERMINATION	CR	CWP RESP	INVL	Authorize termination
AUTHORIZE_TERMINATION_ NR	(Note)	(Note)	(Note)	Authorize termination – nonrestricted

Note: When nonrestricted operation types are specified, parameter restrictions are not enforced and any information element may be datafilled.

—continued—

Table 4-3
CAINRESP supported operation types (continued)

Operation type	Valid DATA-TYPE	Valid PKGTYPE	Valid COMPTYPE	Definition
CALL_INFO_TO_RESOURCE	CR	CWP	RRL	Call information to resource
CALL_INFO_TO_RESOURCE_ NR	(Note)	(Note)	(Note)	Call information to resource – nonrestricted
CANCEL_RESOURCE	CR	CWP	INVL	Cancel resource
CANCEL_RESOURCE_NR	(Note)	(Note)	(Note)	Cancel resource – nonrestricted
CLOSE	CR	RESP	INVL	Close
CLOSE_NR	(Note)	(Note)	(Note)	Close – nonrestricted
COLLECT_INFORMATION	CR	CWP RESP	INVL	Collect information
COLLECT_INFORMATION_ NR	(Note)	(Note)	(Note)	Collect information – nonrestricted
CONNECT_TO_RESOURCE	CR	CWP RESP	INVL	Connect to resource
CONNECT_TO_RESOURCE_ NR	(Note)	(Note)	(Note)	Connect to resource – nonrestricted
CONTINUE	CR	CWP RESP	INVL	Continue
CONTINUE_NR	(Note)	(Note)	(Note)	Continue – nonrestricted
DISCONNECT	CR	RESP	INVL	Disconnect
DISCONNECT_NR	(Note)	(Note)	(Note)	Disconnect – nonrestricted
DISCONNECT_LEG	CR	RESP	INVL	Disconnect leg
DISCONNECT_LEG_NR	(Note)	(Note)	(Note)	Disconnect leg – nonrestricted
FAILURE_REPORT	CR	RESP	RERR	Failure report

Note: When nonrestricted operation types are specified, parameter restrictions are not enforced and any information element may be datafilled.

—continued—

Table 4-3
CAINRESP supported operation types (continued)

Operation type	Valid DATA-TYPE	Valid PKGTYPE	Valid COMPTYPE	Definition
FAILURE_REPORT_NR	(Note)	(Note)	(Note)	Failure report – nonrestricted
FURNISH_AMA_INFO	CR	RESP	INVL	Furnish AMA information
FURNISH_AMA_INFO_NR	(Note)	(Note)	(Note)	Furnish AMA information –nonrestricted
MERGE_CALL	CR	RESP	INVL	Merge call
MERGE_CALL_NR	(Note)	(Note)	(Note)	Merge call – nonrestricted
ORIGINATE_CALL	CR	RESP	INVL	Originate call
ORIGINATE_CALL_NR	(Note)	(Note)	(Note)	Originate call – nonrestricted
REPORT_ERROR	CR	UNI RESP	INVL	Report error
REPORT_ERROR_NR	(Note)	(Note)	(Note)	Report error – nonrestricted
REQ_REP_BCM	NCR	Not datafilled	INVL	Request report basic call model event
REQ_REP_BCM_NR	(Note)	(Note)	(Note)	Request report basic call model event – nonrestricted
SEND_NOTIFICATION	NCR	Not datafilled	INVL	Send notification
SEND_NOTIFICATION_NR	(Note)	(Note)	(Note)	Send notification – nonrestricted
SEND_TO_RESOURCE	CR	CWP RESP	INVL	Send to resource
SEND_TO_RESOURCE_NR	(Note)	(Note)	(Note)	Send to resource – nonrestricted

Note: When nonrestricted operation types are specified, parameter restrictions are not enforced and any information element may be datafilled.

—end—

Table 4-4 shows supported operation types for table IN1RESP.

Table 4-4
IN1RESP supported operation types

Operation type	Valid DATA-TYPE	Valid COMPTYPE	Definition
ACG	NCR	INVL	Automatic code gapping
CONNECT	CR	INVL	Connect
PLAY_ANN	CR	INVL	Play announcement
TERMINATION	NCR	INVL	Termination
ACG_NR	(Note)	(Note)	Automatic code gapping – nonrestricted
CONNECT_NR	(Note)	(Note)	Connect – nonrestricted
PLAY_ANN_NR	(Note)	(Note)	Play announcement – nonrestricted
TERMINATION_NR	(Note)	(Note)	Termination – nonrestricted

Note: When nonrestricted operation types are specified, parameter restrictions are not enforced and any information element may be datafilled.

Troubleshooting the simulator

The most common problems in the simulator are:

- Parameters in the message don't match the CAINMTCH datafill.
- The message, ***UserID***, and ***TriggerCriteriaType*** are not datafilled in table CAINKEY.
- The tuple to match on in table CAINMTCH is not datafilled in table CAINKEY.
- The digit streams received by the simulator don't match the datafill in table CAINCONV (occurs in conversation scenarios).
- Simulator attempts to access a tuple that is incorrectly datafilled in table CAINUID.

Remember the following information when you use the SCP simulator:

- The simulator only processes one message at a time.
- Message processing is completed for one message before moving to the next.
- Digit strings in tables CAINMTCH and CAINCONV have implied wildcards at the end of them. For example, if a digit string is datafilled as 214, all digit strings starting with 214 will match.
- When the switch receives an **Application_Error** from the simulator, check table CAINMTCH datafill.

Datafill procedures

Table 4-5 shows the procedures required to datafill the SCP simulator. All SCP simulator tables (CAINUID, CAINKEY, CAINMTCH, CAINCONV, CAINRESP, IN1RESP, and CAINREXT) must be datafilled. There are several procedures for datafilling tables CAINMTCH, CAINRESP, and IN1RESP. Refer to Table 4-5 for types of datafill available.

Note: There are no procedures for datafilling a nonrestricted operation. Refer to *UCS DMS-250 Data Schema Reference Manual* for complete details on tables CAINRESP and IN1RESP.

Table 4-5
SCP simulator datafill procedures

Procedure	Table name	Page
Procedure 4–1, Enable the CAIN900 logs	CAINPARM	4-20
Procedure 4-2, Define symbolic trunk and switch identifier name	CAINUID	4-21
Procedure 4-3, Define required parameter matching criteria	CAINKEY	4-22
Procedure 4-4, Define extension parameters	CAINREXT	4-24
Procedure 4-5, Define a **ACG** message	CAINRESP	4-25
Procedure 4–6, Define an IN/1 **ACG** message	IN1RESP	4-26
Procedure 4-7, Define an **ACG_Global_Ctrl_Restore** message	CAINRESP	4-27
Procedure 4-8, Define a **Acknowledge** message	CAINRESP	4-28
Procedure 4-9, Define an **Analyze_Route** message	CAINRESP	4-29
Procedure 4-10, Define an **Application_Error** response	CAINRESP	4-30
Procedure 4-11, Define an **Authorize_Termination** response	CAINRESP	4-31
Procedure 4-12, Define a **Call_Info_To_Resource** message	CAINRESP	4-32
Procedure 4-13, Define a **Cancel_Resource_Event**	CAINRESP	4-34
Procedure 4-14, Define a **Close** response	CAINRESP	4-36
Procedure 4-15, Define a **Collect_Information** response	CAINRESP	4-38
Procedure 4-16, Define a **Connect** message	IN1RESP	4-39
Procedure 4-17, Define a conversational **Connect_To_Resource** message	CAINRESP	4-40
Procedure 4-18, Define a non-conversational **Connect_To_Resource** response	CAINRESP	4-42
Procedure 4-19, Define a **Continue** message	CAINRESP	4-44
Procedure 4-20, Define a **Disconnect** response	CAINRESP	4-45
Procedure 4-21, Define a **Disconnect_Leg** message	CAINRESP	4-46
Procedure 4-22, Define a **Failure_Report** response	CAINRESP	4-47
Procedure 4-23, Define a **Furnish_AMA_Information** message	CAINRESP	4-50
Procedure 4-24, Define a **Merge_Call** message	CAINRESP	4-50
Procedure 4-25, Define a **Originate_Call** message	CAINRESP	4-51
Procedure 4-26, Define a **Play_Announcement** message	IN1RESP	4-53

—continued—

Table 4-5
SCP simulator datafill procedures (continued)

Procedure	Table name	Page
Procedure 4-27, Define a **Report_Error** response	CAINRESP	4-54
Procedure 4-28, Define a **Request_Report_BCM_Event** message	CAINRESP	4-56
Procedure 4-29, Define a **Send_Notification** message	CAINRESP	4-58
Procedure 4-30, Define a non-conversational **Send_To_Resource** response	CAINRESP	4-59
Procedure 4-31, Define a conversational **Send_To_Resource** message	CAINRESP	4-61
Procedure 4-32, Define a **Termination** message	IN1RESP	4-63
Procedure 4-33, Define an Abort package	CAINRESP	4-64
Procedure 4-34, Define a Response package with a reject component	CAINRESP	4-65
Procedure 4-35, Define optional parameter matching criteria	CAINMTCH	4-66
Procedure 4-36, Define digit collection validation	CAINCONV	4-68
—end—		

Use Procedure 4-1 to enable the CAIN900 logs.

Procedure 4-1
Enable the CAIN900 logs
At the CI prompt

1 Enter table CAINPARM.

2 Replace the parameter by typing:

>REP CAIN900_LOGS_ENABLED parmval
where

parmval the enabled log number (up to seven values: 900, 901, 902, 903, 904, 905, or 906)

Note: CAIN900 and CAIN901 are the only applicable values for the simulator logs.

Sample entry: **>REP CAIN900_LOGS_ENABLED 900 901 $**

CAIN900 logs are enabled.

Use Procedure 4-2 to define the symbolic trunk and switch identifier names.

Procedure 4-2
Define symbolic trunk and switch identifier names

Note: There are no prerequisite datafill requirements for table CAINUID.

At the CI prompt

1 Enter table CAINUID.

2 Define symbolic trunk and switch identifier names by using the following format:

>ADD key switchid groupnum
where

key is comprised of two subfields: CAINUID_SWITCH_NAME and CAINUID_CLLI_NAME, where
- CAINUID_SWITCH_NAME is the user-defined symbolic name used to identify the agency.
- CAINUID_CLLI_NAME is the symbolic name used to identify the switch.

switchid is the unique numeric identifier assigned to the switch.

groupnum is a symbolic numeric identifier used to identify the trunk and should match the entry in table CLLI.

Sample entry: **>ADD dallas dal220 11 220**

The symbolic name is defined.

Use Procedure 4-3 to define the required parameter matching criteria.

Procedure 4-3
Define required parameter matching criteria

Note: Table CAINUID must be datafilled before table CAINKEY.

At the CI prompt

1 Enter table CAINKEY.

2 Define symbolic trunk and switch identifier names by using the following format:

>ADD key pmi
where

key is comprised of three subfields: CAIN_OPER, USERID_VALUE, and TRIGGER_CRITERIA, where

CAIN_OPER is the CAIN operation requested by the switch (FAILURE_OUTCOME, INFO_ANALYZED, INFO_COLLECTED, NETWORK_BUSY, O_ABANDON, O_ANSWER, O_CALLED_PARTY_BUSY, O_DISCONNECT, O_FEATURE_REQUESTED, O_MID_CALL, O_NO_ANSWER, ORIGINATION_ATTEMPT, O_TERM_SEIZED, TERMINATION_ATTEMPT, TIMEOUT, or the wildcard [NIL]).

USERID_VALUE is a valid KEY in table CAINUID, 10 digit number, or a TrunkGroupID. If the selector is CAIN, datafill the 2-part field consisting of a CAINUID_SWITCH_NAME and CAIN_UID_CLLI datafilled in the KEY field of table CAINUID. If the selector is set to DN, enter the dialed digits. If the selector is TRK, enter a valid TrunkGroupID.

TRIGGER_CRITERIA is the trigger criteria evaluated before querying the SCP simulator (OFFHKIM, O_FEAT, OFFHKDEL, CSP_CLID, CSP_ADDR, CSP_ADIN, CSP_N00, CSP_INTL, SIO_ANI, SIO_ADDR, SIO_ADIN, SIO_N00, SIO_INFO, SIO_INTL, SIO_CIC, SIO_STD, PRI_STD, SPEC_FEAT, CUST_INT, SDS_ADDR, SDS_ADIN, SDS_INTL, SDS_ANI, SDS_N00, SDS_INFO, SDS_CIC, NPA, NPA_N, NPA_NX, NPA_NXX, NPA_NXXX, NPA_NXXXX, NPA_NXXXXX, LNP_OFCD, NETWBSY, O_CLDBSY, O_NOANSW, OIECREO, TERM_ATT, EDPREQ to match on EDP-requests, or the wildcard [NIL]).

pmi is multiple-entry vector that indexes the key in table CAINMTCH, indicating a range of values used for optional parameter criteria matching.

Sample entry: **>ADD info_analyzed cain dallas dal220 cust_int 1_2 4 6_8 $**

Note: The underscore symbol (_), used to separate the PMI, indicates a range of indexes. For example, 6_8 indicates that the 6, 7, and 8 keys of table CAINMTCH are evaluated in that order for matching criteria.

Required matching parameter criteria is defined.

Use Procedure 4-4 to define an extension parameter list.

Procedure 4-4
Define an extension parameter list

Note: Table CAINREXT must be datafilled before table CAINRESP.

At the CI prompt

1 Enter table CAINREXT.

2 Define the outgoing extension parameter list by using the following format:

>ADD rextkey customer_selector extparm $

where

rextkey is a string used by field EXTPARM in table CAINRESP.

customer_selector is a field that identifes the customer type.

Note: This field can have the values of RESVCUST1 or NT. When this field is NT, there is no change. The same existing parameters are supported. RESVCUST1 is for future use and cannot be provision at this time.

extparm is a multiple-entry vector that contains the outgoing extension parameters (ACCTCODE, ALTSTS, AMADIGS, BILLSEQ, BRANDING, CAINGRP, CALLCTRL, CALLTYPE, CONNECTTOSCU, COS, EDPBUFFR, NETBACT, NETINFO, OCLDBACT, ONOANACT, OVFLSTS, PINDIGS, PREXLA, PRISTS, REORIG_ALLOWED, SALSTS, SATREST, SHFELEGS, STRCONTP, STS, SWID, TREATMENT, UNIVIDX, or the wildcard (NIL)).

Sample entry: **>ADD ext01 caingrp 0 $**

An extension parameter list is defined.

Use Procedure 4-5 to define a CAIN ACG message.

Procedure 4-5
Define a CAIN ACG message

Note: Non-call related components must be datafilled before the call-related components with which they will be included.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **ACG** message by using the following format:

>ADD respkey datatype comparea
where

respkey is the user-defined name used in table CAINRESP to determine the non-call related component (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
comparea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as ACG.

5 Datafill CAIN_PARM_VALUE.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–5 parameters can be defined. (CTRLTYPE, TRANSTYPE, GTA, GAPDUR, GAPINTVL)

CTRLTYPE	is the type of ACG control (SCP or SOCC).
TRANSTYPE	is the translation type (0–99).
GTA	is the global title address (2 to 10 global title address digits).
GAPDUR	is the gap duration integer value (in seconds).
GAPINTVL	is the gap interval integer value (in seconds).

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

Sample entry: **>ADD acg01 ncr invl acg cntrltype socc gapdur 4 gapintvl 30 transtype 30 gta 214684 $ $**

The ACG message is defined.

Use Procedure 4-6 to define an IN/1 ACG message.

Procedure 4-6
Define an IN/1 ACG message

Note: Table IN1RESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table IN1RESP.

2 Define an IN/1 **ACG** message by using the following format:

>ADD respkey datatype pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
pkgarea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as ACG.

5 Datafill the CLDNO refinement. Enter an NOA (NATL, INTL), an IN/1 numbering plan, and 0–24 digits.

6 Datafill the ACGIND refinement. Enter an ACG gap interval, gap duration, and control cause indicator.

Sample entry: **>ADD acg_component ncr invl acg natl isdn 8881112222 3_seconds 64_seconds vacant_code**

ACG response is defined.

Use Procedure 4-7 to define an ACG_Global_Ctrl_Restore message.

Procedure 4-7
Define an ACG_Global_Ctrl_Restore message

Note: Non-call related components must be datafilled before the call-related components with which they will be included.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`ACG_Global_Ctrl_Restore`** message by using the following format:

>ADD respkey datatype comparea
where

respkey is the user-defined name used in table CAINRESP to determine the non-call related component (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
comparea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as ACGGLOBAL_CTRL_RESTORE.

5 Datafill ACG_GLOBAL_OVERRIDE parameter.

ACG_GLOBAL_OVERRIDE is currently the only parameter that can be defined. The ACG_GLOBAL_OVERRIDE parameter indicates the type of control to restore. (ALL_ITEMS, SCP_ITEMS, SOCC_ITEMS, SOCC_NON_ZERO, NTMOS_ITEM, NTMOS_NOT_ZERO, CRAFT_ITEMS, or CRAFT_NOT_ZERO)

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

Sample entry: **>ADD acgglobalctrlrestore01 ncr invl acgglobal_ctrl_restore scp_items $**

The ACGGLOBAL_CTRL_RESTORE message is defined.

Use Procedure 4-8 to define an Acknowledge message.

Procedure 4-8
Define an Acknowledge message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Acknowledge`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a conversation with permission (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as ACKNOWLEDGE.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.
Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **> ADD CR 0 CWP INVL ACKNOWLEDGE $ $ (SHI_BOTH) $**

Acknowledge response is defined.

Use Procedure 4-9 to define an Analyze_Route message.

Procedure 4-9
Define an Analyze_Route message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Analyze_Route`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a response (RESP) or conversation (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as ANALYZE_ROUTE.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.
Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response1 cr 0 resp invl analyze_route pritrk dallas dal220 y cldno natl isdn 2148815432 $ $ $**

Analyze_Route response is defined.

Use Procedure 4-10 to define an Application_Error response.

Procedure 4-10
Define an Application_Error response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Application_Error`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters)
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).

3 Datafill subfield PKGTYPE (package type) as response (RESP)

4 Datafill subfield COMPTYPE (TCAP component type) as return error (RERR).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as APPL_ERROR.

6 Datafill refinement APPL_ERROR_STRING with one of the following.

APPL_ERROR_STRING is one of the following application errors:
DATA_ERR – data error
MISS_PARM – missing parameter
T1_EXPIRE – T1 timer expired
UNEXP_PARM – unexpected parameter
UNEXP_MSG– unexpected message
UNXP_MSGSEQ – unexpected message sequence
UNXP_PRMSEQ – unexpected parameter sequence

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response4 cr 0 resp rerr appl_error unexp_parm $ $**

Application_Error response is defined.

Use Procedure 4-11 to define an Authorize_Termination message.

Procedure 4-11
Define an Authorize_Termination message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Authorize_Termination`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a response (RESP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as AUTHORIZE_TERMINATION.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.
Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD termresp1 cr 0 resp invl authorize_termination clgno natl isdn 9726848689 amadigs 001 2222 $ $ $**

Authorize_Termination response is defined.

Use Procedure 4-12 to define a Call_Info_To_Resource response.

Procedure 4-12
Define a Call_Info_To_Resource response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Call_Info_To_Resource`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99)
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as response (CWP)

4 Datafill subfield COMPTYPE (TCAP component type) as return result last (RRL).

5 Datafill subfield RRSCENARIO (message type to be returned to the switch) as CALL_INFO_TO_RESOURCE.

6 Datafill refinements BLOCKTYPE (ANN, ANNDIG, FLEX).

For ANN, where

BLK_FORMAT is the block format (UNINTER).
ANN_BLK: identifies up to four announcements to play. The format for this field is: ann_id info_digs
Where,
ANN_ID is 0–4095
INFO_DIGS is 0–12 digits.
CAIN_CITR_PARM_VALUE is the parameter returned to the switch (NIL, RESOURCE_TYPE).
RESOURCE is the resource type (PLAY_ANN)

For ANNDIG, where

DIGS_TYPE defines the number of digits to collect (FIXED, UP_TO, VARIABLE, NORMAL)
NUM_DIGS 0–255. (Only used if DIGS_TYPE is FIXED or UP_TO).
BLK_FORMAT1 is the block format (UNINTER or INTER). Datafill the ANN_ID and INFO_DIGS refinements.

ANN_ID: 0–4095
INFO_DIGS: 0–12 digits.

BLK_FORMAT2 is the block format (INTER or NIL). Only used if BLK_FORMAT1 is UNINTER. Datafill the ANN_ID and INFO_DIGS refinements.
ANN_ID: 0–4095
INFO_DIGS: 0–12 digits.

CAIN_CITR_PARM_VALUE is the parameter returned to the switch (NIL, RESOURCE_TYPE).

RESOURCE is the resource type (COLL_DIGITS)

For FLEX, where

FLEX_FORMAT is the FLEX format (FLEX_PARM_NT and FLEX_PARM_HEX)

FLEX_PARM_BLOCK is a multiple-entry vector of up to 112 octets

CAIN_CITR_PARM_VALUE is the parameter returned to the switch (NIL, RESOURCE_TYPE).

RESOURCE is the resource type (FLEX_PARM)

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD citr_play cr 0 cwp rrl call_info_to_resource flex flex_parm_hex 11 22 33 44 55 $ $ resource_type flex_parm $ $ $**

Call_Info_To_Resource response is defined.

Use Procedure 4-13 to define a Cancel_Resource_Event message.

Procedure 4-13
Define a Cancel_Resource_Event message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Cancel_Resource_Event`** message by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).

datatype is the type of data in the call response (CR or NCR). Enter CR.

delay is a number in seconds that can be used to simulate the amount of the **`Send_To_Resource`** or **`Connect_To_Resource`** interaction allowed before sending the **`Cancel_Resource_Event`** (0–99).

pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as conversation with permission (CWP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as CANCEL_RESOURCE.

6 Datafill refinement STR_IDX (**`Send_To_Resource`** or **`Connect_To_Resource`** index) with a valid entry into table CAINRESP identifying a conversational **`Send_To_Resource`** or **`Connect_To_Resource`** message.

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD cancel01 5 cwp invl cancel_resource response2 $ $**

Note: In this sample CAINRESP entry, table CAINMTCH indexes the CANCEL01 tuple. The **`Send_To_Resource`** or **`Connect_To_Resource`** message specified in this tuple RESPONSE2 are sent to the CAIN framework. A delay of 5 seconds is performed. Then the **`Cancel_Resource_Event`** message is sent.

The Cancel_Resource_Event message is defined.

Use Procedure 4-14 to define a Close response.

Procedure 4-14
Define a Close response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Close`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99)
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as response (RESP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as CLOSE.

6 Datafill refinement PARM_ID with the following optional parameters.

Note: When CLOSE is selected for OPERTYPE, datafill the USERID, BEARCAP, and CLOSECSE refinements.

USERID is the symbolic name datafilled in table CAINUID.
BEARCAP is the bearer capability (SPEECH, F3_1KHZ, F6_KHZ, F56KBPS, F64KBPS, MULTI).
CLOSECSE Enter CLOSECSE to define the CloseCause parameter. Datafill the CLOSECSE refinement.

CLOSECSE	CALL_TERM – call terminated (robustness) EDP_COMPL – event detection point complete (robustness) UNEXP_COMM – unexpected communication ANSWER – called party answered

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response4 cr 0 resp invl close userid dallas dal220 bearcap speech closecse call_term $ $ $**

Close response is defined.

Use Procedure 4-15 to define an Collect_Information message.

Procedure 4-15
Define an Collect_Information response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Collect_Information`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a response (RESP) or conversation (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as COLLECT_INFORMATION.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.
Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD collinfo1 cr 0 cwp invl collect_information $ $ $**

Collect_Information response is defined.

Use Procedure 4-16 to define a Connect message.

Procedure 4-16
Define a Connect message

Note: Table IN1RESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table IN1RESP.

2 Define a **`Connect`** message by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as CONNECT.

5 Datafill the CARRIER refinement with an NOA of NATL or INTL, an IN/1 numbering plan, and 0–24 digits.

6 Datafill the ROUTENUM refinement with an NOA of NATL or INTL, an IN/1 numbering plan, and 0–24 digits.

7 Datafill the BILLIND refinement with an AMA Call Type (141 or 142) and a 3-digit Service Feature Identification Code.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD dal228cr cr 0 invl connect natl unk 888 natl isdn 2142281234 141 234 $**

Connect message is defined.

Use Procedure 4-17 to define a conversational Connect_To_Resource message.

Procedure 4-17
Define a conversational Connect_To_Resource message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a conversational **`Connect_To_Resource`** message by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as conversation with permission (CWP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as CONNECT_TO_RESOURCE.

6 Datafill resource refinements based on resource type (COLL_DIGITS or FLEX_PARM).

For COLL_DIGITS, where

RESOURCE is the resource type (COLL_DIGITS)
BLOCKTYPE is the block type (ANNDIG)
DIGS_TYPE defines the number of digits to collect (FIXED, UP_TO, VARIABLE, NORMAL)
NUM_DIGS 0–255. (Only used if DIGS_TYPE is FIXED or UP_TO).
BLK_FORMAT1 is the block format (UNINTER or INTER). Datafill the ANN_ID and INFO_DIGS refinements.
ANN_ID: 0–4095
INFO_DIGS: 0–12 digits.
BLK_FORMAT2 is the block format (INTER or NIL). Only used if BLK_FORMAT1 is UNINTER. Datafill the ANN_ID and INFO_DIGS refinements.
ANN_ID: 0–4095

INFO_DIGS: 0–12 digits.

CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined: (DISCFLAG, ANSWIND, DESTADDR, AMAMEAS, AMADIGS, LEGID)

FOR FLEX_PARM, where

RESOURCE is the resource type (FLEX_PARM)

BLOCKTYPE is the block type (FLEX)

FLEX_FORMAT is the FLEX format (FLEX_PARM_NT and FLEX_PARM_HEX)

FLEX_PARM_BLOCK is a multiple-entry vector of up to 112 octets

CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined: (DISCFLAG, ANSWIND, DESTADDR, AMAMEAS, AMADIGS, LEGID)

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response2 cr 0 cwp invl connect_to_resource coll_digs anndig fixed 5 inter 14 $ $ $ $**

Sample entry: **>ADD response3 cr 0 cwp invl connect_to_resource flex_parm flex flex_parm_hex 11 22 33 44 55 $ destaddr natl isdn 2142210000 amaeas ctrdssp $ $ $**

The conversational Connect_To_Resource message is defined.

Use Procedure 4-18 to define a non-conversational Connect_To_Resource response.

Procedure 4-18
Define a non-conversational Connect_To_Resource response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Connect_To_Resource`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a response (RESP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as CONNECT_TO_RESOURCE.

6 Datafill refinements RESOURCE, BLOCKTYPE, BLK_FORMAT, ANN_BLK, and CAIN_STR_PARM_VALUE.

RESOURCE is the resource type (PLAY_ANN)
BLOCKTYPE is the block type (ANN)
BLK_FORMAT is the block format (UNINTER).
ANN_BLK: identifies up to four announcements to play. The format for this field is: ann_id info_digs
Where,
ANN_ID is 0–4095
INFO_DIGS is 0–12 digits.
CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined. (DISCFLAG, ANSWIND, AMAMEAS, AMADIGS, LEGID)

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD ctresp cr 0 resp invl connect_to_resource play_ann ann uninter 324 101 $ discflag $ $ $**

A non-conversational Connect_To_Resource response is defined.

Use Procedure 4-19 to define a Continue message.

Procedure 4-19
Define a Continue message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Continue`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as response (RESP) or conversation with permission (CWP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as CONTINUE.

6 Datafill optional refinement CAIN_PARM_VALUE (parameter returned in the message). Valid value is AMADIGS. When you enter AMADIGS, datafill the AMA_DIGS_TYPE and DIGITS refinements.

AMA_DIGS_TYPE enter the digit type (PIN, ACCT, BILL, CIC, ORIGPRIVNUM, TERMPRIVNUM)
DIGITS enter up to 24 digits

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or Enter $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response3 cr 0 resp invl continue $ $ $**

Continue message is defined.

Use Procedure 4-20 to define a Disconnect response.

Procedure 4-20
Define a Disconnect response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Disconnect`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99)
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as response (RESP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as DISCONNECT.

6 Datafill optional refinement CAIN_PARM_VALUE (parameter returned in the message). Valid value is AMADIGS. When you enter AMADIGS, datafill the AMA_DIGS_TYPE and DIGITS refinements.

AMA_DIGS_TYPE enter the digit type (PIN, ACCT, BILL, CIC, ORIGPRIVNUM, TERMPRIVNUM)
DIGITS enter up to 24 digits

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response3 cr 0 resp invl disconnect $ $ $**

Disconnect response is defined.

Use Procedure 4-21 to define an Disconnect_Leg message.

Procedure 4-21
Define an Disconnect_Leg message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Disconnect_Leg`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a conversation with permission (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as DISCONNECT_LEG.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.
Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **> ADD DISC_LEG CR 0 RESP INVL DISCONNECT_LEG 0 $ $ $**

Disconnect_Leg message is defined.

Use Procedure 4-22 to define a Failure_Report response.

Procedure 4-22
Define a Failure_Report response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Failure_Report`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as response (RESP)

4 Datafill subfield COMPTYPE (TCAP component type) as component type RERR.

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as FAILURE_REPORT.

Note: When FAILURE_REPORT is selected for OPERTYPE, datafill the FAILURECAUSE refinement.

FAILURECAUSE is one of the following failure causes:
HIGH_RATE – rate too high
NO_RSRC – no resource
CHNLBUSY – channel busy
ABORT_F – abort
RSRC_LIM – resource limitation
APPL_ERR – application error
PROT_ERR – protocol error
TIMER_EXP – T1 timer expired
TEMP_FAIL – temporary failure
CTRL_ENC – control encountered
BAD_CODE – improper coding

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

7 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response5 cr 0 resp rerr failure_report appl_err $ $**

Failure_Report response is defined.

Use Procedure 4-23 to define a Furnish_AMA_Information message.

Procedure 4-23
Define a Furnish_AMA_Information message

Note: Non-call related components must be datafilled before the call-related components with which they will be included.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Furnish_AMA_Information`** message by using the following format:

>ADD respkey datatype comparea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
comparea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as FURNISH_AMA_INFO.

5 Datafill the AMABAFMD parameter.

AMABAFMD is an optional parameter. Enter one byte (2 hexadecimal characters) at a time. A maximum of 128 bytes and a minimum of 2 bytes can be entered for this parameter.

6 Datafill the AMAABIND parameter.

AMAABIND is an optional parameter. Enter of Y or N.

7 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

Sample entry: **>ADD furnama ncr invl furnish_ama_info amabafmd 2a cf 82 $ amaabind y $ $**

The Furnish_AMA_Information message is defined.

Use Procedure 4-24 to define an Merge_Call message.

Procedure 4-24
Define an Merge_Call message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Merge_Call`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a conversation with permission (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as MERGE_CALL.

6 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

7 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **> ADD MERGE_CALL CR 0 RESP INVL MERGE_CALL $ $**

Merge_Call message is defined.

Use Procedure 4–25 to define an Originate_Call message.

Procedure 4-25
Define an Originate_Call message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Originate_Call`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that may be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE

3 Datafill subfield PKGTYPE (package type) as a conversation with permission (CWP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as ORIGINATE_CALL.

6 Datafill the CAIN_PARM_VALUE refinement.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. Define up to 8 parameters.

Note: If neither a CLDNO nor a TRK (PRITRK, ALTTRK, or SALTTRK) with an OPULSNO are set for CAIN_PARM_VALUE, a fatal error occurs.

Note: Refer to *UCS DMS-250 Data Schema Reference Manual* for valid entries.

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **> ADD ORIG_CALL CR 0 CWP INVL ORIGINATE_CALL (PRITRK DALLAS DAL265 Y) (OPULSNO NATL ISDN 2142790000) $ $ (SHI_SHF) $**

Originate_Call message is defined.

Use Procedure 4-26 to define a Play_Announcement message.

Procedure 4-26
Define a Play_Announcement message

Note: Table IN1RESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table IN1RESP.

2 Define a **`Play_Announcement`** message by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as PLAY_ANN.

5 Datafill the STDANNC refinement. The STDANNC refinement specifies an index to CAINRSRC (0–255).

6 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD playa3 cr 0 invl play_ann 3 $**

Play_Announcement message is defined.

Use Procedure 4-27 to define a Report_Error response.

Procedure 4-27
Define a Report_Error response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Report_Error`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as UNI or RESP.

4 Datafill subfield COMPTYPE (TCAP component type) as invl last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as REPORT_ERROR.

Note: When REPORT_ERROR is selected for OPERTYPE, datafill the APPL_ERROR_STRING refinement.

APPL_ERROR_STRING is one of the following application errors:
DATA_ERR – data error
MISS_PARM – missing parameter
TI_EXPIRE – T1 timer expired
UNEXP_COMM – unexpected communication
UNEXP_MSG– unexpected message
UNXP_MSGSEQ – unexpected message sequence
UNXP_PRMSEQ – unexpected parameter sequence

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

7 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response6 cr 0 resp invl report_error unexp_comm $ $**

Report_Error response is defined.

Use Procedure 4-28 to define a Request_Report_BCM_Event message.

Procedure 4-28
Define a Request_Report_BCM_Event message

Note: Non-call related components must be datafilled before the call-related components with which they will be included.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Request_Report_BCM_Event`** message by using the following format:

>ADD respkey datatype comparea
where

respkey is the user-defined name used in table CAINRESP to determine the non-call related component (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
comparea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as REQ_REP_BCM.

5 Datafill CAIN_PARM_VALUE.

CAIN_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–5 parameters can be defined. (EDPREQ, EDPNOTIF, ONOANSWT, TNOANSWT, or TIMEOUTT)

- EDPREQ — is the vector of EDPs to be armed to send an EDP-Request to the SCP (NETBUSY, OCLDBUSY, ONOANSWR, TIMEOUT).
- EDPNOTIF — is the vector of EDPs to be armed to send an EDP-Notification to the SCP (OTERMSZ, OANSWR, ODISC).
- ONOANSWT — is the amount of time (in seconds) to wait before encountering the ***O_No_Answer*** EDP (1 to 120).
- TNOANSWT — is the amount of time (in seconds) to wait before encountering a ***T_No_Answer*** EDP (1 to 120).
- TIMEOUTT — is the amount of time (in minutes) to wait before encountering a *Timeout* event (1 to 180).

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

Sample entry: **>ADD rrbcme01 ncr invl req_rep_bcm edpreq netbusy ocldbusy onoanswr $ edpnotif otermsz oanswr $ onoanswt 120 tnoanswt 120 timeoutt 1 $**

The Request_Report_BCM_Event message is defined.

Use Procedure 4-29 to define a Send_Notification message.

Procedure 4-29
Define a Send_Notification message

Note: Non-call related components must be datafilled before the call-related components with which they will be included.

At the CI prompt

1 Enter table CAINRESP.

2 Define an **`Send_Notification`** message by using the following format:

>ADD respkey datatype comparea
where

respkey is the user-defined name used in table CAINRESP to determine the non-call related component (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter NCR.
comparea is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as SEND_NOTIFICATION.

5 Datafill the ECHODATA parameter.

ECHODATA is currently the only parameter that can be defined. The ECHODATA parameter is used to associate the SCP **`Send_Notification`** message with the **`Termination_Notification`** message.

6 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

Sample entry: **>ADD sendnot ncr invl send_notification echodata $ $ $**

The Send_Notification message is defined.

Use Procedure 4-30 to define a non-conversational Send_To_Resource response.

Procedure 4-30
Define a non-conversational Send_To_Resource response

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a **`Send_To_Resource`** response by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as a response (RESP).

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as SEND_TO_RESOURCE.

6 Datafill refinements RESOURCE, BLOCKTYPE, BLK_FORMAT, ANN_BLK, and CAIN_STR_PARM_VALUE.

RESOURCE is the resource type (PLAY_ANN)
BLOCKTYPE is the block type (ANN)
BLK_FORMAT is the block format (UNINTER).
ANN_BLK: identifies up to four announcements to play. The format for this field is: ann_id info_digs
Where,
ANN_ID is 0–4095
INFO_DIGS is 0–12 digits.
CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined. (DISCFLAG, ANSWIND, AMAMEAS, AMADIGS)

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response2 cr 0 resp invl send_to_resource play_ann ann uninter 324 101 $ discflag $ $ $**

A non-conversational Send_To_Resource response is defined.

Use Procedure 4-31 to define a conversational Send_To_Resource message.

Procedure 4-31
Define a conversational Send_To_Resource message

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a conversational **`Send_To_Resource`** message by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype is the type of data in the call response (CR or NCR). Enter CR.
delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).
pkgarea is comprised of three subfields: PKGTYPE, COMPTYPE, and OPERTYPE.

3 Datafill subfield PKGTYPE (package type) as conversation with permission (CWP)

4 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

5 Datafill subfield OPERTYPE (message type to be returned to the switch) as SEND_TO_RESOURCE.

6 Datafill resource refinements based on resource type (COLL_DIGITS or FLEX_PARM).

For COLL_DIGITS, where

RESOURCE is the resource type (COLL_DIGITS)
BLOCKTYPE is the block type (ANNDIG)
DIGS_TYPE defines the number of digits to collect (FIXED, UP_TO, VARIABLE, NORMAL)
NUM_DIGS 0–255. (Only used if DIGS_TYPE is FIXED or UP_TO).
BLK_FORMAT1 is the block format (UNINTER or INTER). Datafill the ANN_ID and INFO_DIGS refinements.
ANN_ID: 0–4095
INFO_DIGS: 0–12 digits.
BLK_FORMAT2 is the block format (INTER or NIL). Only used if BLK_FORMAT1 is UNINTER. Datafill the ANN_ID and INFO_DIGS refinements.
ANN_ID: 0–4095
INFO_DIGS: 0–12 digits.

CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined: (DISCFLAG, ANSWIND, DESTADDR, AMAMEAS, AMADIGS)

For FLEX_PARM, where

RESOURCE is the resource type (FLEX_PARM)
BLOCKTYPE is the block type (FLEX)
FLEX_FORMAT is the FLEX format (FLEX_PARM_NT and FLEX_PARM_HEX)
FLEX_PARM_BLOCK is a multiple-entry vector of up to 112 octets
CAIN_STR_PARM_VALUE is a multiple-entry vector that contains the parameters returned to the switch. 0–4 parameters can be defined: (DISCFLAG, ANSWIND, DESTADDR, AMAMEAS, AMADIGS)

7 Datafill the EXTPARM refinement.

EXTPARM is a vector that indexes outgoing extension parameters datafilled in table CAINREXT. Enter an index into table CAINREXT or $.

8 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response2 cr 0 cwp invl send_to_resource coll_digs anndig fixed 5 inter 14 $ $ $ $**

Sample entry: **>ADD response3 cr 0 cwp invl send_to_resource flex_parm flex flex_parm_hex 11 22 33 44 55 $ destaddr natl isdn 2142210000 amaeas ctrdssp $ $ $**

The conversational Send_To_Resource message is defined.

Use Procedure 4-32 to define a Termination message.

Procedure 4-32
Define a Termination message

Note: Table IN1RESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table IN1RESP.

2 Define a **`Termination`** message by using the following format:

>ADD respkey datatype pkgarea
where

respkey	is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).
datatype	is the type of data in the call response (CR or NCR). Enter NCR.
pkgarea	is comprised of two subfields: COMPTYPE and OPERTYPE.

3 Datafill subfield COMPTYPE (TCAP component type) as invoke last (INVL).

4 Datafill subfield OPERTYPE (message type to be returned to the switch) as TERMINATION.

5 Datafill the ECHODATA refinement with a1 to 8 digit identifier that correlates the termination information with the request for termination information.

Sample entry: **>ADD TFTERM NCR INVL TERMINATION 00000256 $**

Termination message is defined.

Use Procedure 4-33 to define an Abort package.

Procedure 4-33
Define an Abort package

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define an Abort package by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).

datatype is the type of data in the call response (CR or NCR). Enter CR.

delay is a number in seconds that can be used to simulate network or SCP congestion (0–99).

pkgarea is comprised of one subfields: PKGTYPE, where

PKGTYPE is the package type (ABORT_P).

Note: When ABORT_P is selected for PKGTYPE, datafill the CAUSE refinement.

CAUSE is one of the following abort causes:
NIL_ABORT
UNRECOGNIZED_PACKAGE_TYPE
INCORRECT_TRANSACTION_PORTION
BADLY_STRUCTURED_TRANSACTION
UNRECOGNIZED_TRANSACTION_ID
PERMISSION_TO_RELEASE_PROBLEM
RESOURCE_UNAVAIL
MAX_ABORT_CAUSE

3 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

4 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response7 cr 0 abort_p unrecognized_package_type $**

The abort package is defined.

Use Procedure 4-34 to define a Response package with a reject component.

Procedure 4-34
Define a Response package with a reject component

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINRESP.

2 Define a Response package with a reject component by using the following format:

>ADD respkey datatype delay pkgarea
where

respkey is the user-defined name used in table CAINMTCH to determine the response (up to 16 alphanumeric characters).

datatype is the type of data in the call response (CR or NCR). Enter CR.

delay is a number in seconds that can be used to simulate network or SCP congestion (0–99)

pkgarea is a multiple-entry field comprised of two subfields: PKGTYPE and COMPTYPE, where

PKGTYPE	is the package type (RESP).
COMPTYPE	is the TCAP component type (REJ).

Note: When REJ is selected for COMPTYPE, datafill the FAMILY and PROBLEM refinements.

FAMILY	is the problem family (GENERAL_PROBLEM)
PROBLEM	is one of the following problems: NIL_GENERAL_PROB UNRECOGNIZED_COMPONENT INCORRECT_COMPONENT_PORTION BAD_COMPONENT_STRUCTURE MAX_GENERAL_PROB

3 Datafill optional refinement EXTPARM with a valid entry into table CAINREXT. Enter an index into table CAINREXT or $.

4 Datafill the NCRCOMPS refinement.

NCRCOMPS is a vector of indexes. Enter the index of one or more non-call related tuples or $.

Sample entry: **>ADD response8 cr 0 resp rej general_problem bad_component_structure $ $**

The Response package is defined.

Use Procedure 4-35 to define the optional parameter matching criteria.

Procedure 4-35
Define optional parameter matching criteria

Note: Table CAINRESP must be datafilled before table CAINMTCH.

At the CI prompt

1 Enter table CAINMTCH.

2 Define optional parameter matching criteria by using the following format:

>ADD key cainparm mode

where

key is an integer value corresponding the the PMI field in table CAINKEY.

cainparm is a multiple-entry vector that contains the optional parameter matching criteria. Up to 10 parameters can be defined. The following table lists the values and data required for each parameter.

mode identifies the action the SCP simulator should take when a match is made.

3 Datafill the MODE subfield. Enter the action the SCP simulator should take when a match is made. Enter RESP, CONV, or EDPCONV.

RESP identifies that a response should be formed. Datafill the ACTIDX refinement.

where

actidx is a valid RESPKEY value in table CAINRESP the SCP simulator will use to form a response.

CONV identifies that a conversation should be started. Datafill the RESPIDX, CONV_IDX, and ERR_ACT refinements.

where

respidx is a multiple-entry vector that contains the RESPKEY in table CAINRESP the SCP simulator should use to form a response, the timeout value for the response timeout (in seconds), and a vector of up to 5 clear causes allowed. (Refer to *UCS DMS-250 Data Schema Reference Manual* for specific values.)

conv_idx is an index into table CAINCONV. This number must be entered as a single number, or as two numbers separated by an underscore.

err_act is a valid key into table CAINRESP to use when an error occurs.

EDPCONV identifies that an EDP conversation should be started. Datafill the ACTIDX and TIMEOUT refinements.

where

actidx is the RESPKEY in table CAINRESP the SCP simulator should use to form a response.

timeout is the value for the response timeout (in minutes).

Sample entry: **>ADD 99 clgno natl isdn 5851111 colldigs pin prvt 09878 $ resp response1**

Sample entry: **>ADD 100 clgno natl isdn 5851212 $ conv conv1 30 $ $ 100 $ disconnect**

Sample entry: **>ADD 101 $ edpconv response3 30**

Optional matching parameter criteria is defined.

Use Procedure 4-36 to define digit collection validation.

Procedure 4-36
Define digit collection validation

1 Enter table CAINCONV.

2 Define digit collection validation using the following format:

>ADD key criteria respidx
where

key is a number that provides the key into the table. There is an index into table CAINCONV provisioned in table CAINMTCH whenever a conversational scenario is datafilled.

criteria is a multiple-entry vector that contains the digit collection matching criteria. Up to five vectors of up to 24 HEX characters (0–9, A–F, *, #) can be defined.

respidx is the RESPKEY in table CAINRESP the SCP simulator should use to form a response.

Sample entry: **>ADD 205 12345 $ response3**

Sample entry: **>ADD 1481 AA1234CF 231435 $ response3**

Digit collection validation is defined.

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

Procedures 4-44 through 4-47 provide steps and sample datafill to set up the SCP simulator and the CAINTEST tool for performing an SCP simulator and switch interaction test with the Cancel_Resource_Event message.

These procedures are as follows:

- SCP simulator setup, Procedure 4-44
- CAINTEST setup, Procedure 4-45
- Enable log generation, Procedure 4-46
- Send the CAINTEST query, Procedure 4-47

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

Use Procedure 4-44 to set up the SCP simulator.

Procedure 4-44
SCP simulator setup

1 Table CAINUID datafill. Maps the name used by the simulator to the actual switch identifier (11) and trunk number (221).

Incoming agent:

>ADD DALLAS DAL221 11 221

Outgoing agent:

>ADD DALLAS DAL220 11 220

2 Table CAINKEY datafill. Identifies the message (**`Info_Analyzed`**), the user identification, trigger criteria for a first pass match. Once a match is made on this data, identifies an index into table CAINMTCH.

>ADD INFO_ANALYZED TRK 0221 CUST_INT (202) $

3 Table CAINRESP datafill. Defines the **`Cancel_Resource_Event`** message sent to the switch and identifies the **`Send_To_Resource`** message.

`Cancel_Resource_Event` datafill:

>ADD CANCEL_RES_EVENT CR 0 CWP INVL CANCEL_RESOURCE CONV5 $ $

`Send_To_Resource` datafill:

>ADD CONV5 CR 0 CWP INVL SEND_TO_RESOURCE COLL_DIGITS ANNDIG FIXED 5 INTER (14 $) $ $ $ $

4 Table CAINRESP datafill. Defines the **`Disconnect`** message sent to the switch when an error occurs during conversation.

>ADD DISCONNECT CR 0 RESP INVL DISCONNECT $ $ $

`Analyze_Route` datafill:

>ADD VALIDADDR01 CR 0 RESP INVL ANALYZE_ROUTE (PRITRK DALLAS DAL220 N) $ $

5 Table CAINMTCH datafill. Indexed from table CAINKEY, identifies the matching parameters for comparison against the query and defines the SCP action (send a response or initiate a conversation).

>ADD 202 (CLDNO NATL ISDN 2142200005) $ CONV (CANCEL_RES_EVENT 30 (RSRCXLD) $) $ (205) $ DISCONNECT

6 Table CAINCONV datafill. Identifies the digits to match against digits collected by the switch and the response.

>ADD 205 (12345) $ VALIDADDR01

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

Use Procedure 4-45 to set up the CAINTEST tool.

Procedure 4-45
CAINTEST setup

Note: Specify whether the test message is to be sent to the SCP or the SCP simulator by datafilling table C7GTT appropriately. Volume 1, Chapter 2, "Provisioning CAIN," for more information on CCS7 setup. The query built using CAINTEST does not go through switch call processing logic. The query or conversation message is sent through the CAIN SS7 subsystem to the CAIN simulator tables. For this example, assume a trigger at CUSTDP. CAINTEST does not examine trigger tables.

1 Set up the CAINTEST messages (queries and conversations).

>caintest
>setappl cain02
>timeout 30
>setfam dms250
>setquery 1
>settrans tcap_sccp cain_addr_gt 214
>setmsg info_analyzed
>setparm std userid trk 221
>setparm std bearcap speech
>setparm std trigcrit cust_int
>setparm std cldno natl 2142200005
>setparm ext adin 1
>setparm ext swid 11
>showflds

Example of a MAP response:

```
       QUERY     1

APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    214
TIMEOUT         : 30    sec
FAMILY          : DMS250
MESSAGE         : info_analyzed
     USERID     : TRK  221
     BEARCAP    : SPEECH
     CLDNO      : NATL          2142200005
     TRIGCRIT   : CUST_INT
     ADIN       : 01
     SWID       : 11
```

SCP simulator and CAINTEST interaction
Cancel_Resource_Event (continued)

Use Procedure 4-46 to enable log generation.

Procedure 4-46
Enable log generation

1 Table CAINPARM. Enable the simulator logs (CAIN900 and CAIN901) in parameter CAIN900_LOGS_ENABLED.

>REP CAIN900_LOGS_ENABLED 900 901

2 Enable VPTRACE for VAMP901 and VAMP902 log generation.

>VPTRACE ENABLE

```
250X     VAMP902 SEP09 16:49:46 4000 INFO VAMP OUTBOUND
MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 15
        ROUTE   : GTTYPE=CAIN_CLID_GT, GTDIGS=214
        PACKAGE : QWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 01
        OPCODE  : INFO_ANALYZED
          *** PARAMETERS ***
          USERID  : TRK  221
          BEARCAP : SPEECH
          CLDNO   : NOA = NATL          NUMPLAN = ISDN
DIGITS = 2142200005
          TRIGCRIT: CUST_INT
            *** Extension Parameter ***
            NT DMS250 EXTENSION PARAMETER FAMILY
            SWID    : 011
            ADIN    : 1
        ------------- hex dump of message -------------
        E2 42 C7 04 00 00 15 00 E8 3A E9 38 CF 01 01 D1
        02 64 03 30 2F BF 35 04 85 02 00 DD 8D 01 00 8F
        07 03 10 12 24 02 00 50 9F 34 01 03 BF 54 15 06
        09 2A 86 48 86 F6 7D 08 41 10 31 08 9F 29 02 10
        01 8D 01 01
```

3 Access the LOGUTIL command set and issue the START command.

>LOGUTIL
>START
>QUIT

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

Use Procedure 4-47 to send the CAINTEST query. Either the switch or the SCP simulator performs all Steps except Steps 1, 10, and 11. The log generation steps require your verification.

Procedure 4-47
Send the CAINTEST query

1 Send the query.

Note: The query built using CAINTEST does not go through switch call processing logic. The query or conversation message is sent through the CAIN SS7 subsystem to the CAIN simulator tables. Refer to Volume 1, Chapter 2, "Provisioning CAIN," for more information on CCS7 setup. For this example, assume a trigger at CUSTDP. CAINTEST does not examine trigger tables.

>send query 1

Example of a MAP response:

```
Message sent.
```

2 The switch generates a VAMP902 log.

Example of a MAP response:

```
250Z     VAMP902 APR02 15:57:23 6900 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 03 00 2E
        ROUTE   : GTTYPE=CAIN_ADDR_GT, GTDIGS=214
        PACKAGE : QWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 01
        OPCODE  : INFO_ANALYZED
          *** PARAMETERS ***
          USERID  : CAIN  SWID = 011  TRKGRP = 00221
          BEARCAP : SPEECH
          CLDNO   : NOA = NATL     NUMPLAN = ISDN     DIGITS = 2142200005
          TRIGCRIT: CUST_INT
            *** Extension Parameter ***
            NT DMS250 EXTENSION PARAMETER FAMILY
            ADIN    : 1
        ------------- hex dump of message -------------
        E2 40 C7 04 03 00 2E 00 E8 38 E9 36 CF 01 01 D1
        02 64 03 30 2D BF 35 07 81 05 00 10 01 20 12 8D
        01 00 8F 07 03 10 12 24 02 00 50 9F 34 01 03 BF
        54 10 06 09 2A 86 48 86 F6 7D 08 41 10 31 03 8D
        01 01
```

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

3 The simulator checks table CAINKEY to determine if the query matches the datafill.

INFO_ANALYZED TRK 0221 CUST_INT (202) $

If the message, USERID, and TRIGCRIT match, the simulator indexes table CAINMTCH and looks for a match in key 202.

4 The simulator enters table CAINMTCH to determine if the query parameters match the datafill. The dialed address was 2142200005 and a match is found at tuple 202.

202 (CLDNO NATL ISDN 2142200005) $ CONV (CANCEL_RES_EVENT 30 (RSRCXLD) $) $ (205) $ DISCONNECT

Tuple 202 indicates that the simulator should initiate conversation with the switch by indexing table CAINRESP on the CANCEL_RES_EVENT tuple. Tuple 202 also indicates that tuple 205 of table CAINCONV is indexed when the switch returns the collected digits. If any error occurs during these transactions, the simulator should index table CAINRESP on the DISCONNECT tuple and return the message defined there.

Note: If the simulator does not find a match in table CAINMTCH, an **`Application_Error`** is returned to the switch.

5 The switch generates a CAIN900 log to show SCP simulator matching information.

Example of a MAP response:

```
250X     CAIN900 SEP09 16:49:46 4100 INFO SCP SIMULATOR INDICES
          TRANSID = 0000 0015
          CAINKEY = INFO_ANALYZED
                    TRK  0221
                    CUST_INT
         CAINMTCH = 202
```

6 The simulator enters table CAINRESP and identifies the **`Send_To_Resource`** message identified by the CANCEL_RES_EVENT tuple.

CANCEL_RES_EVENT CR 0 CWP INVL CANCEL_RESOURCE CONV5 $

The simulator indexes the **`Send_To_Resource`** message defined in the CONV5 tuple.

CONV5 CR 0 CWP INVL SEND_TO_RESOURCE COLL_DIGITS ANNDIG FIXED 5 INTER (14 $) $ $ $ $

The simulator sends a conversational **`Send_To_Resource`** message to the switch, requesting the switch to play an interruptible announcement (14) and to collect 5 digits.

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

7 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP901 SEP09 16:49:46 4200 INFO VAMP INBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 15
        ROUTE   : PC=01-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 0E 01
        OPCODE  : SEND_TO_RESOURCE
          *** PARAMETERS ***
          RSRCTYPE: COLL_DIGITS
          STRPARM : DIGS_BLK:
                      MAXDIGS = FIXED: 5
                      INTER   :
                        ANNC_ID =   14  DIGITS = NO DIGITS
        ------------- hex dump of message -------------
        E5 2B C7 08 00 00 05 00 00 00 15 00 E8 1F E9 1D
        CF 02 0E 01 D1 02 66 01 30 13 9F 2D 01 01 BF 32
        0C A1 0A 80 01 05 A2 05 04 03 00 0E 00
```

8 CAINTEST receives the **`Send_To_Resource`** message.

Example of a MAP response:

```
TIME --> Received response in 0.18 sec
Conversation mode between SSP and SCP.
SEND_TO_RESOURCE received
     RSRCTYPE   : COLL_DIGITS
     STRParm    : Action:DIGS_BLK  Size:Fixed Number:5
          :INTER        1 14
Response 1 is not datafilled.
You must manually build the response message.
Please enter in parms for a response message.
```

9 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP901 SEP09 16:49:46 4300 INFO VAMP INBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 15
        ROUTE   : PC=01-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 0F
        OPCODE  : CANCEL_RESOURCE_EVENT
          *** PARAMETERS ***
        ------------- hex dump of message -------------
        E5 17 C7 08 00 00 05 00 00 00 15 00 E8 0B E9 09
        CF 01 0F D1 02 66 03 30 00
```

SCP simulator and CAINTEST interaction Cancel_Resource_Event (continued)

10 Build the conversational response.

>settrans tcap_sccp cain_clid_gt 214
>setmsg resource_clear
>setparm std clearcse ResourceCancelled
>showflds
Example of a MAP response:

```
       RESPONSE     0
APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    214
TIMEOUT         : 30    sec
FAMILY          : DMS250
MESSAGE         : resource_clear
     CLEARCSE   : RESOURCECANCELLED
```

11 Send the response.

>send

```
Message sent, waiting for reply.
```

12 CAINTEST receives the **`Cancel_Resource_Event`** message.

Example of a MAP response:

```
TIME --> Received response in -24575.1015 sec
CANCEL RESOURCE received
```

13 The switch generates a VAMP902 log.

Example of a MAP response:

```
250X     VAMP902 SEP09 16:49:54 4800 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 15
        ROUTE   : PC=01-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 03 0E
        OPCODE  : RESOURCE_CLEAR
          *** PARAMETERS ***
          CLEARCSE: RESOURCECANCELL
        ------------- hex dump of message -------------
        E5 1B C7 08 00 00 15 00 00 00 05 00 E8 0F E9 0D
        CF 02 03 0E D1 02 66 02 30 03 95 01 03
```

SCP simulator and CAINTEST interaction Cancel_Resource_Event (end)

14 A **Resource_Clear** was received with a value other than normal.

202 (CLDNO UNK UNK 2142200005) $ CONV (CANCEL_RES_EVENT 30 (RSRCXLD) $) $ (205) $ DISCONNECT

15 The simulator enters table CAINRESP and builds the appropriate response.

DISCONNECT CR 0 RESP INVL DISCONNECT $ $ $

16 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP901 SEP09 16:49:54 5000 INFO VAMP INBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 15
        ROUTE   : PC=01-09-01, SSN=33
        PACKAGE : RESP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 10 03
        OPCODE  : DISCONNECT
          *** PARAMETERS ***
        ------------- hex dump of message -------------
        E4 14 C7 04 00 00 15 00 E8 0C E9 0A CF 02 10 03
        D1 02 65 03 30 00
```

17 CAINTEST reports status.

Example of a MAP response:

```
TIME --> Received response in -24575.1015 sec
CANCEL RESOURCE received
TIME --> Received response in 0.4 sec
DISCONNECT received
Message from the SCP to discontinue the call.
Mailbox deallocated.
```

SCP simulator and CAINTEST interaction EDP conversation (continued)

Procedures 4-48 through 4-51 provide steps and sample datafill to set up the SCP simulator and the CAINTEST tool for performing an SCP simulator and switch interaction test with Event Detection Point (EDP) conversation messages.

These procedures are as follows:

- SCP simulator setup, Procedure 4-48
- CAINTEST setup, Procedure 4-49
- Enable log generation, Procedure 4-50
- Send the CAINTEST query, Procedure 4-51

SCP simulator and CAINTEST interaction EDP conversation (continued)

Use Procedure 4-48 to set up the SCP simulator.

Procedure 4-48
SCP simulator setup

1 Table CAINUID datafill. Maps the name used by the simulator to the actual switch identifier (11) and trunk numbers (273) and (220).

Incoming agent:

DALLAS DAL273 11 273

Outgoing agent:

DALLAS DAL220 11 220

2 Table CAINKEY datafill. Identifies the message (**`Info_Analyzed`**), the user identification, trigger criteria for a first pass match. Once a match is made on this data, identifies an index into table CAINMTCH.

INFO_ANALYZED TRK 0273 CUST_INT (71) $

3 Table CAINRESP datafill. Defines the **`Analyze_Route`** message in the call-related component with a **`Request_Report_BCM_Event`** message in the non-call related component of a conversation package sent to the switch.

RRBCME01 NCR INVL REQ_REP_BCM (EDPREQ (NETBUSY) (OCLDBUSY) (ONOANSWR) $) (EDPNOTIF (OTERMSZ) (OANSWR) $) (ONOANSWT 120) (TNOANSWT 120) (TIMEOUTT 1) $ $

ROUTEARM CR 0 CWP INVL ANALYZE_ROUTE (PRITRK DALLAS DAL220 N) (CHGNO AUTH PRVT 6113311) (CLDNO NATL ISDN 2201234) $ $ (RRBCME01) $

4 Table CAINMTCH datafill. Indexed from table CAINKEY, identifies the matching parameters for comparison against the query and defines the SCP action (send a response or initiate a conversation).

71 (CLDNO NATL ISDN 2731200) $ EDPCONV ROUTEARM 12

5 Table CAINKEY datafill. Defines the EDP-Request (***Network_Busy***), ***`UserID`***, and EDPREQ to match EDP-Request for a first pass match. Once a match is made on this data, identifies an index into table CAINMTCH.

NETWORK_BUSY TRK 0273 EDPREQ (21) (22) (23) (24) (25) (26) (27) (29) $

6 Table CAINRESP datafill. Defines the **`Disconnect`** message sent to the switch.

DISCONNECT01 CR 0 RESP INVL DISCONNECT $ $ $

SCP simulator and CAINTEST interaction EDP conversation (continued)

7 Table CAINMTCH datafill. Index from table CAINKEY, identifies the matching parameters for comparison against the query and the SCP action (send a response or initiate a conversation).

21 $ RESP DISCONNECT01

SCP simulator and CAINTEST interaction EDP conversation (continued)

Use Procedure 4-49 to set up the CAINTEST tool.

Procedure 4-49
CAINTEST setup

Note: Specify whether the test message is to be sent to the SCP or the SCP simulator by datafilling table C7GTT appropriately. Refer to Volume 1, Chapter 2, “Provisioning CAIN,” for more information on CCS7 setup. The query built using CAINTEST does not go through switch call processing logic. The query or conversation message is sent through the CAIN SS7 subsystem to the CAIN simulator tables. For this example, assume a trigger at CUSTDP. CAINTEST does not examine trigger tables. This example may not represent a scenario possible with call processing logic.

SCP simulator and CAINTEST interaction
EDP conversation (continued)

1 Set up the CAINTEST messages (queries and conversations).

>caintest
CAINTEST:
>setappl cain02
>settrans tcap_sccp cain_addr_gt 801
>setfam dms250
>setmsg info_analyzed
>timeout 5
>setparm std userid trk 273
>setparm std bearcap speech
>setparm std trigcrit cust_int
>setparm std cldno natl 2731200
>setparm ext swid 11
>
>setresp 1
>settrans tcap_sccp cain_addr_gt 801
>setmsg o_term_seized
>setparm std userid trk 273
>setparm std bearcap speech
>setparm std notifind y
>setparm ext termtrk dal dal220twdtgs 11
>
>setresp 2
>settrans tcap_sccp cain_addr_gt 801
>setmsg o_answer
>setparm std userid trk 273
>setparm std bearcap speech
>setparm std notifind y
>
>setresp 3
>settrans tcap_sccp cain_addr_gt 801
>setmsg Network_Busy
>timeout 5
>setparm std userid trk 273
>setparm std bearcap speech
>setparm std notifind n
>showflds all

Example of a MAP response:

```
       QUERY     1

APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    801
TIMEOUT         : 5     sec
```

SCP simulator and CAINTEST interaction
EDP conversation (continued)

```
FAMILY          : DMS250
MESSAGE         : info_analyzed
     USERID     : TRK  273
     BEARCAP    : SPEECH
     CLDNO      : NATL          2731200
     TRIGCRIT   : CUST_INT
     SWID       : 11


       QUERY     2

APPLICATION     : CAIN02
TIMEOUT         : 30    sec
FAMILY          : DMS250


       QUERY     3

APPLICATION     : CAIN02
TIMEOUT         : 30    sec
FAMILY          : DMS250


       RESPONSE     1

APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    801
TIMEOUT         : 30    sec
FAMILY          : DMS250
MESSAGE         : o_term_seized
     USERID     : TRK  273
     BEARCAP    : SPEECH
     NOTIFIND   : Y
     TERMTRK    : DAL    DAL220TWDTGS      11


       RESPONSE     2

APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    801
TIMEOUT         : 30    sec
FAMILY          : DMS250
MESSAGE         : o_answer
     USERID     : TRK  273
     BEARCAP    : SPEECH
     NOTIFIND   : Y


       RESPONSE     3

APPLICATION     : CAIN02
TRANSPORT       : TCAP_SCCP     CAIN_ADDR_GT    801
TIMEOUT         : 5     sec
FAMILY          : DMS250
MESSAGE         : network_busy
     USERID     : TRK  273
     BEARCAP    : SPEECH
```

SCP simulator and CAINTEST interaction
EDP conversation (continued)

```
                    NOTIFIND   : N
               >
```

SCP simulator and CAINTEST interaction
EDP conversation (continued)

Use Procedure 4-50 to enable log generation.

Procedure 4-50
Enable log generation

1 Table CAINPARM. Enable the simulator logs (CAIN900 and CAIN901) in parameter CAIN900_LOGS_ENABLED.

>REP CAIN900_LOGS_ENABLED 900 901

2 Enable VPTRACE for VAMP901 and VAMP902 log generation.

>VPTRACE ENABLE

3 Access the LOGUTIL command set and issue the START command.

>LOGUTIL
>START
>QUIT

SCP simulator and CAINTEST interaction
EDP conversation (continued)

Use Procedure 4-51 to send the CAINTEST query. Either the switch or the SCP simulator performs all steps, except Steps 1, 9, 11, and 13. The log generation steps require your verification.

Procedure 4-51
Send the CAINTEST query

1 Send the query.

Note: The query built using CAINTEST does not go through switch call processing logic. The query or conversation message is sent through the CAIN SS7 subsystem to the CAIN simulator tables. Refer to Volume 1, Chapter 2, "Provisioning CAIN," for more information on CCS7 setup. For this example, assume a trigger at CUSTDP. CAINTEST does not examine trigger tables.

>send query 1

Example of a MAP response:

```
Message sent.
```

2 The switch generates a VAMP902 log.

Example of a MAP response:

```
250X     VAMP902 SEP09 17:12:25 1600 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : GTTYPE=CAIN_ADDR_GT, GTDIGS=801
        PACKAGE : QWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 01
        OPCODE  : INFO_ANALYZED
          *** PARAMETERS ***
          USERID  : TRK  273
          BEARCAP : SPEECH
          CLDNO   : NOA = NATL        NUMPLAN = ISDN     DIGITS = 2731200
          TRIGCRIT: CUST_INT
            *** Extension Parameter ***
            NT DMS250 EXTENSION PARAMETER FAMILY
            SWID    : 011
        ------------- hex dump of message -------------
        E2 3E C7 04 00 00 17 00 E8 36 E9 34 CF 01 01 D1
        02 64 03 30 2B BF 35 04 85 02 01 11 8D 01 00 8F
        06 83 10 72 13 02 00 9F 34 01 03 BF 54 12 06 09
        2A 86 48 86 F6 7D 08 41 10 31 05 9F 29 02 10 01
```

3 The simulator checks table CAINKEY to determine if the query matches the datafill.

INFO_ANALYZED TRK 0273 CUST_INT (71) $

SCP simulator and CAINTEST interaction EDP conversation (continued)

If the message, USERID, and TRIGCRIT match, the simulator indexes table CAINMTCH and looks for a match in key 71.

4 The simulator enters table CAINMTCH to determine if the query parameters match the datafill. The dialed address was 2731200 and a match is found at tuple 71.

71 (CLDNO NATL ISDN 2731200) $ EDPCONV ROUTEARM 12

Tuple 71 indicates that the simulator should initiate EDP conversation with the switch by indexing table CAINRESP on the ROUTEARM tuple.

5 The switch generates a CAIN900 log to show SCP simulator matching information.

Example of a MAP response:

```
250X     CAIN900 SEP09 17:12:25 4600 INFO SCP SIMULATOR INDICES
          TRANSID = 0000 0017
          CAINKEY = INFO_ANALYZED
                    TRK  0273
                    CUST_INT
         CAINMTCH = 71
```

6 The simulator enters table CAINRESP and identifies the **`Analyze_Route`** message with **`Request_Report_BCM_Event`** component identified by the ROUTEARM tuple.

RRBCME01 NCR INVL REQ_REP_BCM (EDPREQ (NETBUSY) (OCLDBUSY) (ONOANSWR) $) (EDPNOTIF (OTERMSZ) (OANSWR) $) (ONOANSWT 120) (TNOANSWT 120) (TIMEOUTT 1) $ $

ROUTEARM CR 0 CWP INVL ANALYZE_ROUTE (PRITRK DALLAS DAL220 N) (CHGNO AUTH PRVT 6113311) (CLDNO NATL ISDN 2201234) $ $ (RRBCME01) $

SCP simulator and CAINTEST interaction EDP conversation (continued)

7 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP901 SEP09 17:12:25 4700 INFO VAMP INBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : PC=00-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 14 01
        OPCODE  : ANALYZE_ROUTE
          *** PARAMETERS ***
          CHGNO   : NOA = AUTH     NUMPLAN = PRIV     DIGITS = 6113311
          CLDNO   : NOA = NATL     NUMPLAN = ISDN     DIGITS = 2201234
          PRITRK  : OPULSNO = N    SWID = 011         TRKGRP = 00220
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 15
        OPCODE  : REQUEST_REPORT_BCM
          *** PARAMETERS ***
          EDPREQ  : OCLDBUSY  ONOANSWR  NETBUSY
          EDPNOTIF: OTERMSZ   OANSWR
          ONOANSWT: 120
          TNOANSWT: 120
          TIMEOUTT: 1
        ------------- hex dump of message -------------
        E5 53 C7 08 00 00 07 00 00 00 17 00 E8 47 E9 22
        CF 02 14 01 D1 02 65 01 30 18 93 06 E4 50 16 31
        13 01 8F 06 83 10 22 10 32 04 9F 2A 05 80 10 01
        20 02 E9 21 CF 01 15 D1 02 6D 01 30 18 9F 5C 03
        07 C0 80 9F 5D 02 04 30 9F 5B 01 78 9F 63 01 78
        9F 81 0B 01 01
```

SCP simulator and CAINTEST interaction EDP conversation (continued)

8 CAINTEST receives the **Analyze_Route** and **Request_Report_BCM_Event** messages.

Example of a MAP response:

```
TIME --> Received response in 0.12 sec
Conversation mode between SSP and SCP.
ANALYZE_ROUTE received
     CHGNO      : AUTH          6113311
     CLDNO      : NATL          2201234
     PRITRK     : opls=N swid=11 trkgrp=220
Request_Report_BCM_Event received.
     EDPREQ     : ONOANSWR  OCLDBUSY  NETBUSY
     EDPNOTIF   : OANSWR    OTERMSZ
     ONOANSWT   : 120
     TIMEOUTT   : 1
EDPs armed, EDP conversation started.
Valid responses are EDP-Request and EDP-Notification
messages armed by EDPREQ and EDPNOTIF parameters of the
REQUEST_REPORT_BCM_EVENT message respectively.
Close messages can also be sent to indicate EDP is completed.
The response order is not present.
You must manually build the response message.
Please enter in parms for a response message.
```

9 Send the response **O_Term_Seized** EDP-Notification message from CAINTEST.

>send response 1

Example of a MAP response:

```
Message sent.
No reply expected
```

SCP simulator and CAINTEST interaction
EDP conversation (continued)

10 The switch generates a VAMP902 log and a CAIN901 log.

Example of a MAP response:

```
250X     VAMP902 SEP09 17:12:33 5500 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : PC=00-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 03
        OPCODE  : O_TERM_SEIZED
          *** PARAMETERS ***
          USERID  : TRK  273
          BEARCAP : SPEECH
          NOTIFIND: YES
            *** Extension Parameter ***
            NT DMS250 EXTENSION PARAMETER FAMILY
            TERMTRK : TRKTYPE = DAL          TRKGRP =  220  TRKMEM = 11
        ------------- hex dump of message -------------
        E5 42 C7 08 00 00 17 00 00 00 07 00 E8 36 E9 34
        CF 01 03 D1 02 64 0C 30 2B BF 35 04 85 02 01 11
        8D 01 00 9F 6F 01 01 BF 54 1A 06 09 2A 86 48 86
        F6 7D 08 41 10 31 0D B1 0B 80 01 00 81 02 00 DC
        82 02 00 0B


250X     CAIN901 SEP09 17:12:33 5600 INFO SCP SIMULATOR ACTION
         TCAP ORIG TRANS ID = 0000 0017
         TCAP RESP TRANS ID = 0000 0007

         SCP RECEIVED = TCAP_OK
         ERROR CAUSE  = EDP_NOTIFICATION_RECEIVED
```

11 Send the response **`O_Answer`** EDP-Notification message from CAINTEST.

>send response 2

Example of a MAP response:

```
Message sent.
No reply expected.
```

SCP simulator and CAINTEST interaction EDP conversation (continued)

12 The switch generates a VAMP902 log and a CAIN901 log.

Example of a MAP response:

```
250X     VAMP902 SEP09 17:12:39 6100 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : PC=00-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 04
        OPCODE  : O_ANSWER
          *** PARAMETERS ***
          USERID  : TRK  273
          BEARCAP : SPEECH
          NOTIFIND: YES
        ------------- hex dump of message -------------
        E5 25 C7 08 00 00 17 00 00 00 07 00 E8 19 E9 17
        CF 01 04 D1 02 64 0B 30 0E BF 35 04 85 02 01 11
        8D 01 00 9F 6F 01 01

250X     CAIN901 SEP09 17:12:39 6200 INFO SCP SIMULATOR ACTION
         TCAP ORIG TRANS ID = 0000 0017
         TCAP RESP TRANS ID = 0000 0007

         SCP RECEIVED = TCAP_OK
         ERROR CAUSE  = EDP_NOTIFICATION_RECEIVED
```

13 Send the response **`Network_Busy`** EDP-Request message from CAINTEST.

>send response 3

Example of a MAP response:

```
Message sent.
EDPs disarmed, waiting for a reply.
```

SCP simulator and CAINTEST interaction
EDP conversation (continued)

14 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP902 SEP09 17:12:46 6700 INFO VAMP OUTBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : PC=00-09-01, SSN=33
        PACKAGE : CWP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 05
        OPCODE  : NETWORK_BUSY
          *** PARAMETERS ***
          USERID  : TRK  273
          BEARCAP : SPEECH
          NOTIFIND: NO
        ------------- hex dump of message -------------
        E5 25 C7 08 00 00 17 00 00 00 07 00 E8 19 E9 17
        CF 01 05 D1 02 64 17 30 0E BF 35 04 85 02 01 11
        8D 01 00 9F 6F 01 00
```

15 The simulator checks table CAINKEY to determine if the EDP-Request matches the datafill.

NETWORK_BUSY TRK 0273 EDPREQ (21) (22) (23) (24) (25) (26) (27) (29) $

Since the message, USERID, and TRIGCRIT match, the simulator indexes table CAINMTCH and looks for a match in key 21.

16 The simulator enters CAINMTCH to determine if the parameters match the datafill. A match is found at tuple 21.

21 $ RESP DISCONNECT01

Tuple 21 indicates that the simulator should respond by indexing table CAINRESP on the DISCONNECT01 tuple.

17 The switch generates a CAIN900 log to show SCP simulator matching information.

Example of a MAP response:

```
250X     CAIN900 SEP09 17:12:46 6800 INFO SCP SIMULATOR INDICES
          TRANSID = 0000 0017
          CAINKEY = NETWORK_BUSY
                    TRK  0273
                    EDPREQ
         CAINMTCH = 21
```

18 The simulator enters table CAINRESP and identifies the **`Disconnect`** message identified by the DISCONNECT01 tuple.

DISCONNECT01 CR 0 RESP INVL DISCONNECT $ $ $

SCP simulator and CAINTEST interaction EDP conversation (end)

19 The switch generates a VAMP901 log.

Example of a MAP response:

```
250X     VAMP901 SEP09 17:12:46 6900 INFO VAMP INBOUND MESSAGE
        KEY     : MSGSET=CAIN02, PROTO=TCAP_SCCP
        CONTXT  : DEFAULT
        TRID    : 00 00 00 17
        ROUTE   : PC=00-09-01, SSN=33
        PACKAGE : RESP
        *** COMPONENT ***
        COMP    : INVL
        COMP_ID : 16 05
        OPCODE  : DISCONNECT
          *** PARAMETERS ***
        ------------- hex dump of message -------------
        E4 14 C7 04 00 00 17 00 E8 0C E9 0A CF 02 16 05
        D1 02 65 03 30 00
```

20 CAINTEST receives the **`Disconnect`** message.

Example of a MAP response:

```
TIME --> Received response in 0.18 sec
DISCONNECT received
Message from the SCP to discontinue the call.
Mailbox deallocated.
```

NetworkBuilder SOC functionality

Software Optionality Control (SOC) enables software to be defined and delivered in product computing-module loads (PCL). All functionality in a PCL is categorized as either base or optional. Base functionality is available for immediate use. Optional functionality is grouped into commercial units called SOC options, which can be purchased by operating companies. SOC options correspond to functional groups and functions and are controlled by Nortel Networks-supplied passwords.

SOC is the tool for managing options in a PCL. These options reside in the software. When an operating company purchases an option, SOC allows the company to monitor and control its use. Options can be ordered, activated, and used without a software reload or restart.

SOC commands

There are four commands associated with SOC options. This section briefly describes each of the commands. The *DMS-100 Family Software Optionality Control User's Manual* provides detailed information on using these commands in SOC-related procedures.

ASSIGN command

The ASSIGN command allows users to grant the right-to-use (RTU) for an option, to assign a usage limit to an option, to change the state of an option, or to assign a warning threshold to an option.

DBAUDIT command

The DBAUDIT command allows users to tell the system to perform an audit of the SOC database. SOC does regular audits automatically. The audit requested by the DBAUDIT command provides the user with the information at the MAP terminal in addition to the logs generated by the audit.

REMOVE command

The REMOVE command allows users to remove the RTU from a state option.

SELECT command

The SELECT command allows users to display information and generate reports on SOC options.

Note: Refer to the *UCS DMS-250 Software Optionality Control User's Manual* for more information on UCS DMS-250 SOC or the *DMS-100 Family Software Optionality Control User's Manual* for general SOC information.

NetworkBuilder SOC

This chapter describes the NetworkBuilder SOCs. Table 5-1 provides a list of the NetworkBuilder SOC order codes.

Table 5-1
NetworkBuilder SOC order codes

Order code	SOC name	Available functionality when SOC is on
CAIN0100	Messages	Usage option for all CAIN messages to the SCP
CAIN0200	Extension Parms	Extension parameter set
CAIN0300	SCP Simulator	SCP simulator to test NetworkBuilder functionality
CAIN0400	Test Query Tool	Ability to send test messages (CAINTEST)
CAIN0500	CUSTDP Trigger	Controls the *Customized_Dialing_Plan* trigger
CAIN0501	SPECDIG Trigger	Controls the *Specific_Digit_String* trigger
CAIN0502	OFFHKIM Trigger	Controls the *Off_Hook_Immediate* trigger
CAIN0503	SIOTRK Trigger	Controls the *Shared_Interoffice_Trunk* trigger
CAIN0504	PRIBCHNL Trigger	Controls the *PRI_B-Channel* trigger
CAIN0505	ONOANSWER Trigger	Controls the *O_No_Answer* trigger
CAIN0506	NETBUSY Trigger	Controls the *Network_Busy* trigger
CAIN0507	OCLDBUSY Trigger	Controls the *O_Called_Party_Busy* trigger
CAIN0508	OFTRREQ Trigger	Controls the *O_Feature_Requested* trigger
CAIN0509	OIECREO Trigger	Controls the *O_IEC_Reorigination* trigger
CAIN0510	TERMATT Trigger	Controls the *Termination_Attempt* trigger
CAIN0511	SPECFEAT Trigger	Controls the *Specific_Feature_Code* trigger
CAIN0512	OFFHKDEL Trigger	Controls the *Offhook_Delay* trigger

—continued—

Table 5-1
NetworkBuilder SOC order codes (continued)

Order code	SOC name	Available functionality when SOC is on
CAIN0513	TOLLFREE Trigger	Controls the *TOLLFREE* Trigger
CAIN0600	Con Digit Collect	Controls CAIN conversational digit collection
CAIN0601	SCP Trigger Sub	Controls SCP trigger subscription
CAIN0602	EDPs	Controls Network_Busy, O_Term_Seized, O_Called_Party_Busy, O_Answer, and O_No_Answer EDPs
CAIN0603	STR Connection	Controls the STR connection
CAIN0604	Inter IMT Support	Controls triggering for SS7 Inter-IMTs
CAIN0605	Global IMT Support	Controls triggering for SS7 Global-IMTs
CAIN0606	1129-Style IP	Controls 1129-style IP interaction
CAIN0607	Virtual IP	Controls virtual IP interaction
CAIN0609	Term Notification	Controls Termination Notification
CAIN0610	CAINPRT Digit Coll	Controls the CAIN pretranslator digit collectible
CAIN0700	LNP QOO	Controls triggering for Local Number Portability
CAIN0800	Mid Call Services 1	Controls the Timeout and O_Disconnect events
CAIN0801	Mid Call Services 2	Controls Connect_To_Resource message processing at the Timeout event
CAIN0802	Takeback & Transfer	Controls the O_Abandon and Switch_Hook_Flash events
CAIN0900	Auto Code Gapping	Controls Automatic Code Gapping
CAIN0901	Manual Code Gapping	Controls Manual Code Gapping

—end—

Feature descriptions

This section briefly describes each of the features included in the NetworkBuilder SOC, as well as any dependencies.

Order code CAIN0100

The CAIN0100 order code provides a usage option for all NetworkBuilder messages to the SCP. This usage option has a default hard limit of zero. A hard limit means that no resources beyond this limit can be assigned. Refer to the *UCS DMS-250 Software Optionality Control User's Manual* for more information on usage limits.

Note: SOC usage counters are displayed in incremental units of 1000.

Order code CAIN0200

The CAIN0200 order code provides the extension parameter set that can be used in addition to the standard message parameters.

Order code CAIN0300

The CAIN0300 order code provides an SCP simulator. This tool accepts queries from the switch and simulates SCP messages back to the switch. The switch uses the simulator to test its NetworkBuilder functionality.

Order code CAIN0400

The CAIN0400 order code provides the CAINTEST query tool. This tool enables operating companies to test their SCP by sending messages and evaluating responses.

Order code CAIN0500-series

NetworkBuilder call processing uses triggers to determine when the switch needs to query the SCP. The following trigger order codes are available:

- CAIN0500 (CUSTDP Trigger)
- CAIN0501 (SPECDIG Trigger)
- CAIN0502 (OFFHKIMM Trigger)
- CAIN0503 (SIOTRK Trigger)
- CAIN0504 (PRIBCHNL Trigger)
- CAIN0505 (ONOANSWER Trigger)
- CAIN0506 (NETBUSY Trigger)
- CAIN0507 (OCLDBUSY Trigger)
- CAIN0508 (OFTRREQ Trigger)
- CAIN0509 (OIECREO Trigger)
- CAIN0510 (TERMATT Trigger)
- CAIN0511 (SPECFEAT Trigger)
- CAIN0512 (OFFHKDEL Trigger)
- CAIN0513 (TOLLFREE Trigger)

Order code CAIN0600-series

The CAIN0600-series of order codes provide special NetworkBuilder features. The following order codes are available:

- CAIN0600 (Con Digit Collect)

- CAIN0601 (SCP Trigger Sub)
- CAIN0602 (EDPs)
- CAIN0603 (STR Connection)
- CAIN0604 (Inter IMT Support)
- CAIN0605 (Global IMT Support)
- CAIN0606 (1129-Style IP)
- CAIN0607 (Virtual IP)
- CAIN0609 (Termination Notification)
- CAIN0610 (CAINPRT Digit Coll)

Order code CAIN0700

The CAIN0700 (LNP QOO) order code provides support for LNP functions.

Order code CAIN0800-series

The CAIN0800-series of order codes provide Mid-Call services features. The following order codes are available:

- CAIN0800 (Mid Call Services 1)
- CAIN0801 (Mid Call Services 2)
- CAIN0802 (Takeback & Transfer)

Order code CAIN0900-series

The CAIN0900-series of order codes provide code gapping features. The following order codes are available:

- CAIN0900 (Auto Code Gapping)
- CAIN0901 (Manual Code Gapping)

Dependencies

Table 5-2 shows NetworkBuilder SOC dependencies.

Table 5-2
NetworkBuilder SOC dependencies

SOC	Software Prerequisite	Release issued
CAIN0100	None	UCS05
CAIN0200	CAIN0100 and at least one of the CAIN0500-series through CAIN0512	UCS05
CAIN0300	CAIN0100 and at least one of the CAIN0500-series or CAIN0700	UCS05
CAIN0400	CAIN0100 and at least one of the CAIN0500-series or CAIN0700	UCS05
CAIN0500	CAIN0100	UCS05
CAIN0501	CAIN0100	UCS05
CAIN0502	CAIN0100	UCS06
CAIN0503	CAIN0100	UCS06
CAIN0504	CAIN0100	UCS06
CAIN0505	CAIN0100	UCS06
CAIN0506	CAIN0100	UCS06
CAIN0507	CAIN0100	UCS06
CAIN0508	CAIN0100	UCS06
CAIN0509	CAIN0100	UCS08
CAIN0510	CAIN0100	UCS08
CAIN0511	CAIN0100	UCS08
CAIN0512	CAIN0100	UCS08
CAIN0513	CAIN0100 and UTRS0001	UCS09
CAIN0600	CAIN0100 and at least one of the CAIN0500-series through CAIN0512	UCS06
CAIN0601	CAIN0100, CAIN0200, and at least one of the CAIN0500-series through CAIN0512	UCS06
CAIN0602	CAIN0100 and at least one of the CAIN0500-series through CAIN0512	UCS07
CAIN0603	CAIN0100, CAIN0600, and at least one of the CAIN0500-series through CAIN0512	UCS07

—continued—

Table 5-2
NetworkBuilder SOC dependencies (continued)

SOC	Software Prerequisite	Release issued
CAIN0604	NSER0003, CAIN0100, and at least one of the CAIN0500-series or CAIN0700	UCS07
CAIN0605	GIMT0001, CAIN0100, and at least one of the CAIN0500-series	UCS07
CAIN0606	CAIN0100, CAIN0600, and at least one of the CAIN0500-series through CAIN0512	U CS08
CAIN0607	CAIN0100, CAIN0600, and at least one of the CAIN0500-series through CAIN0512	UCS08
CAIN0609	CAIN0100 and at least one of the CAIN0500-series	UCS09
CAIN0610	CAIN0100 and CAIN0508	UCS08
CAIN0700	CAIN0100	UCS07
CAIN0800	CAIN0100 and at least one of the CAIN0500-series through CAIN0512	UCS08
CAIN0801	CAIN0100, CAIN0600, CAIN0800, at least one of the CAIN0500-series through CAIN0512, and Hardware prerequisite NT1X81 or NT3X67	UCS09
CAIN0802	CAIN0100 and at least one of the CAIN0500-series through CAIN0512	UCS11
CAIN0900	CAIN0100 and at least one of the CAIN0500-series or CAIN0700	UCS09
CAIN0901	CAIN0100 and at least one of the CAIN0500-series through CAIN0512 or CAIN0700	UCS09

—end—

Datafill information

The NetworkBuilder SOC options do not affect any datafill.

Office parameters

The NetworkBuilder SOC options do not affect any office parameters.

TRAVER

The TRAVER (translation verification) tool simulates a call from a user specified originating trunk to a user-specified address. TRAVER examines and displays translation and routing data for a single call leg.

TRAVER performs the following functions:

- verifies the translation tables
- aids in debugging and analyzing translation and routing datafill.
- helps determine reasons for unexpected results and changes required to achieve the expected results.

Note: Unexpected results may include unexpected treatments or improper routing.

TRAVER is capable of displaying the following information:

- tables used to translate and route a call
- treatment
- CAIN subscription method and group
- tuple (from the appropriate trigger table) where trigger criteria was met
- message parameters

Note: For AXXESS agents the FLEXSIM tool is used in place of the Traver tool. Refer to *UCS DMS-250 CAIN/FlexDial Interactions* for more information.

Access

TRAVER is accessible from any MAP level. You must set a default STS before using the TRAVER tool. Enter the following command:

>UTVSTS sts

where

sts is the serving translation scheme (000 to 999)

Sample entry: **>UTVSTS 214**

Parameters

The TRAVER command has a complex syntax. This complexity is due to having various types of required and optional subfields for each of several originators. If you were to list the TRAVER command syntax online, you would see functionality for some originators other than those that exist on the current load for the switch. CAIN only uses the TR originator.

CAUTION
TRAVER contains functionality that is not available.
Some TRAVER command functionality visible on your screen is not available on your load. Please ignore it.

TRAVER allows three trace options: `t`, `nt`, and `b`. CAIN uses only the `t` option to display the tables and tuple accessed in each table used to translate and route the call. The `t` option uses parallel software to simulate a call and display the tables used to translate and route a call along the appropriate tuple for each table.

TRAVER syntax for CAIN

Use the following command syntax to obtain CAIN translation information.

>**TRAVER TR clli digits [digit_options] T [trace options]**

Table 6-1 describes the originator parameter and the required and optional parameters that follow the originator up to the digits parameter.

Table 6-1
TRAVER origination and other parameters

Parameter		Subfields	Value	Description
<clli>			Valid CLLI datafilled in table TRKGRP as being CAIN capable	Entered when an incoming trunk is the originator. TR must be followed by the trunk CLLI identifier. CLLI can be followed by the CLLI option and the MBG option.
<digits>			0 to 30 digits	The digits entered are the called party's number. These are the digits entered by the originator or received by the originator on an incoming trunk. N entered in the <digits> field simulates trunk seizure.
[digit_options]	[cdnie]	CDN [<npi>] [<cdnton>] <cdn>	"CDN" followed by subfields	Use this parameter for PRI originations. Indicates the called party number (CDN) option is specified. This is followed by two optional and one required CDN subfields.
		[<npi>]	string; as follows:	(Optional) To perform a TRAVER on a PRA trunk, the TR parameter can include the numbering plan indicator (NPI) option. The <npi> specifies that an NPI is associated with the called party number.
			E164	Public NPI
			PUB	Public NPI (default)
			PVT	Private NPI

Note 1: Required parameters are enclosed by < >. Optional parameters are enclosed by [ ].

Note 2: Digit strings consisting of fewer than six digits require the numbers to be placed in quotes (for example, enter '12345', not 12345.

—continued—

Table 6-1
TRAVER origination and other parameters (continued)

Parameter		Subfields	Value	Description
		[<cdnton>]	string; as follows:	(Optional)
			IN	
			NA	
			L	
			NET	
			ABBR	
			UNK	
		<cdn>	string	Required parameter for CDN option
[trace_options]	[<authcode>]		0 to 7 digits	Authorization code. Use the NIL value (N) before entering an ANI.
Note 1: Required parameters are enclosed by < >. Optional parameters are enclosed by [ ].				
Note 2: Digit strings consisting of fewer than six digits require the numbers to be placed in quotes (for example, enter '12345', not 12345.				
—end—				

Example commands

The following procedural examples show TRAVER command entries for CAIN.

Example one

Subscription method: Agent subscribes to CAIN group OFFGRP4

- OFFGRP4 enables the **O_Null**, ***Origination_Attempt***, *Off_Hook_Immediate* trigger set for this call.

  — Trigger action: BLOCK

1 >UTVSTS 611

2 >traver tr dal220twdtgs 2211234 t

3 *Example of a MAP response:*

```
STS USED FOR TRAVER IS: 611
TABLE STSTOPAR
611 00 111
TABLE TRKGRP
DAL220TWDTGS DAL 30 NPDGP NCIT 0 2W DAL MIDL 16 7 16 16 S 7 NIL ID 0 7 111
    MANUAL 214 0 6113311 RTE2 0 3_1KHZ Y 1 N Y NONE 00 160 (CAIN )
    (CAINGRP OFFGRP4) $
--> ENTER TDP ORIGATT <--
+++ CAIN SUBSCRIPTION VIA AGENT +++
TABLE CAINGRP
OFFGRP4 23 CAIN02 TCAP_SCCP (O_NULL ORIGATT OFFHKIMM) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE OFFHKIMM
OFFGRP4 BLOCK $
TREATMENT ROUTES.  TREATMENT IS: AINF

+++ TRAVER: SUCCESSFUL CALL TRACE +++
```

Example two – Multiple CAIN group subscription

Subscription method: Agent subscribes to group OFFGRP
Office subscribes to group SPECGRP

- OFFGRP enables the **O_Null**, ***Origination_Attempt***, *Off_Hook_Immediate* trigger set for this call.
  - — Trigger action: IGNORE
- SPECGRP enables the **Analyze_Information**, ***Info_Analyzed***, *Specific_Digit_String* trigger set for this call.
  - — Trigger criteria: ADDR
  - — Trigger action: QUERY

1 >UTVSTS 611

2 >traver tr dal221twdtls 2201234 t

3 *Example of a MAP response:*

```
STS USED FOR TRAVER IS: 611
TABLE STSTOPAR
611 00 111
TABLE TRKGRP
DAL221TWDTLS DAL 30 NPDGP NCIT 0 2W DAL MIDL 16 7 16 16 S 7 NIL ID 0 7 111
    MANUAL 214 0 6113311 RTE2 0 3_1KHZ Y 1 N Y ANISNPA 00 160 (ALTTRTMT )
    (VANIDB ) (CAIN ) (CAINGRP OFFGRP) $
--> ENTER TDP ORIGATT <--
+++ CAIN SUBSCRIPTION VIA AGENT +++
TABLE CAINGRP
OFFGRP 20 CAIN02 TCAP_SCCP (O_NULL ORIGATT OFFHKIMM) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE OFFHKIMM
OFFGRP IGNORE $
```

```
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
--> ENTER TDP O_FTRREQ <--
+++ CAIN SUBSCRIPTION VIA AGENT +++
TABLE CAINGRP
OFFGRP 20 CAIN02 TCAP_SCCP (O_NULL ORIGATT OFFHKIMM) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
--> ENTER TDP INFOCOLL <--
+++ CAIN SUBSCRIPTION VIA AGENT +++
TABLE CAINGRP
OFFGRP 20 CAIN02 TCAP_SCCP (O_NULL ORIGATT OFFHKIMM) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE STDPRTCT
DAL ( 1) ( 0) 0
 . SUBTABLE STDPRT
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
 . 22 304 CT ONNET 6 7 0 $
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
TABLE HNPACONT
611 999 10 ( 423) ( 1) ( 0) ( 0) 0
 . SUBTABLE HNPACODE
 . 220 220 LRTE 220
 . SUBTABLE RTEREF
 .  220 S D DAL220TWDTGS
 . EXIT TABLE RTEREF
EXIT TABLE HNPACONT
--> ENTER TDP INFOANLZ <--
+++ CAIN SUBSCRIPTION VIA AGENT +++
TABLE CAINGRP
OFFGRP 20 CAIN02 TCAP_SCCP (O_NULL ORIGATT OFFHKIMM) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE SPECDIG
SPECGRP ADDR 2201234 2201234 QUERY SDS_ADDR ROUTE $

             UserID:    DAL221TWDTLS 11
```

```
   BearerCapability:    SPEECH
      CalledPartyID:    2201234
TriggerCriteriaType:    SDS_ADDR
     CallingPartyID:    214

+++ TRAVER: SUCCESSFUL CALL TRACE +++
```

Example three

Subscription method: ANI subscribes to group SPECGRP

- SPECGRP enables the **Analyze_Information**, ***Info_Analyzed***, *Specific_Digit_String* trigger set for this call.
  - — Trigger criteria: ANI
  - — Trigger action: BLOCK

1 >UTVSTS 611

2 >traver tr ean671twmfwk 2201234 t n 2148622279

3 *Example of a MAP response:*

```
STS USED FOR TRAVER IS: 611
TABLE STSTOPAR
611 00 111
TABLE TRKGRP
EAN671TWMFWK EANT 0 NPDGP NCIT 0 2W EAN MIDL 16 7 16 16 EAPT 5 5 214 NILIDX
NIL
    214 111 MANUAL 0 NONE 0 1 3_1KHZ 160 (ALTTRTMT ) (TMANIDLV ALWAYS) (CAIN
) $
--> ENTER TDP O_FTRREQ <--
+++ CAIN SUBSCRIPTION VIA ANI +++
TABLE ANISCUSP
2148622279 SUB AL 0 N 3_1KHZ 862 0 N 0 0 $ 0 ALWAYS (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
--> ENTER TDP INFOCOLL <--
+++ CAIN SUBSCRIPTION VIA ANI +++
TABLE ANISCUSP
2148622279 SUB AL 0 N 3_1KHZ 862 0 N 0 0 $ 0 ALWAYS (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE STDPRTCT
EAN ( 1) ( 0) 0
```

```
 . SUBTABLE STDPRT
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
 . 22 380 CT ONNET 6 7 0 $
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
TABLE HNPACONT
611 999 10 ( 423) ( 1) ( 0) ( 0) 0
 . SUBTABLE HNPACODE
 . 220 220 LRTE 220
 . SUBTABLE RTEREF
 .  220 S D DAL220TWDTGS
 . EXIT TABLE RTEREF
EXIT TABLE HNPACONT
--> ENTER TDP INFOANLZ <--
+++ CAIN SUBSCRIPTION VIA ANI +++
TABLE ANISCUSP
2148622279 SUB AL 0 N 3_1KHZ 862 0 N 0 0 $ 0 ALWAYS (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE SPECDIG
SPECGRP ANI 2148622279 2148622279 BLOCK $
TREATMENT ROUTES.  TREATMENT IS: AINF

+++ TRAVER: SUCCESSFUL CALL TRACE +++
```

Example four

Subscription method: Authorization code subscribes to group SPECGRP, Office subscribes to SPECGRP

- SPECGRP enables the **Analyze_Information**, ***Info_Analyzed***, *Specific_Digit_String* trigger set for this call.
  - — Trigger criteria: ANI
  - — Trigger action: IGNORE

1 >UTVSTS 611

2 >traver tr ean671twmfwk 2201234 t 6112211 2148622278

3 *Example of a MAP response:*

```
STS USED FOR TRAVER IS: 611
TABLE STSTOPAR
611 00 111
TABLE TRKGRP
EAN671TWMFWK EANT 0 NPDGP NCIT 0 2W EAN MIDL 16 7 16 16 EAPT 5 5 214 NILIDX
NIL
    214 111 MANUAL 0 NONE 0 1 3_1KHZ 160 (ALTTRTMT ) (TMANIDLV ALWAYS) (CAIN
) $
--> ENTER TDP O_FTRREQ <--
+++ CAIN SUBSCRIPTION VIA AUTHCODE +++
TABLE AUTHCODU
6112211 VALID 0 0 111 0 $ 0 $ 51 N N N N 0 Y 0 0 (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
```

```
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
--> ENTER TDP INFOCOLL <--
+++ CAIN SUBSCRIPTION VIA AUTHCODE +++
TABLE AUTHCODU
6112211 VALID 0 0 111 0 $ 0 $ 51 N N N N 0 Y 0 0 (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE STDPRTCT
EAN ( 1) ( 0) 0
 . SUBTABLE STDPRT
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
 . 22 380 CT ONNET 6 7 0 $
WARNING: CHANGES IN TABLE STDPRT MAY ALTER OFFICE
BILLING.  CALL TYPE DEFAULT IS NP.  PLEASE REFER TO
DOCUMENTATION.
TABLE HNPACONT
611 999 10 ( 423) ( 1) ( 0) ( 0) 0
 . SUBTABLE HNPACODE
 . 220 220 LRTE 220
 . SUBTABLE RTEREF
 .  220 S D DAL220TWDTGS
 . EXIT TABLE RTEREF
EXIT TABLE HNPACONT
--> ENTER TDP INFOANLZ <--
+++ CAIN SUBSCRIPTION VIA AUTHCODE +++
TABLE AUTHCODU
6112211 VALID 0 0 111 0 $ 0 $ 51 N N N N 0 Y 0 0 (CAINGRP SPECGRP) $
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE SPECDIG
SPECGRP ANI 2148622278 2148622278 IGNORE $
+++ CAIN SUBSCRIPTION VIA OFFICE +++
TABLE CAINPARM
CAIN_OFFICE_GROUP SPECGRP
TABLE CAINGRP
SPECGRP 7 CAIN02 TCAP_SCCP (ANLZINFO INFOANLZ SPECDIG) $ $ (ADIN) (CAINGRP)
    (ORGTINFO) $
TABLE SPECDIG
SPECGRP ANI 2148622278 2148622278 IGNORE $

+++ TRAVER: SUCCESSFUL CALL TRACE +++
```

VPTRACE

VPTRACE allows you to enable or disable the tracing of VAMP messages (logged in VAMP901 and VAMP902 logs). If VPTRACE is used with no parameter, the command displays the current status of tracing (enabled or disabled).

Inbound messages are displayed in VAMP901 logs. Outbound messages are displayed in VAMP902 logs.

CAUTION
Using VPTRACE incurs real-time cost
Enabling VAMP 90x message tracing logs incurs a real-time cost on every AIN message sent or received. This can affect call processing capacity during high traffic periods. Only enable these logs during low traffic periods or when necessary to debug a messaging problem.

Parameters

>VPTRACE logcmd

Table 7-1 describes the VPTRACE parameter.

Table 7-1
VPTRACE parameter field descriptions

Field	Value	Explanation and action
Logcmd	ENABLE or DISABLE	(OPTIONAL) Enables or disables VAMP 90x logs

MAP responses

The following example shows a MAP response to the VPTRACE command.

>VPTRACE DISABLE

VPTRACE (end)

Table 7-2 lists possible responses to the VPTRACE command.

Table 7-2
VPTRACE responses

Response	Explanation and action
`VAMP MESSAGE TRACE LOGS DISABLED`	VAMP message tracing is disabled. No further messages will be logged.
`VAMP MESSAGE TRACE LOGS ENABLED`	VAMP message tracing is enabled. All inbound and outbound messages will be logged.

VAMP message trace logs

Table 7-3 lists the VAMP logs that require the use of the VPTRACE tool.

Table 7-3
VAMP message trace logs

Log	Explanation
VAMP901	The UCS DMS-250 switch generates this log when an inbound message is received through the VAMP framework and the VPTRACE tool has enabled message monitoring.
VAMP902	The UCS DMS-250 switch generates this log when an outbound message is sent through the VAMP framework and the VPTRACE tool has enabled message monitoring.
Note: Refer to *UCS DMS-250 Logs Reference Manual* for more information.	

List of terms

advanced intelligent network
: A network that allows the switch to off-load some of the call processing functions to an intelligent service control point (SCP).

AMA
: automatic message accounting

AIN
: advanced intelligent network

AIND
: AIN disconnect treatment

AINF
: AIN final treatment

ANI
: automatic number identification

ANISCUSP
: Automatic Number Identification Screening Customer Profile table. May be used to assign a CAIN group to a particular ANI.

ANIVAL
: Automatic Number Identification Value table. May be used to a 3-, 6-, or 10-digit ANI, a profile type, and an index that maps to table UNIPROF.

AUTHCDU2
: Authcode Database 2 table. May be used to assign a CAIN group to a particular authorization code.

AUTHCDU3
: Authcode Database 3 table. May be used to assign a CAIN group to a particular authorization code.

AUTHCDU4

Authcode Database 4 table. May be used to assign a CAIN group to a particular authorization code.

AUTHCDU5

Authcode Database 5 table. May be used to assign a CAIN group to a particular authorization code.

AUTHCODU

Authcode Database table. May be used to assign a CAIN group to a particular authorization code.

authorization code

A unique multidigit code that identifies an authorized subscriber. Authorization codes are usually 5-7 digits (due to UCS DMS-250 limitations) and identify a subscriber, bill a call, prevent unauthorized network use, determine the originating caller's class of service, and control access to special features. For example, a caller can be required to enter an authorization code to retrieve voice mail messages.

automatic number identification

billing number for the calling party provided to the IEC from the LEC

BAF

Bellcore Automatic Message Accounting Format

BC

bearer capability

BCD

binary coded decimal

bearer capability

A characteristic associated with a directory number to indicate the type of call (voice or data) and the rate of transmission allowed.

CC

Call configuration. Call configurations provide 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).

C7GTT

CCS7 Global Title Translation table. Maps a translation type (defined in table C7GTTYPE) to a CCS7 network address.

C7GTTYPE

CCS7 Global Title Type table. Maps a CCS7-defined translation to a network-defined global title translation type.

C7LINK

CCS7 Link table. Makes the association between the physical equipment of the link and the logical view of the link as a member of a linkset.

C7LKSET

CCS7 Link Set table. Defines the characteristics of a linkset. A linkset is a set of links used as a group. Each link carries traffic between the origination point code and a destination point code. The table also defines attributes that are common to all links in the link set. The links are defined in table C7LINK.

C7LOCSSN

CCS7 Location Subsystem Number table. Defines the subsystems located on the switch.

C7NETSSN

CCS7 Network Subsystem Number table. Provides the set of remote point codes (PC) and subsystems, at the remote PCs, where messages are routed by the SCCP. A PC is a node in the CCS7 network that may be an SSP, an STP, or an SCP.

C7NETWRK

CCS7 Network table. Describes the signaling networks in use in a switching office.

C7RPLSSN

CCS7 Replicate Subsystem table. Provides the set of remote subsystem replicate pairs. It has a one part key, the subsystem name. For each subsystem a list of PC pairs at which the replicated subsystems reside must be given.

C7RSSCRN

CCS7 Remote Subsystem Concerned Node table. Provides a list of concerned nodes for a remote subsystem point code combination. The table has a two part key. The first part is the PC and the second part is the subsystem name. The PC and subsystem combination must be datafilled in table C7NETSSN.

C7RTESET

CCS7 Route Set table. Associates linksets used as possible routes for each signaling point in the network. An office point code identifies a signaling point within any network. Each office point code must have a routeset. The information in this table records which routes and linksets can carry the signaling information to the destination signaling point. This table is also used for alternate routing decisions.

CAIN routing parameters

Parameters provided in an **`Analyze_Route`** response. The parameters are: ***`PrimaryTrunkGroup`***, ***`AlternateTrunkGroup`***, ***`SecondAlternateTrunkGroup`***, ***`Carrier`***, ***`AlternateCarrier`***, ***`SecondAlternateCarrier`***, ***`CalledPartyID`*** and `servTranslationScheme` (or `univIdx` for SS7 Global-IMTs), and the ***`GenericAddressList`***'s `OverflowRoutingNo`.

CAINCONV

CAIN Conversation table. Controls SCP simulator interaction during conversation with the switch.

CAINGRP

CAIN Group table. Identifies CAIN groups and trigger sets used for CAIN subscription.

CAINKEY

CAIN Key table. Determines a range of possible responses for a defined three-part key into the SCP simulator. The range of possible responses is represented by an option vector of indexes into table CAINMTCH.

CAINMTCH

CAIN Matching table. Screens and selects possible responses for the SCP simulator

CAINPARM

CAIN Parameters table. Assigns CAIN office parameter values.

CAINRESP

CAIN Response table. Contains response data to return to the switch. The SCP simulator's encoder takes this data and builds a Transaction Capabilities Application Part (TCAP) message.

CAINREXT

Cain Response Extension Parameters table. This table contain's the extension parameters used to build a response message.

CAINRSRC

CAIN Resource table. Maps the data returned from the SCP to a resource available on the switch.

CAINSCPT

CAINSCPT supports the CAIN SCP Simulator. With this tool the user is able to select different commands to perform different tasks having to do with transaction IDs (TRIDs).

CAINUID

CAIN User Identification table. Provides symbolic names for trunk groups and switches used in the simulator. It is similar to table CLLI in function. The use of symbolic names rather than numbers provides enhanced clarity when datafilling the simulator tables.

CAINXDFT

CAIN Extension Parameter Defaults table. Defines default values for five extension parameters: `servTranslationScheme`, `callType`, `satRestriction`, `classofSvc`, and `callBranding`.

call branding

Tones or announcements (returned from the SCP or provisioned in table CAINXDFT) played by the switch as directed by CAIN call processing before routing is attempted.

call detail record

Formatted billing data used to generate subscriber billing.

call model

generic representation of a basic call in terms of the processing activities required to establish, maintain, and clear a call.

call processing

The function of call processing software is to establish connections among telephony agents. A number of functional steps are required to process a call, such as detecting the incoming call, receiving digits, analyzing (translating) digits to determine call destination, selecting terminating agent, establishing connection, signaling to and detecting an answer from the terminating agent, and detecting disconnect.

CALLATTR

Call Attributes table. Used to provision a PRI call attribute.

called number

The number of the party receiving the call. Also known as called party ID.

called party

The end user that receives a call.

calling line identification

In data transmission, a feature provided by the network that allows a called terminal to be notified by the network of the address from which the call was originated.

calling number

The number of the party initiating the call. This number can identify the origin of a call to the called party. Also known as calling party number or calling party ID.

calling party

The end user that originates a call.

carrier identification code

Three- to four-digit number that identifies which interexchange carrier a call will use. Subscribers can dial these digits with each long distance call, or can pre-subscribe to a particular carrier and let the digital switch software add the CIC.

CCS7

Common Channel Signaling No. 7

CDR

call detail record

CIC

carrier identification code

CIP

carrier identification parameter

CLI

calling line identification

CLLI

common language location identifier

CM

computing module

common channel signaling No. 7

A digital, message-based network signaling standard defined by the CCITT which separates call signaling information from voice channels so that interoffice signaling is exchanged over a separate signaling link.

common language location identifier

A standard identification method for trunk groups, tones, and announcements.

computing module

The processor and memory of the dual-plane combined core used by the DMS SuperNode. Each CM consists of a pair of central processing units (CPU) with associated memory that operate in a synchronous matched mode on two separate planes. Only one plane is active; it maintains overall control of the system while the other plane is on standby.

conversation package

A transaction capabilities application part (TCAP) package expecting a reply. It can be sent by the service control point (SCP), or adjunct, to the service switching point (SSP) that instructs the SSP to perform a user interaction, collect digits, and send a reply to the SCP. The SSP can also send a conversation package to the SCP.

COS

class of service

CPID

calling party identification

CSI

carrier selection indicator

CUSTDP

Customized Dialing Plan table. Defines trigger criteria for *Customized_Dialing_Plan*.

DAL

dedicated access line

dedicated access line

Network connection, often leased from a local exchange carrier or competitive access provider, that provides a direct link from a customer to the long distance network. Typical DALs include outbound WATS lines, PBX tie trunks, and foreign exchange lines.

digital recorded announcement machine (DRAM)

A peripheral module (PM), developed for the DMS switch, in which voice messages are stored in digital form, providing access to up to 30 different service voice announcements.

DP

detection point

dual-tone multifrequency (DTMF) signaling
A signaling method employing set combinations of two specific voice-band frequencies, one of which is selected from a group of four low frequencies, and the other from a group of three or four relatively high frequencies.

EANT
equal access network trunking

EDP
event detection point

FAMA
Furnish AMA

fast interdigit (FIDT) timer
time allowed for the subscriber to dial digits between the minimum and maximum required

FCC
Federal Communications Commission

Federal Communications Commission
An agency of the U. S. Government that regulates standards and companies within all aspects of the communications industry (radio, television, telephony).

FGD
feature group D

FIDT
fast interdigit timer

FlexDial
FlexDial Framework is a UCS DMS-250 feature that allows you to program the call origination side of a call to implement customized functionality. UCS06 software does not support CAIN/FlexDial interaction.

GDP
generic digits parameter

generic digits parameter
ISUP parameter used to transport generic digits with a specified identifier tag

global title
An application address. The SCCP global title translation (GTT) function is required to translate a GT into a valid network address.

global title translation

The process that translates an application-specific address (such as a dialed 800 number) into the Common Channel Signaling No. 7 (CCS7) PC subsystem address, usually that of the appropriate service control point (SCP).

GT

global title

GTT

global title translation

GVNS

global virtual network service

hotline

A connection that has another address mapped to it. This second address can be filed in a table, and therefore not changeable by the end user, or it can be operated by an authcode for an effect similar to speed dialing.

IAM

initial address message

IN

intelligent network

IN/1

UCS proprietary intelligent network protocol for offboard databases with limited call control

IEC

interexchange carrier

IMT

intermachine trunk

information digit

Digit received by the switch either in the incoming digit stream or with the incoming signaling information. An information digit carries additional information about the call that cannot be determined by the address digits. The type of information digit received and the method it is received depends on the incoming signaling system.

Integrated Services Digital Network

A network that provides end-to-end digital connectivity using CCS7 to support a wide range of voice and data services to the end-user.

intelligent peripheral
Contains functionality and resources for exchanging information (such as voice announcements and dual-tone multifrequency digit collection) with a subscriber.

intermachine trunk
A trunk that connects the UCS DMS-250 IEC networks.

IP
intelligent peripheral

IPI
intelligent peripheral interface

ISDN
Integrated Services Digital Network

ISDN User Part
SS7 protocol that defines the messages, parameters, and procedures to set up and tear down all circuit switched calls, both ISDN and non-ISDN, in U.S. SS7 networks. It includes support for ISDN Supplementary Voice services and interworks with Q.931/932 to provide end-to-end ISDN.

ISUP
ISDN User Part

LEC
local exchange carrier

LNP
local number portability

MCCS
mechanized calling card service

mechanized calling card service
The service that allows a call to be billed to a calling card number.

message parameters
Parameters within a message. Each message type has its own set of mandatory and optional parameters.

message set
the set of standard messages required by an advanced intelligent network (AIN) specification, such as Bellcore's Release 0.2, for communication among AIN network elements.

message switch and buffer

A peripheral module used by the switch, along with a signaling terminal, to act as an interface to and operate within a common channel signaling environment. The message switch and buffer supports the signaling terminal and routes the messages received by the signaling terminal through the network module to the digital trunk controller. The message switch and buffer also receives messages sent from central control and routes them to the signaling link through the signaling terminal. A different configuration of the message switch and buffer exists for each of the two protocols used to implement common channel signaling.

message switch and buffer 7

The message switch and buffer (MSB) for Common Channel Signaling 7 (CCS7) protocol. *See also* message switch and buffer (MSB)

MSB

message switch and buffer

MSB7

message switch and buffer 7

NETBUSY

Network Busy table. Defines trigger criteria for *Network_Busy*.

NEL

next event list

numbering plan area

Any of the designated geographical divisions of the United States, Canada, Bermuda, and Northwestern Mexico within which no two telephones have the same seven-digit number. Each NPA is assigned a unique three digit area code for World Zone 1 dialing.

OCLDBUSY

O Called Party Busy table. Defines trigger criteria for *O_Called_Party_Busy*.

OFFHKIMM

Off Hook Immediate table. Defines trigger criteria for *Off_Hook_Immediate*.

OFFHKDEL

Off Hook Delay table. Defines trigger criteria for *Off_Hook_Delay*.

off-hook

The condition existing in telephone operations when the receiver or handset is removed from its hookswitch.

OFRT

office route

OFTRREQ

O Feature Requested table. Defines trigger criteria for *O_Feature_Requested*.

OFFCCODE

Office Code table. Defines trigger criteria for *Office_Code*.

OIECREO

Originating Inter-Exchange Carrier Reorigination table. Defines trigger criteria for *O_IEC_Reorigination*.

OM

operational measurements

ONOANSWR

O No Answer table. Defines trigger criteria for O_No_Answer.

operational measurements

The software resources of the switch that control the collection and display of measurements taken on an operating system. The OM subsystem organizes the measurements data and manages its transfer to displays and records. The OM data is used for maintenance, traffic, accounting, and provisioning decisions.

PANI

pseudo-automatic number identification

parameter

Data contained within a message.

partial dial (PDIL) timer

time allowed between each subscriber dialed digit (before minimum number of digits are dialed)

PBX

private branch exchange

PC

point code

PCL

product computing module loads

PDIL

partial dial timer

per-trunk signalling

A conventional telephony method of signaling that multiplexes the control signal of a call with voice or data over the same trunk (in-band signaling).

permanent signal timer

time allowed before the subscriber enters the first digit

personal identification number

Authorization number (usually composed of the caller's telephone number plus a four-digit code) that allows subscribers to access their long distance carriers when away from home.

PIN

personal identification number

PIC

point in call

point in call

A generic representation of a sequence of switch based call processing actions considered essential to establish, maintain, or clear a two-party call. PICs are separated by trigger detection points (TDP).

PRI

primary rate interface

PRIBCHNL

PRI B-CHANNEL table. Defines trigger criteria for *PRI_B-Channel*.

primary rate interface

An interface that carries nB+D channels over a digital DS-1 facility (23B+D in North America and 30B+D in Europe). PRI is used to link private networking facilities, such as private branch exchanges (PBX), local area networks (LAN), and host computers with a standardized architecture acting as the bridge between private switching equipment and the public network. Formerly known as primary rate access.

private branch exchange

A private telephone exchange, either automatic or attendant operated, serving extensions in an organization and providing access to the public network.

pseudo automatic number identification

A 10-digit translations code derived from a combination of the authorization code, personal identification number, and serving number plan area number.

PSIG

permanent signal timer

PSN

programmable service node

PTS

per trunk signaling

query

A type of communication message sent by the service switching point (SSP) to the service control point (SCP), or adjunct, requesting call processing instructions. In AIN Release 0.2, the message is contained in a transaction capabilities application part (TCAP) query package.

REL

CCS7 release message

response package

A transaction capabilities application part (TCAP) package containing one or more messages sent in response to another TCAP package. The service control point (SCP), or adjunct, can use a response package to instruct the service switching point (SSP) to perform an activity. The SSP can also send response packages.

response processing

The service switching point (SSP) receives and processes response messages from the service control point (SCP), or adjunct. There are several different types of response messages, each of which is handled differently according to the information it contains.

SCCP

signaling connection control part

SCE

service creation environment

SCP

service control point

service control point

A node in a Common Channel Signaling No 7 (CCS7) signaling network that supports application databases. The function of an SCP is to accept a query for information, retrieve the requested information from one of its application databases, and to send a response message to the originator of the request.

service switching point

A switch that is capable of interacting with the Common Channel Signaling No. 7 (CCS7) network databases. It contains hardware to support CCS7 signaling, software to control call processing and also create network database query messages, and software to interpret network database response messages.

serving translation scheme

The scheme the UCS DMS-250 switch uses to translate and route a call. STS codes are three digits codes (000-999). The switch uses the three digits serving translation scheme codes (000-999) to derive routing information.

signal transfer point

A switch that is used to provide signaling link connections between switches. That is, it is a tandem node for Common Channel Signaling No. 7 (CCS7) signaling links and contains hardware to support CCS7 hardware and software to route CCS7 messages. It does not contain any software to create or interpret CCS7 messages. STPs are deployed in pairs. If one STP fails, the mate takes over, ensuring that service continues without interruption. One of the STPs primary functions is performing global title translations.

signaling

Communication between switches, or switches and end points, to set-up, manage, and tear-down calls. Signaling methods include dial pulse (rotary dial), dual-tone multifrequency (DTMF) (touch-tone), and digital ”packet” technology (ISDN, SS7)

Signaling Connection Control Part

A level of common Channel Signaling No. 7 (CCS7) layered protocol. The main functions of the SCCP include the transfer of signaling units with or without the use of a logical signaling connection at the provisioning of flexible translations (GTT) for different applications.

SIOTRK

Shared Interoffice Trunk table. Defines trigger criteria for *Shared_Interoffice_Trunk*.

SMS

service management system

SOC

software optionality control

SPECDIG

Specific Digit String table. Defines trigger criteria for *Specific_Digit_String*.

SPECFEAT

Specific Feature Code table. Defines trigger criteria for *Specific_Feature_Code*.

SS7

signaling system number 7

SSP

service switching point

STDPRT

standard pretranslator

STDPRTCT

Standard Pretranslator Control table. The first table indexed for digit pretranslation when the incoming or two-way trunk group associated with the call is assigned a standard pretranslator name. Call processing then indexes the appropriate STDPRT subtable. The CAINGRP option identifies the CAIN group for address-based subscription.

STP

signal transfer point

STR-Connection

A connection that is made from an SSP to an IP over an ISDN IP interface in response to a **`Send_To_Resource`** message.

switchHookFlash event

switch hook flash event. Allows a terminator to transfer or redirect the calling party to a third party.

switchHookFlashLegs

switchHookFlashLegs enabled extension parameter. This parameter indicates which legs the SSP monitors for a switchHookFlash.

SUS

CCS7 suspend message

TANDMRTE

Tandem Routing table. Provisions routing through tandem switches within the IEC network to reach the required terminating switch.

TCAP

transaction capabilities application part

TCM

terninating call model

TDP

trigger detection point

TERMATT

Termination Atempt table. Defines trigger criteria for *Termination_Attempt*.

TERMRTE

Termination Routing table. Provisions routing to a terminating network directly connected to the current network.

TNS

transit network selector

Transaction Capabilities Application Part

A service that provides a common protocol for remote operations across the Common Channel Signaling No. 7 (CCS7) network. The protocol consists of message formatting, content rules, and exchange procedures.

translation verification

A diagnostic tool that allows the operating company to access and simulate a telephone call in software and display the tables and tuples used to establish the lines, trunks, or positions to which a call is routed.

TRAVER

translation verification

TRID

transaction identifier

trigger

A trigger defines the actions taken once trigger criteria is met at a TDP on the SSP.

trigger criteria

Trigger criteria defines the conditions used to determine whether a particular call will trigger. A trigger may contain criteria of one or more trigger criteria types. In order for a call to trigger, all trigger criteria must be met.

trigger detection point

A point in basic call processing, as modeled by the basic call model (BCM), which identifies when a service control point (SCP) can receive notification of a given event and influence subsequent call processing. TDPs are located at transitions between points in call (PIC).

trigger tables

Trigger tables store information about the advanced intelligent network (AIN) application, triggers, trigger detection points, trigger criteria, transport protocol.

triggering

The process where a call indicates that it requires advanced intelligent network (AIN) service(s).

TRKGRP

Trunk Group table. Assigns a CAIN group to a particular agent.

UNIPROF

Universal Profile table. This table stores the possible profiles for different ANI values. When an ANI is received it is validated in table ANIVAL and then indexes this table to find the profile for the received ANI number.

VAMP

variable AIN messaging platform

VAMPTRID

VAMP Transaction Identifiers table. Provisions the key resources used in Carrier AIN messaging, including transaction and component identifiers and message buffers.

VPN

virtual private network

Ordering information

Use the following table for ordering Nortel Networks NTPs (Northern Telecom Publications) and Product Computing-Module Loads (PCLs):

Type of product	Source	Phone	Cost
Technical documents (paper or CD-ROM)	Nortel Networks Product Documentation	1-877-662-5669	Yes
Individual NTPs (paper)	Merchandising Order Service	1-877-662-5669	Yes
Marketing documents	Sales and Marketing Information Center (SMIC)	1-800-4NORTEL (1-800-466-7835)	No
PCL software	Nortel Networks	Consult your Nortel Networks sales representative	Yes

When ordering publications on CD

Please have the CD number and software version available, for example, **HLM-2621-ENCDRPDF 02.02**.

When ordering individual paper documents

Please have the document number and name available, for example, **297-2621-001, UCS DMS-250 Master Index of Publications**.

When ordering software

Please have the eight-digit ordering code, for example, **UCS00014**, as well as the ordering codes for the features you wish to purchase. Contact your Nortel Networks representative for assistance.

Digital Switching Systems

UCS DMS-250

NetworkBuilder Application Guide,
Volume 5 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.