# Troubleshooting Cisco Unity Express Integrated Voice Mail Features

This application note describes the debugging and tracing aspects of the Cisco Unity Express voice mail application for Cisco IP Communications (IPC) Express.

Specific troubleshooting sections presented in this application note are as follows:



## Common Voice Mail show Commands

Although tracing is one of the key troubleshooting tools in Cisco Unity Express, it is not always necessary. Many of the **show** commands are sufficient to identify certain problems. The following sections discuss the most common **show** commands you can use to pinpoint trouble spots:

# Listing Mailboxes with Usage Statistics

The **show voicemail mailboxes** command gives a very helpful summary of each mailbox. Although you can display the same data for each mailbox in the graphical user interface (GUI), no GUI window provides the same comprehensive system summary that is given by the output of this command-line interface (CLI) command. The **show voicemail mailboxes** command lists mailbox statistics in table format and contains the following fields:

- Total number of messages
- Number of new and saved messages
- Total time used for the mailbox
- Total mailbox size and the percentage of the mailbox used

The following example shows sample output of the **show voicemail mailboxes** command. In this output, User U1's mailbox does not contain any messages, but still the message time (MSGTIME) associated with the mailbox is given as 9 seconds. That is because User U1 has recorded one or more greetings, and the time for the greetings is counted as part of the total mailbox message time. However, spoken names are not considered part of the mailbox time, because they are stored in Lightweight Directory Access Protocol (LDAP) rather than the SQL database.

```
Cue# show voicemail mailboxes
OWNER                          MSGS NEW   SAVED   MSGTIME MBXSIZE   USED
"useru2"                         4    2     2       60      3000      2 %
"useru1"                         0    0     0       9       3000      1 %
"group1"                         0    0     0       0       3000      0 %
"group2"                         0    0     0       0       3000      0 %
"group3"                         0    0     0       0       3000      0 %
"group4"                         0    0     0       0       3000      0 %
"group5"                         0    0     0       0       3000      0 %
"useru3"                         0    0     0       0       3000      0 %
"useru4"                         0    0     0       14      3000      1 %
```

# Displaying Mailbox Details

To display the details of a particular mailbox, use the **show voicemail detail mailbox userid** command. As shown in the following example , it displays a mailbox's characteristics, including the message counts, date of creation, and last-modified time stamp.

```
Cue# show voicemail detail mailbox useru3
Owner:                                /sw/local/users/useru3
Type:                                 Personal
Description:                          useru3 mailbox
Busy state:                           idle
Enabled:                              disabled
Mailbox Size (seconds):               3000
Message Size (seconds):               60
Play Tutorial:                        false
Space Used (seconds):                 0
Total Message Count:                  0
New Message Count:                    0
Saved Message Count:                  0
Expiration (days):                    30
Greeting:                             standard
Created/Last Accessed:                Oct 10 2003 05:17:04 PDT
```

# Displaying Voice Mail Usage Details

To get a summary of the entire voice mail system, including the number of mailboxes and summary usage statistics, use the **show voicemail usage** command. Sample output of this command is shown in the following example:

```
Cue# show voicemail usage
personal mailboxes:                     67
general delivery mailboxes:             9
orphaned mailboxes:                     0
capacity of voicemail (minutes):        6000
allocated capacity (minutes):           3800.0
message time used (seconds):            31
message count:                          2
average message length (seconds):       15.5
greeting time used (seconds):           115
greeting count:                         11
average greeting length (seconds):      10.454545454545455
total time used (seconds):              146
total time used (minutes):              2.433333396911621
percentage used time (%):               1
```

# Displaying System Licenses

If you are experiencing problems creating new users, groups, or mailboxes, check the license installed on the Cisco Unity Express system, and make sure that the system limits are not exceeded.

The maximum amount of voice mail that you can store on the system is a licensed feature. On an NM-CUE system, it is 100 hours, and on the AIM-CUE it is 14 hours for the 1-GB flash card. You can allocate this total number of voice mail hours among your mailboxes in any manner suitable to your needs. Although this is the default system operation, there is no requirement to divide the amount of time equally among all mailboxes. As soon as the total of all the allocated voice mailboxes' storage equals the maximum allowed system storage, you cannot create any more mailboxes, even though you might not yet have reached the mailbox limit.

Sample output for the **show software licenses** command is shown in the following example. If you have exceeded the number of mailboxes allowed by the license, you cannot create more until the license is upgraded. The number of users allowed on a Cisco Unity Express system is typically double the number of mailboxes specified by the license.

```
Cue# show software licenses
Core:
 - application mode: CME
 - total usable system ports: 4
Voicemail/Auto Attendant:
 - max system mailbox capacity time: 480
 - max general delivery mailboxes: 15
 - max personal mailboxes: 50
Languages:
 - max installed languages: unlimited
 - max enabled languages: 1
```

## Displaying Default Voice Mailbox System Settings

The **show voicemail limits** command shows the default system settings for different voice mail parameters. These settings are applied to every new mailbox created. Although the administrator can override these settings on a per-mailbox basis, these system settings are the defaults applied. Sample output of this command is shown in the following example.

```
Cue#show voicemail limits
Default Mailbox Size (seconds):         420
Default Caller Message Size (seconds):  60
Maximum Recording Size (seconds):       900
Default Message Age (days):             30
System Capacity (minutes):              480
Default Prompt Language:                en_US
Operator Telephone:                     5700
```

# Troubleshooting Mailbox GUI Configuration Problems

You can select to create mailboxes manually when users or groups are added, or you can add mailboxes in bulk fashion during the Cisco Unity Express Initialization Wizard when user definitions are imported from Cisco CME. The following issues can occur in the GUI:


- Mailbox Configuration, page 192
- Orphaned Mailboxes, page 193
- User and Group Configurations, page 194
- General Delivery Mailboxes, page 197


## Mailbox Configuration

To troubleshoot mailbox configuration problems in the GUI, the **trace webInterface mailbox all** command is useful.

The following example shows sample output of this command where the user attempts to access a mailbox that does not exist. The administrator creates a new mailbox, deletes another mailbox, and then verifies that the new mailbox has been created successfully by checking the mailbox GUI page. Other mailboxes are also shown on the GUI page. The PERSONAL_00000000000000000000009 text shown in the trace is an internal unique identifier for each mailbox. You can use this identifier to isolate operations and events that pertain to a particular mailbox in the system.

```
Cue# trace webInterface mailbox all
Cue# show trace buffer tail

2607 02/28 19:39:30.308 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000006
2178 02/28 19:40:23.483 webI mail 1 Error while reading Mailbox by owner. no such
attribute
2178 02/28 19:40:23.563 webI mail 2 Created mailbox:
PERSONAL_00000000000000000000009
2178 02/28 19:40:23.564 webI mail 2 Proceeding to Populate other values for new mailbox
2178 02/28 19:40:23.564 webI mail 2 Saving Mailbox
2178 02/28 19:40:23.625 webI mail 2 Finished populating other values for new mailbox
2667 02/28 19:40:23.893 webI mail 1 Filter for mailboxes : (&(ownerDn=*))
2667 02/28 19:40:23.899 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000004
```

```
2667 02/28 19:40:23.902 webI mail 1 Attempt to read mailbox
  GENERAL_00000000000000000000000
2667 02/28 19:40:23.904 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000008
2667 02/28 19:40:23.906 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000002
2667 02/28 19:40:23.909 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000009
2667 02/28 19:40:23.918 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000005
2667 02/28 19:40:23.919 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000006
2672 02/28 19:40:39.660 webI mail 3 Going to sysdb to delete Mailbox for :
  /sw/local/users/test
2672 02/28 19:40:39.699 webI mail 3 Deleted mailbox: /sw/local/users/test Mailbox
  deleted
2672 02/28 19:40:39.699 webI mail 3 Returned from sysdb. Mailbox delete for :
  /sw/local/users/test
2672 02/28 19:40:39.700 webI mail 1 Filter for mailboxes : (&(ownerDn=*))
2672 02/28 19:40:39.700 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000004
2672 02/28 19:40:39.703 webI mail 1 Attempt to read mailbox
  GENERAL_00000000000000000000000
2672 02/28 19:40:39.705 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000008
2672 02/28 19:40:39.708 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000002
2672 02/28 19:40:39.711 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000005
2672 02/28 19:40:39.712 webI mail 1 Attempt to read mailbox
  PERSONAL_00000000000000000000006
```

# Orphaned Mailboxes

An orphaned mailbox is a mailbox without an owner. This can occur when the owner (the user profile) is deleted, but the mailbox remains in the system, now unassociated with a valid user. When a user is deleted using the Cisco Unity Express GUI, the attached mailbox is automatically deleted as well, so the orphaned mailbox condition does not occur if you do system administration via the GUI. The CLI does not operate this way, however. When the user is deleted via the Cisco Unity Express CLI, the mailbox remains in the system (unless explicitly deleted with a separate command), and it appears as an orphaned mailbox.

To check the status of all mailboxes in the system, use the **show voicemail usage** command. the following example shows sample output for this command.

```
Cue# show voicemail usage
personal mailboxes:                     100
general delivery mailboxes:             9
orphaned mailboxes:                     1
capacity of voicemail (minutes):        6000
allocated capacity (minutes):           5450.0
message time used (seconds):            1183
message count:                          38
average message length (seconds):       31.13157894736842
greeting time used (seconds):           63
greeting count:                         8
average greeting length (seconds):      7.875
total time used (seconds):              1246
total time used (minutes):              20.766666412353516
percentage used time (%):               0
```

The preceding example shows that one orphaned mailbox exists in this system. To find out the details of the orphaned mailbox, use the **show voicemail mailbox orphaned** command (see the following example).

```
Cue# show voicemail mailbox orphaned
OWNER                  TYPE           ORPHANED TIME
"Bill"               Personal        Jul 28 2004 18:18:06 PST
```

In the GUI, orphaned mailboxes are shown in the **Voice Mail > Mailboxes** window. The mailbox is marked as orphaned. An orphaned mailbox does not affect the overall Cisco Unity Express operation, but it does count as one of the licensed mailboxes. An orphaned mailbox indicates that a mailbox defined in the system cannot be used by any of the subscribers, so it is desirable to reassociate orphaned mailboxes with a valid user or remove them. To delete an orphaned mailbox in the GUI, go to the **Voice Mail > Mailboxes** window, and select the orphaned mailboxes to delete. Or create a user with the same user ID as the one the orphaned mailbox belongs to. The system prompts you to reassociate the mailbox with this user.

# User and Group Configurations

Users and groups are the owners of personal and general delivery mailboxes, respectively. The following are some problems you might encounter with users and groups:

- A deleted user still appears in the GUI
- A user is not associated with a voice mailbox
- A user is not associated with a group

Use the **trace webinterface user all** command or the **trace webinterface group all** command to troubleshoot these problems.

## Deleted User Appears in the GUI

To troubleshoot a deleted user who still appears in the GUI, in the CLI first enter **trace webinterface user all** to turn on the traces to be monitored. In the GUI, click the deleted user to prompt more detailed information. A new window will appear. A normal web interface trace for a valid user is given in the following example.

```
Cue# trace webinterface user all
Cue# show trace buffer
2604 03/01 01:03:21.383 webI user 1 Resetting UserForm
2604 03/01 01:03:21.750 webI user 1 Start reading user Bill
2604 03/01 01:03:21.753 webI user 1 End reading user Bill
```

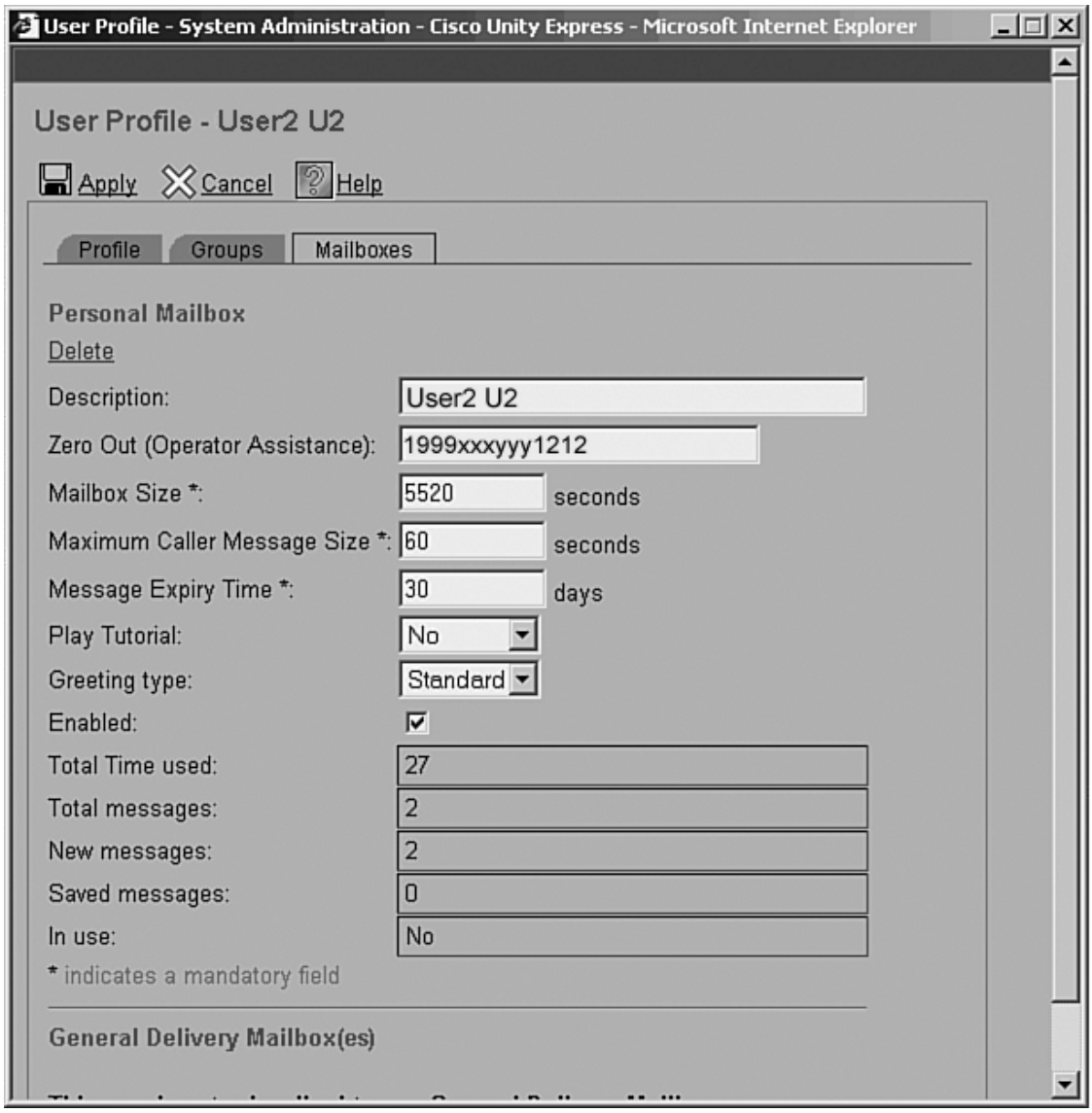
Because the user has been deleted, it is likely that no user window pops up and no trace is collected. This indicates that the user is no longer a valid user in the system and that the database does not have any record of that person. Use the **show voicemail users** command to verify whether the user is in the user list. Enter the to **user** *name* **delete** to delete the unwanted user.

## User Is Not Associated with a Voice Mailbox in the GUI

It is possible that you have configured a user with a voice mailbox, but the voice mailbox does not show up in the GUI. Verify with the **show voicemail users** command that the user appears in the system. If the voice mail user is in the system, but the mailbox does not show up in the GUI, the problem is likely a GUI issue.

To troubleshoot this problem, enable trace webinterface user all and clear the trace buffer. In the GUI, select **Configure > Users** and then select the user whose mailbox does not show up. A User Profile window opens. Click the **Mailboxes** tab. The mailbox information will appear in the same window, as shown in Figure 49.

***Figure 49 User Mailbox***




If no mailbox information appears, go to the CLI and examine the trace buffer. The following example shows a normal trace showing a user with a legitimate mailbox.

```
Cue# trace webinterface user all
Cue# show trace buffer
3125 03/01 04:57:48.737 webI user 1 Start reading user Bill
3125 03/01 04:57:48.740 webI user 1 End reading user Bill
3125 03/01 04:57:48.943 webI user 1 1 entries read for authorized mailboxes
```

If no such trace is collected, delete and re-add the user. Also add the voice mailbox, and verify that the problem has been resolved.

## User Is Not Associated with a Group in the GUI

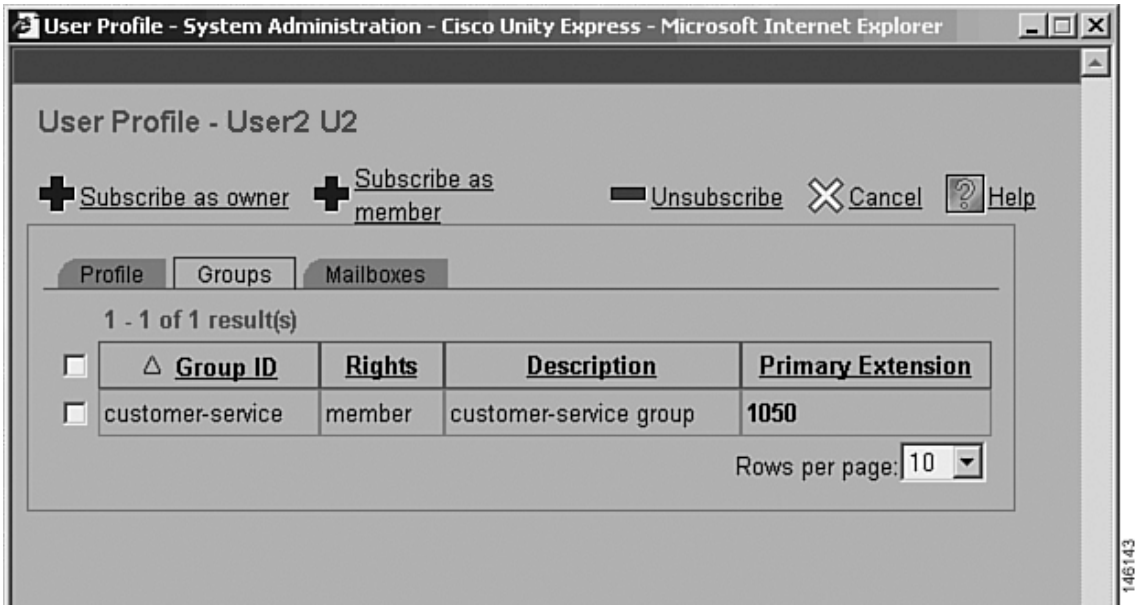
For a user who has been configured as a member of a group, but the group association does not show up in the GUI, the first step is to determine whether this is a system issue or a GUI issue. The useful CLI commands are as follows:

- **show users**—Lists all the users in the system
- **show groups**—Lists all the groups in the system
- **show group detail groupname** *group_name*—Lists all the members of the particular group

Execute all these commands on your system and note the output. If the voice mail user shows up in the group, but in the GUI the user is not associated with the group, check the GUI interface.

To troubleshoot, enable the **trace webinterface group all** command and clear the trace buffer. In the GUI, select **Configure > Users**, and then click the user under investigation. The User Profile window opens. In this window, click the **Groups** tab, as shown in Figure 50, and then go to the CLI to examine the trace buffer.

***Figure 50 User's Group Membership***




The following example shows a normal trace showing a user belonging to a legitimate group.

```
Cue# trace webinterface group all
Cue# show trace buffer
3237 03/01 13:01:15.520 webI grup 1 Start reading group Sales
3237 03/01 13:01:15.521 webI grup 1 End reading group Sales
3237 03/01 13:01:15.521 webI grup 1 Reading owners for Sales
3237 03/01 13:01:15.522 webI grup 1 Reading members for Sales
3237 03/01 13:01:15.522 webI grup 1 Bill
```

If no such trace is collected, delete and readd the user. Assign the user to a group, and verify that the problem has been resolved.

# General Delivery Mailboxes

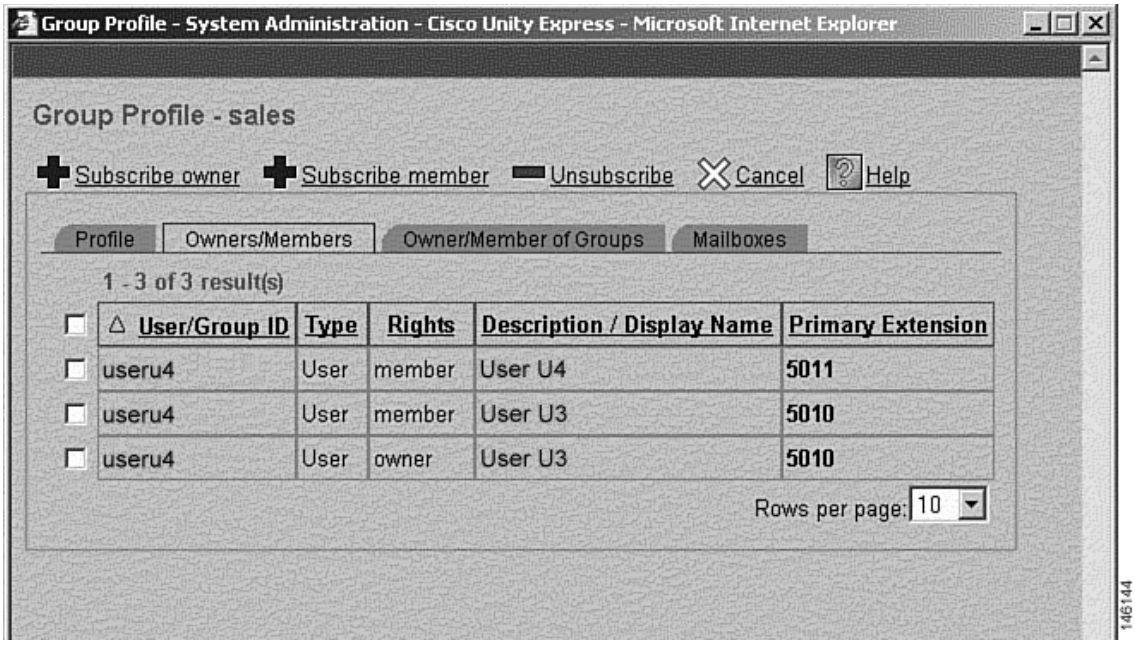
General delivery mailboxes (GDMs) are similar to personal mailboxes. However, they are associated with groups (as opposed to individual users), and more than one user (member) can access them, although only one user can log in at a time.

Problems with GDMs can be configuration-related. When a GDM gets a message, it is not copied to every member's personal mailbox; it resides in the GDM. Any member of the group can check to see if there are new messages in a GDM by pressing 9 after logging in to his or her personal mailbox. Note that an owner of the group does not have access to the GDM unless that user is also a member of the group.

One of the most common problems users experience with GDMs is that when they log in to their personal mailboxes, they do not have the option to log in to the GDMs they are supposed to be a member of. This might be because the user is not a member of the group associated with the GDM. To ensure that a user is part of the group in question, follow the steps described next with either the GUI or the CLI.

Go to the Group Profile GUI window by selecting **Configure > Groups**, click the group you are interested in, and then choose the **Owners/Members** tab. This window is shown in Figure 51.

***Figure 51 Group Profile***




You can also execute the **show voicemail detail mailbox** CLI command to see which GDM a particular user is a member of. This command is executed for user useru3 in the following example. The output shows that he is part of the sales GDM.

```
Cue# show voicemail detail mailbox useru3
Owner:                                  /sw/local/users/useru3
Type:                                   Personal
Description:                            useru3 mailbox
Busy state:                             idle
Enabled:                                true
Mailbox Size (seconds):                 420
Message Size (seconds):                 60
Play Tutorial:                          true
Space Used (seconds):                   0
```

```
Total Message Count:                    0
New Message Count:                      0
Saved Message Count:                    0
Expiration (days):                      30
Greeting:                               standard
Created/Last Accessed:                  Dec 04 2003 04:09:25 EET

Owner:                                  /sw/local/groups/sales
Type:                                   General Delivery
Description:                            sales mailbox
Busy state:                             idle
Enabled:                                true
Mailbox Size (seconds):                 420
Message Size (seconds):                 60
Play Tutorial:                          true
Space Used (seconds):                   0
Total Message Count:                    0
New Message Count:                      0
Saved Message Count:                    0
Expiration (days):                      30
Greeting:                               standard
Created/Last Accessed:                  Dec 04 2003 04:08:11 EET
```

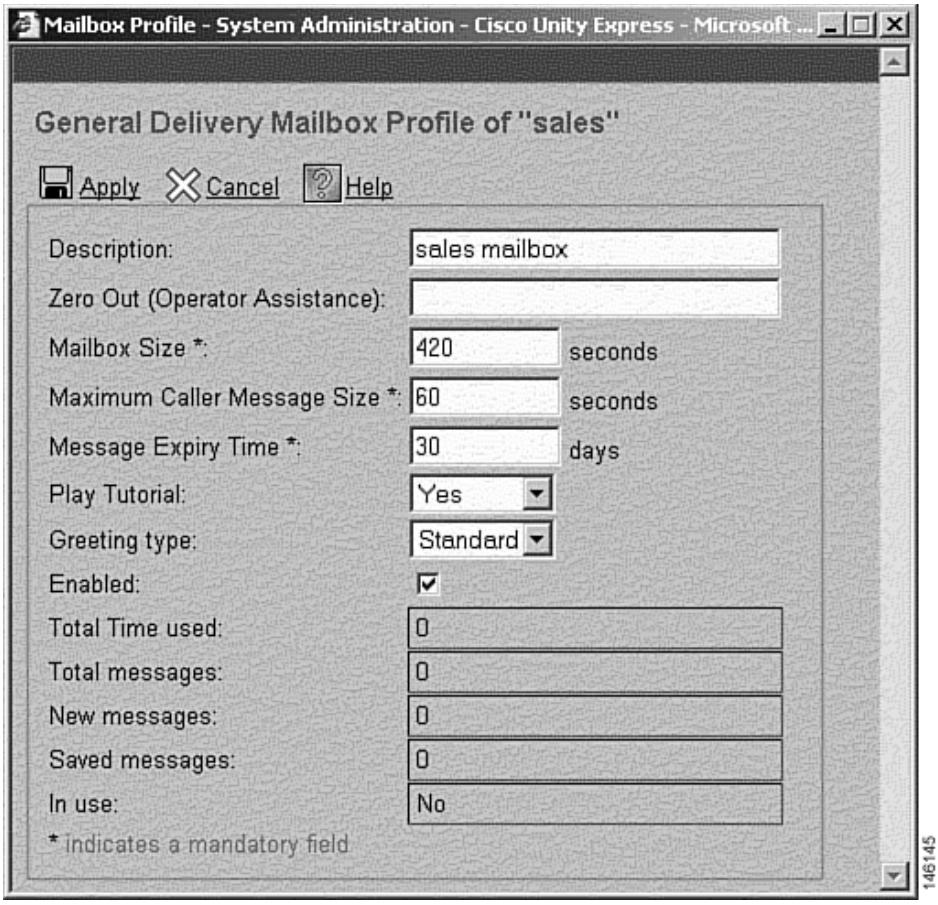
**Note** Note that the first part of the mailbox information in the following example describes useru3's personal mailbox (of which he is the owner). The following section of the output describes his associations with GDMs. The example shows that he is associated with the sales GDM, which is owned by a group called sales.

The **show voicemail detail mailbox** command can also be used to show the details of the sales GDM, as shown in the following example.

```
Cue# show voicemail detail mailbox sales
Owner:                                  /sw/local/groups/sales
Type:                                   General Delivery
Description:                            sales mailbox
Busy state:                             idle
Enabled:                                true
Mailbox Size (seconds):                 420
Message Size (seconds):                 60
Play Tutorial:                          true
Space Used (seconds):                   0
Total Message Count:                    0
New Message Count:                      0
Saved Message Count:                    0
Expiration (days):                      30
Greeting:                               standard
Created/Last Accessed:                  Dec 04 2003 04:08:11 EET
```

You can check this same information in the GUI by selecting **Configure > Groups**, clicking the group of interest (sales, in this example), and then selecting the Mailboxes tab. This window is shown in Figure 52. By clicking the **Owner/Member** tab on this window, you can list the users who currently have access to this GDM.

*Figure 52 General Delivery Mailbox*




Apart from the **show** commands described in this section, you can also use all mailbox- and message-related **trace** commands to troubleshoot GDMs. The CLI command **show group detail groupname** *name* also has very helpful output.

# Troubleshooting Cisco CME and Cisco Unity Express Integration

Cisco CME is the system's call control engine. All calls coming into voice mail to leave or retrieve messages involve an interaction (via SIP) between Cisco CME and Cisco Unity Express. Cisco CME integration with Cisco Unity Express voice mail includes the following activities:

- Forwarding of a call to Cisco Unity Express on busy or ring-no-answer
- Dual-tone multifrequency (DTMF) relay to interact with the voice mail system
- A call transfer from the automated attendant (AA) that forwards into voice mail
- Turning the message waiting indicator (MWI) on or off

Call forwarding into voice mail requires passing the right original called number to the voice mail system so that the correct user's mailbox is selected to play the greeting and leave a message. DTMF relay encompasses the conversion of IP phone keypad digit presses sent via Skinny Client Control Protocol (SCCP) messages from the phone to Cisco CME call control, and from there to Cisco Unity Express via SIP NOTIFY messages. Call transfer from the AA is implemented as a blind transfer using the SIP BYE/Also message sequence.

The most common problems encountered with calls into voice mail are caused by incorrect or incomplete configuration:

- The wrong mailbox selection or no mailbox selection on call forward to voice mail (which might also result in the wrong greeting being played to the caller)
- Digit presses from the phone do not get the required response (in other words, DTMF messages do not reach the application)
- Call transfers from the AA do not work
- Fast-busy tone is heard for calls going to voice mail
- MWI does not operate properly

Correcting each of these problems is described in the following sections.

# Wrong Mailbox Selection or Unexpected Greeting

Calls to a Cisco CME IP phone can arrive from two different sources:

- A call received from an internal extension, where the dialed number is the phone's extension
- A call received from the PSTN or from another site (routed via H.323 or SIP), where the called number may be the phone's E.164 number

Therefore, depending on what number was dialed, the original called number (also called the last redirecting number in the case of multiple forwards) sent from Cisco CME to Cisco Unity Express is either the internal extension of the called IP phone or the complete E.164 number dialed to reach the IP phone. Cisco Unity Express voice mail must be able to recognize both types of numbers.

Furthermore, when a subscriber calls Cisco Unity Express voice mail directly (for example, to check voice mail), the calling number is used to identify the appropriate mailbox. This calling number can also be an internal extension or a fully qualified E.164 phone number (if the **dialplan-pattern** command is configured on Cisco CME). Therefore, it is important that the calling number be delivered correctly to Cisco Unity Express for these calls. Various PSTN and Cisco CME configurations can transform (via digit manipulation) the calling number, which can cause the voice mail system to play the wrong prompt. The following sections demonstrate how to identify and rectify these problems.

## Call from an Internal Extension to Voice Mail

The following example shows the output of the debug ccsip messages command on Cisco CME when a call to extension 5010 arrives from internal extension 7010. A SIP INVITE message is sent from Cisco CME to Cisco Unity Express. 7010 is the calling number, used to recognize whether the caller is a subscriber on the system. Subsequently, when the called person listens to the message left in voice mail, this Calling Number field determines which prompts are played. If the calling number of the original call (extension 7010) matches a valid subscriber with a mailbox on Cisco Unity Express, the voice mail system plays "Extension 7010 sent message 1...." If the caller has a spoken name recorded in the system, the system plays "Spoken name sent message 1...." On the other hand, if the calling number does not match any user in the system, the voice mail plays "An unknown caller sent message 1..." or "7010 sent message 1..." if the **voicemail callerid** configuration on Cisco Unity Express is enabled.

```
Ccme# debug ccsip messages
INVITE sip:6800@192.168.0.2:5060 SIP/2.0
Via: SIP/2.0/UDP  192.168.0.1:5060
From: "7010" <sip:7010@192.168.0.1>;tag=691AE6E4-223C
To: <sip:6800@192.168.0.2>
Call-ID: E5D39E6A-8FC011D7-9025DAEC-459632B0@192.168.0.1
CSeq: 101 INVITE
```

```
Max-Forwards: 6
Remote-Party-ID: <sip:7010@192.168.0.1>;party=calling;screen=no;privacy=off
Timestamp: 1054070868
Contact: <sip:7010@192.168.0.1:5060>
Diversion: <sip:5010@192.168.0.1>;reason=no-answer;counter=1
```

The voice mail pilot number (extension 6800 in the preceding example) is shown in the To: field of the SIP INVITE message. The most important part of this trace is the Diversion field. It contains the number (extension 5010) that the caller originally called (extension 7010, which shows up in the From: field) before the call was forwarded to voice mail. The value in the Diversion header (5010) is the mailbox number for which the message will be left. Also, the SIP message has a Reason field, which indicates whether the called party did not answer the call or was busy on the phone. Some voice mail systems play different greetings depending on these reason codes, but Cisco Unity Express voice mail currently plays the same prompt for both situations.

There is a case in which a call is forwarded more than once before it reaches voice mail. For example, if extension 5010 forwards its calls to 5012 on busy or ring-no-answer and then 5012 forwards all calls to voice mail, the Diversion header contains 5012 instead of 5010. This can create confusion, because the caller thinks he called 5010 and wants to leave a message for 5010, but instead he gets the voice mail greeting for the person at extension 5012. Cisco CME uses the last redirecting number rather than the original called number for mailbox selection.

## Call Using an E.164 Number to Voice Mail

The following example shows a slightly different variation of a call-forward-to-voice-mail situation. The call comes from the PSTN and goes to an IP phone in E.164 number format (4xx.5yy.5010). The original Called Number field in the Diversion header is populated with this E.164 number. If the E.164 field is not configured for this subscriber within the Cisco Unity Express voice mail application, this call does not reach a mailbox, and the caller does not hear the right greeting.

**Note** In the following debugs, some output has been removed to show only the most important parts of the trace.

```
Ccme# debug ephone state
Ccme# debug ccsip messages
*Mar  4 00:19:34.566: ephone-2[2]:SetCallState line 1 DN 1 chan 1 ref 7 TsRingIn
*Mar  4 00:19:34.566: ephone-2[2]:Call Info DN 1 line 1 ref 7 called 4xx5yy5010
  calling 5702 origcalled 4085255010 calltype 1
*Mar  4 00:19:34.566: ephone-2[2]: No-Name calling
*Mar  4 00:19:34.566: ephone-2[2]: 5010
*Mar  4 00:19:34.566: ephone-2[2]:Ringer Outside Ring On
*Mar  4 00:19:44.566: ephone-2[2]:SetCallState line 1 DN 1 chan 1 ref 7 TsOnHook
*Mar  4 00:19:44.566: dn_tone_control DN=1 chan 1 tonetype=0:DtSilence onoff=0
  pid=137
*Mar  4 00:19:44.566: ephone-2[2]:SpeakerPhoneOnHook
*Mar  4 00:19:44.566: ephone-2[2]:Ringer Off
*Mar  4 00:19:44.570: Sent:
INVITE sip:5800@a.3.6.129:5060 SIP/2.0
Via: SIP/2.0/UDP  a.3.6.29:5060
From: "5702" <sip:5702@a.3.6.29>;tag=F85274C-2696
To: <sip:5800@a.3.6.129>
Date: Thu, 04 Mar 1993 00:19:44 GMT
Remote-Party-ID: <sip:5702@a.3.6.29>;party=calling;screen=no;privacy=off
Timestamp: 731204384
Contact: <sip:5702@a.3.6.29:5060>
Call-Info: <sip:a.3.6.29:5060>;method="NOTIFY;Event=telephone-event;Duration=2000"
```

```
Diversion: <sip:4xx5yy5010@a.3.6.29>;reason=no-answer;counter=1
Expires: 180
```

If a call fails to get a greeting from the right mailbox, make sure that the recipient subscriber has the E.164 number field configured, as shown in the following example. This also can be done through the GUI's user profile page.

```
Cue# show user detail username useru3
Full Name:          User U3
First Name:         User
Last Name:          U3
Nickname:           User U3
Phone:              5010
Phone(E.164):       4xx5yy5010
Language:           en_US
```

If this field is not yet configured, populate it as follows:

```
Cue(config)# username useru3 phonenumberE164 4xx5yy5010
```

Sometimes, when a subscriber directly calls Cisco Unity Express to check voice mail, the person hears the "Enter your ID" prompt instead of the "Enter your password" prompt. The reason could be that the calling number is not delivered properly to the Cisco Unity Express application or the application doesn't recognize the number because of configuration mismatches.

## Digit Manipulation

Often Cisco Unity Express voice mail doesn't recognize a calling number because of digit manipulation done by the PSTN voice gateway router or the Cisco CME call control software.

One of the digit manipulation features is the Cisco CME **dialplan-pattern** command. If configured, Cisco CME call control promotes the extension of an IP phone to the fully qualified E.164 number by inserting the appropriate digits. For instance, extension 5010 is promoted to 4xx.5yy.5010.

In this case, the From field in the SIP INVITE from Cisco CME call control to Cisco Unity Express is populated with the E.164 number. For this to work properly, you must configure the E.164 field for the Cisco Unity Express subscribers, as shown in the preceding example. To check whether Cisco CME has dial plan patterns configured, use the **show telephony-service** command, as shown in the following example:

```
Ccme# show telephony-service
CONFIG (Version=3.0)
====================
Version 3.0ip source-address 10.10.10.1 port 2000
max-ephones 12
max-dn 48
max-conferences 8
huntstop
dialplan-pattern 1 4xx5yy5... extension-length 4
time-format 12
date-format mm-dd-yy
```

The debug traces shown in the following example demonstrate the calling number being populated with the E.164 number when the call is sent to the Cisco Unity Express voice mail application.

```
Ccme# debug ccsip messages
SIP Call messages tracing is enabled
c2600-ITS-NM#
Jun  9 08:55:04.214: Sent:
INVITE sip:5800@a.3.6.130:5060 SIP/2.0
Via: SIP/2.0/UDP  a.3.6.30:5060;branch=z9hG4bK373
```

```
From: <sip:4xx5yy5010@a.3.6.30>;tag=1F5DE9-996
To: <sip:5800@a.3.6.130>
Date: Sun, 09 Jun 2002 08:55:04 GMT
Call-ID: 6C827377-7ABD11D6-800FA41E-A3784778@a.3.6.30
Supported: timer,100rel
Min-SE:  1800
Cisco-Guid: 1767160890-2059211222-2148312094-2742568824
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, REFER, SUBSCRIBE, NOTIFY,
  INFO, UPDATE, REGISTER
CSeq: 101 INVITE
Max-Forwards: 6
```

Another digit manipulation feature that can convert the calling number is the Cisco IOS translation rule feature. Translation rules might be configured for the calling ephone-dn or for the dial peers governing the incoming PSTN or VoIP calls. Or a global translation rule might be translating calling numbers for every incoming VoIP call using the Cisco IOS **voip-incoming** command. This translation rule might convert the calling number to a different digit string that Cisco Unity Express does not recognize. In this case, correct the Cisco IOS configuration such that the translated calling number matches either the extension or the E.164 field value in the Cisco Unity Express user configuration for the subscriber.

# Troubleshooting DTMF—No Response for Digit Presses from Cisco Unity Express

During calls to either voice mail or the AA, the system might not respond to the caller's digit presses. This can be caused by an incorrect configuration on the SIP dial peer directing calls to Cisco Unity Express. It also might be caused by a failed SIP SUBSCRIBE/NOTIFY transaction.

The following example shows a working DTMF configuration for the SIP dial peer on the Cisco CME router to direct calls to Cisco Unity Express applications. Note the DTMF relay specified as sip-notify. Currently, this is the only DTMF relay option supported by Cisco Unity Express applications. Also note that voice activity detection (VAD) is turned off. The Cisco Customer Response Software (CRS) component does not detect DTMF correctly if VAD is enabled for calls.

```
dial-peer voice 6800 voip
 destination-pattern 5...
 session protocol sipv2
 session target ipv4:a.3.6.127
 dtmf-relay sip-notify
 codec g711ulaw
 no vad
```

In the SIP-NOTIFY DTMF relay mechanism, the Cisco Unity Express application subscribes internally to Cisco CME call control for telephony events (in this case, the event is the pressing of a digit) as soon as the call is established. Cisco CME accepts this subscription and sends a NOTIFY to Cisco Unity Express whenever a digit is pressed on the phone.

If the DTMF digits for a call to Cisco Unity Express do not take effect, the configuration of the SIP dial peer is the first place to inspect. There might be other causes of DTMF recognition problems, but the incorrect configuration of the dial peer is by far the most common mistake.

The trace shown in the following example demonstrates how a digit press on an IP phone is translated to a SIP-NOTIFY DTMF relay event to the application. For every KeypadButtonMessage from the phone, you can see a SIP-NOTIFY message going toward Cisco Unity Express. If these messages do not match, this indicates a failure in the Cisco CME code to translate digit presses to the application, effectively breaking DTMF relay.

```
Ccme# debug ephone detail
EPHONE detail debugging is enabled

Ccme# debug ccsip messages
SIP Call messages tracing is enabled
Ccme#
*Mar  4 00:34:55.649: ephone-2[2]:Call Info DN 1 line 1 ref 11 called 5700 calling
  5010 origcalled 5700 calltype 2
*Mar  4 00:34:55.649: ephone-2[2]: 5010 calling
*Mar  4 00:34:55.649: ephone-2[2]: No-Name
*Mar  4 00:35:01.949: ephone-2[2]:KeypadButtonMessage 1
*Mar  4 00:35:01.949: SkinnyGetCallState for DN 1 chan 1 CONNECTED
*Mar  4 00:35:01.949: called DN -1 chan 1, calling DN -1 chan 1 phone 2 s2s:0
*Mar  4 00:35:01.949: Sent:
NOTIFY sip:5700@a.3.6.129:5060 SIP/2.0
Via: SIP/2.0/UDP  a.3.6.29:5060
From: <sip:4085255010@a.3.6.29>;tag=F930B80-674
To: "Cisco SIP Channel3" <sip:5700@a.3.6.129>;tag=c2e25d00-284
CSeq: 102 NOTIFY
Event: telephone-event;rate=1000
Contact: <sip:4085255010@a.3.6.29:5060>
Content-Length: 10
Content-Type: audio/telephone-event

0x01800064
*Mar  4 00:35:01.949: Received:
SIP/2.0 200 OK
Via: SIP/2.0/UDP  a.3.6.29:5060
From: <sip:4085255010@a.3.6.29>;tag=F930B80-674
CSeq: 102 NOTIFY

*Mar  4 00:35:05.225: ephone-2[2]:KeypadButtonMessage 5
*Mar  4 00:35:05.225: SkinnyGetCallState for DN 1 chan 1 CONNECTED

NOTIFY sip:5700@a.3.6.129:5060 SIP/2.0
Via: SIP/2.0/UDP  a.3.6.29:5060
From: <sip:4085255010@a.3.6.29>;tag=F930B80-674
To: "Cisco SIP Channel3" <sip:5700@a.3.6.129>;tag=c2e25d00-284
CSeq: 103 NOTIFY
Contact: <sip:4085255010@a.3.6.29:5060>
Content-Type: audio/telephone-event

0x05800064
```

The highlighted hexadecimal number represents the digits pressed for the SIP protocol NOTIFY message. Table 15 shows the mapping of hex values to digit, buttons on a phone. The important thing is the first byte in the hex value. The remaining bytes can vary depending on the configuration on the calling gateway.

***Table 15 Digit-to-SIP Protocol Value Mapping***

| Digit | SIP Protocol Value |
|---|---|
| 0 | 0x00800064 |
| 1 | 0x02800064 |
| 2 | 0x02800064 |
| 3 | 0x03800064 |
| 4 | 0x04800064 |
| 5 | 0x05800064 |

*Table 15 Digit-to-SIP Protocol Value Mapping*

| Digit | SIP Protocol Value |
|---|---|
| Digit | SIP Protocol Value |
| 6 | 0x06800064 |
| 7 | 0x07800064 |
| 8 | 0x08800064 |
| 9 | 0x09800064 |
| * | 0x0A800064 |
| # | 0x0B800064 |

The trace segment shown in the preceding example ensures that Cisco CME is correctly translating and sending the digits to the Cisco Unity Express application. This does not necessarily mean that digits are reaching the correct component within Cisco Unity Express to which the call is established. Other components of the Cisco Unity Express software must work properly for the digits to take effect. For example, the CRS SIP stack must handle the incoming SIP-NOTIFY event properly, and send the digit to the CRS application software. If this application software is using the voice browser component, the digits must reach the voice browser properly. As described in the application note entitled "Troubleshooting Cisco Unity Express Automated Attendant," many CRS step customizations change the behavior of DTMF handling in applications. Be sure to verify those customizations if the application script was not supplied as part of the Cisco Unity Express software.

# Troubleshooting Automated Attendant Transfers

It might happen that a caller interacting with the Cisco Unity Express AA presses the extension of the user he wants to speak to, but then the call to the extension is unsuccessful, and the caller hears overflow tone instead of a ringing phone. When a caller chooses an extension in the AA, the Cisco Unity Express AA application does a blind transfer operation. This operation might fail for various reasons.

The simplest reason is that the extension chosen does not exist on the Cisco CME router. Other reasons could be a mistake in either the Cisco IOS, Cisco Unity Express software, or Class of Restriction (COR) configuration.

For example, suppose a call goes from the PSTN to the AA, and the caller presses the extension of a phone in the break room. A COR can be configured for the ephone-dn associated with the break room phone such that it can receive calls only from internal phones and not from the PSTN. Check for this kind of configuration before looking more deeply into troubleshooting AA call transfer failures.

Cisco Unity Express uses the SIP BYE/Also message to instruct Cisco CME call control software to execute a call transfer to a requested extension. The SIP BYE/Also mechanism triggers a blind transfer. This means that there are no intermediate steps between initiating the transfer, making sure the destination phone rings, and completing the transfer. A blind transfer is a one-step mechanism in which control of the call is relinquished at the same time that the call is redirected to the new extension, so the Cisco Unity Express application has no further control over the call after the transfer has been executed. So if the caller dials a nonexistent extension or an extension where the phone isn't registered, the caller hears fast-busy (also called overflow) tone.

In the trace shown in the preceding example, assume that the call was to the Cisco Unity Express AA (instead of voice mail) and that the caller selected dial-by-extension and then pressed the extension of the destination user to talk to.

The trace shown in the following example shows how Cisco Unity Express executes the transfer using the BYE/Also mechanism. The BYE message is the SIP message to disconnect the original call. The BYE message additionally includes a header called Also, which further tells the Cisco CME call control software to set up a new call to the number specified in the message. This sequence effectively transfers the call.

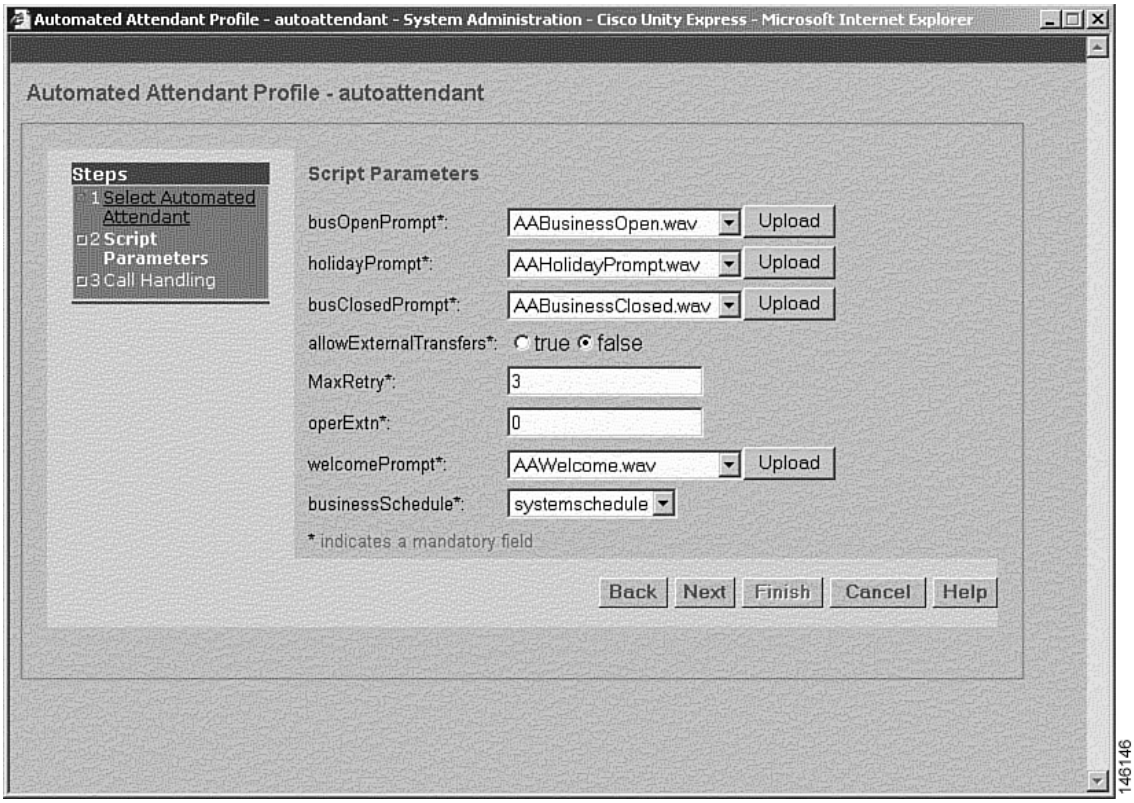
```
Ccme# debug ccsip messages
*Mar  4 00:35:13.769: Received:
BYE sip:4xx5yy5010@a.3.6.29:5060 SIP/2.0
Via: SIP/2.0/UDP a.3.6.129:5060
From: <sip:5700@a.3.6.129>;tag=c2e25d00-284
To: "5010" <sip:4xx5yy5010@a.3.6.29>;tag=F930B44-165C
Call-ID: E7B1CE85-174E11CC-802B8935-B1821821@a.3.6.29
CSeq: 53 BYE
User-Agent: Jasmin UA / ver 1.1
Max-Forwards: 50
Content-Length: 0
Also: <sip:5012@0.0.0.0;user=phone>
```

The Cisco Unity Express AA before release 2.1 allowed transfers to any digit string. It did not mandate that the destination telephone number must be available or configured in the Cisco Unity Express system's LDAP database.

Using the mechanism just described, a local PSTN caller can potentially call the Cisco CME AA and dial a long-distance or even an international destination telephone number, and the call will be transferred to that number. This effectively executes a hairpinned call (in from the PSTN and back out to the PSTN on the same router) and charges the call to the business hosting Cisco Unity Express. This process can result in toll fraud. If you are using Cisco Unity Express 2.0 or earlier, you can avoid this by configuring the COR feature on the dial peers pointing to long-distance PSTN numbers so that only select phones can call these numbers. In such cases, it is especially important to consider COR configurations while troubleshooting AA call transfer failures.

The COR configuration introduces some complexity into Cisco CME operation. The other option is to use Cisco Unity Express 2.1 or higher, which has a configurable parameter in the AA for allowing (or disallowing) external transfers. The GUI window where you can configure this parameter is shown in Figure 53.

*Figure 53 Allowing or Blocking AA Transfers to External Numbers*




If you are writing your own customized AA scripts, you can use the Extension To User step provided in the release 2.1 Cisco Unity Express Script Editor to block transfers to external numbers.

# Calls to Cisco Unity Express Get Fast-Busy

Sometimes when a call is placed to Cisco Unity Express (to either the AA or voice mail applications), you might hear fast-busy tone unexpectedly. That might be because the call setup from the originating system (such as a Cisco CME or PSTN voice gateway at another site) might not be using the right codec.

Cisco Unity Express supports only the G.711 u-law codec with 20-ms packetization. It is important that all the dial peers pointing to Cisco Unity Express on Cisco CME, or from any other PSTN voice gateway, are configured with this codec type. If calls with other types of codecs must be supported, a transcoder must be used to change the call to G.711 before it enters the Cisco Unity Express application. Transcoding with Cisco CME is supported as of release 3.2.

You can check that the codec in the Cisco CME configuration is correct as shown in the following example:

```
dial-peer voice 6800 voip
 destination-pattern 5...
 session protocol sipv2
 session target ipv4:a.3.6.127
 dtmf-relay sip-notify
 codec g711ulaw
 no vad
```

If Cisco Unity Express is getting calls from other PSTN gateways in the network apart from the Cisco CME host router, another way to verify the codec type used for the calls to Cisco Unity Express is to enable SIP debugging on Cisco Unity Express. In the **trace ccn stacksip dbug** output shown in the following example, you see that the codec used is pcmu (which is G.711 μ-law).

```
Cue# trace ccn stacksip dbug
Cue# show trace buffer tail
Press <CTRL-C> to exit...
2407 07/26 19:34:50.582 ACCN SIPL 0 -------
INVITE sip:5800@a.4.14.134:5060 SIP/2.0
Via: SIP/2.0/UDP  a.4.14.34:5060;branch=z9hG4bK165B
From: <sip:5010@a.4.14.34>;tag=372DC22C-19CA
To: <sip:5800@a.4.14.134>
Date: Mon, 26 Jul 2004 12:27:48 GMT
Call-ID: A3342E0-DE3611D8-80A2AFCE-EEF507D3@a.4.14.34
Supported: 100rel,timer
Min-SE: 1800
Cisco-Guid: 155931842-3728085464-2157948878-4009035731
User-Agent: Cisco-SIPGateway/IOS-12.x
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK, PRACK, COMET, REFER, SUBSCRIBE, NOTIFY,
  INFO, UPDATE, REGISTER
CSeq: 101 INVITE
Max-Forwards: 70
Remote-Party-ID: <sip:5010@a.4.14.34>;party=calling;screen=no;privacy=off
Timestamp: 1090844868
Contact: <sip:5010@a.4.14.34:5060>
Call-Info: <sip:a.4.14.34:5060>;method="NOTIFY;Event=telephone-event;
  Duration=2000"
Expires: 180
Allow-Events: telephone-event
Content-Type: application/sdp
Content-Length: 182

v=0
o=CiscoSystemsSIP-GW-UserAgent 2366 3655 IN IP4 a.4.14.34
s=SIP Call
c=IN IP4 a.4.14.34
t=0 0
m=audio 16506 RTP/AVP 0
c=IN IP4 a.4.14.34
a=rtpmap:0 PCMU/8000
a=ptime:20
Content-Length: 0
```

In countries where G.711 a-law is used, the conversion from G.711 a-law to G.711 u-law is done automatically by the Cisco IOS voice infrastructure software and is not a cause for concern for calls to Cisco Unity Express. No additional configuration is needed.

# Troubleshooting the Telephony User Interface and VXML Browser

The main interface used to interact with a voice mail system such as Cisco Unity Express is the Telephony User Interface (TUI). When a call arrives at a voice mail system, the system starts playing specific prompts depending on the call type (for example, an external caller, a subscriber, or a call forwarded into voice mail). To interact with the system, the user presses DTMF digits on his phone's keypad, and the voice mail system responds by playing more prompts until the end of the session. This is implemented using a Voice Extensible Markup Language (VXML) browser, also called a voice

browser. Think of the voice browser as a web browser that fetches VXML scripts instead of HTML pages over HTTP. VXML is mainly used for voice-based interactive systems such as interactive voice response (IVR) and voice mail. In Cisco Unity Express, the voice browser fetches VXML scripts over HTTP from an internal web server and interprets them to implement the TUI. The Tomcat web server uses Java Server Pages (JSP) to query the voice mail back-end databases to build VXML scripts, depending on the call information and subscriber's mailbox properties.

Sometimes it might be necessary to troubleshoot this aspect of the caller voice mail system interaction, because the caller might hear unexpected prompts or no prompts at all after pressing a digit. When this situation occurs, it is quite possible that the digit pressed on the phone (and translated to a SIP NOTIFY message) did not reach the CRS application or the voice browser software components of the Cisco Unity Express voice mail application.

To understand what is happening if this situation arises, it is useful to troubleshoot the voice browser's operation. It is very difficult to be specific about common problems encountered in this area, so instead this section uses an example of an active TUI session to Cisco Unity Express and describes the salient aspects of the trace output. Showing how the TUI session works helps you understand how to troubleshoot any problem that might arise in this area.

You can use the same tracing techniques to troubleshoot any issues with response time for DTMF from the voice mail system. For example, if you feel that the voice mail system's response to DTMF digits pressed on a phone is too slow, you can view the response time in these traces. Look at the time stamp at which the digit arrives at the voice mail application and the time stamp at which the next prompt is played. A large delay between the two time stamps might point to a performance issue on the Cisco Unity Express system.

The following example shows a voice browser trace for a call forwarded to Cisco Unity Express voice mail. You can use task ID (Task:18000000082, in this case) to differentiate between multiple simultaneous calls. This task ID is generated from the CRS Workflow infrastructure. As described in the preceding section, the calling, called, and original called numbers are important for the proper functioning of voice mail. Because voice mail is implemented using a VXML voice browser software component, these Call Information fields must be passed to the voice browser. As you can see in the voice browser trace in the following example, the voice browser receives all these Call Information fields.

```
Cue# trace ccn vbroswercore dbug
Cue# show trace buffer tail
2702 10/10 12:31:10.338 ACCN VBRW 0 Task:18000000082 Here is the
  DefaultCompilationConfig set in the JVM en_US
2702 10/10 12:31:10.338 ACCN VBRW 0 Task:18000000082 DefaultCompilationConfig will
  be set to en_US
2702 10/10 12:31:10.338 ACCN VBRW 0 Task:18000000082 DefaultCompilationConfig is
  now set to JVM en_US
2702 10/10 12:31:10.341 ACCN VBRW 0 Task:18000000082 enterLevel: SESSION_LEVEL
2702 10/10 12:31:10.341 ACCN VBRW 0 callContact.getSessionID() = 5000000007
2702 10/10 12:31:10.341 ACCN VBRW 0 Task:18000000082 VoiceBrowser sessionid:
  5000000007, taskid: 18000000082
2702 10/10 12:31:10.341 ACCN VBRW 0 callContact.getCallingNumber() = 6003
2702 10/10 12:31:10.341 ACCN VBRW 0 callContact.getDNIS() = null
2702 10/10 12:31:10.341 ACCN VBRW 0 callContact.getDNIS() = null
2702 10/10 12:31:10.341 ACCN VBRW 0 callContact.getCalledNumber() = 5800
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getANIIIDigits() = null
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getUserToUserInfo() = null
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getRDNIS() = null
```

The first trace line shown in the following example is important, especially when calls are forwarded into voice mail. The LastRedirectingNumber field comes into play when a call to an extension on Cisco CME is forwarded on busy or no-answer to voice mail. This identifies the original extension dialed and, hence, is the extension associated with the mailbox for which this message must be left.

If a value for the *callContact.getLastRedirectedNumber()* field is present in the trace, you can safely assume that this was a forwarded call and not a direct call to voice mail. This is shown in the following example:

```
Cue# show trace buffer tail
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact getLastRedirectedNumber() = 5010
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getRDNIS() = null
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getLastRedirectedNumber() = 5010
2702 10/10 12:31:10.342 ACCN VBRW 0 callContact.getSessionID() = 5000000007
2702 10/10 12:31:10.348 ACCN VBRW 0 Task:18000000082 VBContext::pushLang language
  = en_US
2702 10/10 12:31:10.348 ACCN VBRW 0 Task:18000000082 VBContext::pushLang Adding
  new language : en_US
2702 10/10 12:31:10.348 ACCN VBRW 0 Task:18000000082
```

After processing the call information, the voice browser starts the first VXML script, login.vxml, which is the top-level login script for the voice mail system. Depending on the type of call, either the external caller login script (if it is a forwarded call, the script to leave a message is executed) or the subscriber login script (if it is a direct call, the user is asked to identify himself) is executed next, as shown in the following example.

```
Cue# show trace buffer tail
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082
  VoiceBrowser.invokeApplication(level:0): [URI=http://localhost/voicemail/
  vxmlscripts/login.vxml fragment=null]
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082 enterScope: application
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082 enterLevel: APPLICATION_LEVEL
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082 Got document:
  http://localhost/voicemail/vxmlscripts/login.vxml from cache.
............
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082 enterLevel: APPLICATION_LEVEL
2702 10/10 12:31:10.349 ACCN VBRW 0 Task:18000000082
  VoiceBrowser.invokeRootDocument: [URI=http://localhost/voicemail/vxmlscripts/
systemLangID.jsp fragment=null]2702 10/10 12:31:10.357 ACCN VBRW 0 Task:18000000082
Fetch: HTTP 200 OK
2702 10/10 12:31:10.357 ACCN VBRW 0 Task:18000000082 Fetch: HTTP Response is not
  from Cache
2702 10/10 12:31:10.358 ACCN VBRW 0 Task:18000000082 Browser:
  com.cisco.voicebrowser.browser.CookieSet@3df95571: 1 cookies:
  JSESSIONID=mqh3bxs2a1;Path=/voicemail;Domain=localhost
2702 10/10 12:31:10.358 ACCN VBRW 0 Task:18000000082 Successfully fetched bytes:
  239, duration(s): 0.0090, URI: http://localhost/voicemail/vxmlscripts/
  systemLangID.jsp
```

This call is a forwarded call, so the caller login script to leave a message in a mailbox is executed. This interchange is shown in the following example.

```
Cue# show trace buffer tail
2702 10/10 12:31:10.389 ACCN VBRW 0 Task:18000000082 VoiceDomTraverser:
  handleElementBlock(): is called
2702 10/10 12:31:10.390 ACCN VBRW 0 Task:18000000082 <goto> destination:
  [URI=http://localhost/voicemail/vxmlscripts/caller_login.vxml fragment=null]
2702 10/10 12:31:10.391 ACCN VBRW 0 Task:18000000082 Got document:
  http://localhost/voicemail/vxmlscripts/caller_login.vxml from cache.
2702 10/10 12:31:10.391 ACCN VBRW 0 Task:18000000082 exitScope: anonymous
2702 10/10 12:31:10.391 ACCN VBRW 0 Task:18000000082 exitLevel: FIELD_LEVEL
2702 10/10 12:31:10.391 ACCN VBRW 0 Task:18000000082 exitLevel: DIALOG_LEVEL
2702 10/10 12:31:10.391 ACCN VBRW 0 Task:18000000082 Form context:
  check_pin_status=not defined
  _block1001="_defined"
  validate_id=not defined
  _block1000="_defined"
```

Because this is a forwarded call originally directed to extension 5010, the voice mail application must ensure that extension 5010 has a mailbox in this system. It verifies this by executing extValidation.jsp, as shown in the following example:

```
Cue# show trace buffer tail
2702 10/10 12:31:10.401 ACCN VBRW 0 Task:18000000082 Invoke subdialog[mbox_check]
  [URI=http://localhost/voicemail/vxmlscripts/extValidation.jsp fragment=null]
  Params: null
2702 10/10 12:31:10.401 ACCN VBRW 0 Task:18000000082 VBContext.pushContext level:1
2702 10/10 12:31:10.401 ACCN VBRW 0 Task:18000000082
  VoiceBrowser.invokeApplication(level:1):
  [URI=http://localhost/voicemail/vxmlscripts/extValidation.jsp fragment=null]
2702 10/10 12:31:10.402 ACCN VBRW 0 Task:18000000082 enterScope: application
2702 10/10 12:31:10.402 ACCN VBRW 0 Task:18000000082 enterLevel: APPLICATION_LEVEL
2702 10/10 12:31:10.420 ACCN VBRW 0 Task:18000000082 Fetch: HTTP 200 OK
2702 10/10 12:31:10.421 ACCN VBRW 0 Task:18000000082 Successfully fetched bytes:
  815, duration(s): 0.019, URI: http://localhost/voicemail/vxmlscripts/
  extValidation.jsp
2702 10/10 12:31:10.423 ACCN VBRW 0 Task:18000000082 exitLevel: APPLICATION_LEVEL
2702 10/10 12:31:10.424 ACCN VBRW 0 Task:18000000082 exitScope: application
2702 10/10 12:31:10.424 ACCN VBRW 0 Task:18000000082 enterScope: application
2702 10/10 12:31:10.424 ACCN VBRW 0 Task:18000000082 enterLevel: APPLICATION_LEVEL
2702 10/10 12:31:10.424 ACCN VBRW 0 Task:18000000082 enterScope: document
2702 10/10 12:31:10.425 ACCN VBRW 0 Task:18000000082 enterLevel: DOCUMENT_LEVEL
2702 10/10 12:31:10.425 ACCN VBRW 0 Task:18000000082 traverseDocument:
  http://localhost/voicemail/vxmlscripts/extValidation.jsp
  base=http://localhost/voicemail/vxmlscripts/extValidation.jsp
2702 10/10 12:31:10.425 ACCN VBRW 0 Task:18000000082
  VoiceDomParser:handleMetaElement(): - NodeName=meta:NodeValue=null:LocalName=
```

As you can see from this trace, the voice mail system continues to execute the appropriate JSP, which, in turn, access data from the LDAP and SQL databases to build dynamic VXML scripts and deliver these to the voice browser to execute.

The previous set of trace examples demonstrate the voice browser's operation. Sometimes it is necessary to troubleshoot TUI sessions for some calls. An example is when a caller hears unexpected prompts. You might need to see the DTMF digits the user is pressing and the prompts the voice mail system plays in response.

The following trace examples demonstrate TUI troubleshooting. In the voicemail vxml trace output, you can see a hexadecimal number (0x0000000430e2344f in the trace output). This is the unique call identification (or call ID) from the voice mail system's point of view. It is used to differentiate between multiple simultaneous calls.

This section describes TUI operation with a sample TUI session where a subscriber logs in and addresses a message to another subscriber. As soon as you understand this example, it is very easy to troubleshoot any other TUI-related situations that might crop up. Use the following commands to troubleshoot these types of issues:

- **trace ccn vbrowseroutput dbug**
- **trace voicemail vxml all**
- **show trace buffer tail**

One tip for using these commands is to clear the trace buffer before troubleshooting (**clear trace**) and turn off all other traces (**no trace all**). Then turn on the **trace ccn vbrowseroutput dbug** and **trace voicemail vxml all** commands and make the test call. Use the **show trace buffer long** command to see the trace buffer contents, because this command shows detailed messages about prompts, including the text that is the content of the voice mail prompt. The long format of traces helps you understand the TUI

interaction much better. One thing to note while looking at these traces is that **voicemail vxml** trace output is coming from the voicemail back end running as part of the Tomcat web server, and the trace lines containing VBRO are from the voice browser (client).

When the voice mail system answers a direct (not forwarded) incoming call, the voice browser takes the following actions:

- It starts accessing VXML scripts.
- It checks the default system language (which is only English until Cisco Unity Express 2.0).
- It validates the extension.
- It asks the user to enter a PIN.

The trace in the following example shows the output from the voice browser and voice mail VXML scripts. Note that traces come from modules ACCN and voicemail and entities VBRO and VXML, respectively.

```
Cue# show trace buffer tail
2699 10/10 12:23:30.115 ACCN VBRO 0 Task:18000000079 Invoke:
  http://localhost/voicemail/vxmlscripts/login.vxml
2699 10/10 12:23:30.118 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/systemLangID.jsp
2699 10/10 12:23:30.141 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/extValidation.jsp
2699 10/10 12:23:30.165 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/checkPinStatus.jsp
```

At this point, the voice mail application requests that the prompt AvSubSignInENU002.wav (which contains the phrase “Please enter your password”) be played. This is shown in the **voicemail vxml** trace in the following example.

```
Cue# show trace buffer tail
2699 10/10 12:23:30.234 voicemail vxml "Please enter your password.<1.5 sec
  silence>" 0x0000000430e2344f AvSubSignInENU002.wav
2699 10/10 12:23:30.235 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvSubSignIn/AvSubSignInENU002.wav
```

The DTMF digits pressed by the user are received by the voice browser, as shown in the following example. These digits are delivered by the SIP stack (as described in the “Troubleshooting DTMF—No Response for Digit Presses from Cisco Unity Express” section on page 203) and CRS infrastructure components to the voice browser. This is another place to look for correct processing of digits in a case where voice mail is not responding as expected to digits pressed by a caller. This is to ensure that the digits are indeed reaching the voice browser application component.

```
Cue# show trace buffer tail
2699 10/10 12:23:30.238 ACCN VBRO 0 Task:18000000079 Listen:
2699 10/10 12:23:34.332 ACCN VBRO 0 Task:18000000079 Heard:    'dtmf-2 dtmf-2
  dtmf-2 dtmf-#'
```

At this point, the voice mail system must execute a JSP to validate the caller’s password. The interaction between the user and voice mail system continues in this manner until the end of the TUI session. If you are facing any issues with the voice mail TUI (including TUI performance), the traces described in this section are the best way to understand which component is not working as expected and why.

The following example shows the continuation of the direct call. This example shows that the voice mail system has successfully logged in the subscriber.

```
Cue# show trace buffer tail
2699 10/10 12:23:34.337 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/validate_password.jsp
2699 10/10 12:23:34.416 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/setSubSession.jsp
```

```
2699 10/10 12:23:34.430 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/playSpokenName.jsp
2699 10/10 12:23:34.453 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/root.jsp
2699 10/10 12:23:34.617 voicemail vxml "hello." 0x0000000430e2344f
  AvSubMsgCountENU128.wav
2699 10/10 12:23:34.618 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvSubMsgCount/AvSubMsgCountENU128.wav
2699 10/10 12:23:35.023 voicemail vxml "You have no new messages."
  0x0000000430e2344f AvSubMsgCountENU001.wav
2699 10/10 12:23:35.023 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvSubMsgCount/AvSubMsgCountENU001.wav
```

At this point, the user presses 2 to choose to send a message, as shown in the following example.

```
Cue# show trace buffer tail
2699 10/10 12:23:35.029 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvSubMenu/AvSubMenuENU006.wav
2699 10/10 12:23:35.030 ACCN VBRO 0 Task:18000000079 Listen:
2699 10/10 12:23:42.340 ACCN VBRO 0 Task:18000000079 Heard:    'dtmf-2'
2699 10/10 12:23:42.353 voicemail vxml "Spell the name of the person or group."
  0x0000000430e2344f AvAesopCustomENU002.wav
2699 10/10 12:23:42.354 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAesopCustom/AvAesopCustomENU002.wav
2699 10/10 12:23:42.354 voicemail vxml "Spell the last and first name"
  0x0000000430e2344f AvAddrSearchENU005.wav
2699 10/10 12:23:42.355 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAddrSearch/AvAddrSearchENU005.wav
2699 10/10 12:23:42.355 voicemail vxml "For Q, press 7. For Z, press 9."
  0x0000000430e2344f AvAddrSearchENU030.wav
2699 10/10 12:23:42.356 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAddrSearch/AvAddrSearchENU030.wav
2699 10/10 12:23:42.357 voicemail vxml "To switch between spelling and number
  entry, press ##." 0x0000000430e2344f AvAddrSearchENU075.wav
2699 10/10 12:23:42.357 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAddrSearch/AvAddrSearchENU075.wav
```

The user presses ## to select addressing by extension number, as shown in the following example.

```
Cue# show trace buffer tail
2699 10/10 12:23:42.358 ACCN VBRO 0 Task:18000000079 Listen:
2699 10/10 12:23:45.990 ACCN VBRO 0 Task:18000000079 Heard:    'dtmf-#'
2699 10/10 12:23:45.994 ACCN VBRO 0 Task:18000000079 Listen:
2699 10/10 12:23:46.251 ACCN VBRO 0 Task:18000000079 Heard:    'dtmf-#'
2699 10/10 12:23:46.292 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/setAddressingMode.jsp
2699 10/10 12:23:46.305 voicemail vxml "Enter the extension." 0x0000000430e2344f
  AvAddrSearchENU037.wav
2699 10/10 12:23:46.305 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAddrSearch/AvAddrSearchENU037.wav
2699 10/10 12:23:46.306 voicemail vxml "To switch between spelling and number
  entry, press ##." 0x0000000430e2344f AvAddrSearchENU075.wav
2699 10/10 12:23:46.306 ACCN VBRO 0 Task:18000000079 Play:
  ../prompts/ENU/AvAddrSearch/AvAddrSearchENU075.wav
2699 10/10 12:23:46.307 ACCN VBRO 0 Task:18000000079 Listen:
```

The user enters the extension of the recipient mailbox, as shown in the following example.

```
Cue# show trace buffer tail
2699 10/10 12:23:52.081 ACCN VBRO 0 Task:18000000079 Heard:    'dtmf-6 dtmf-0
  dtmf-0 dtmf-1 dtmf-#'
2699 10/10 12:23:52.111 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/extValidation.jsp
2699 10/10 12:23:52.136 ACCN VBRO 0 Task:18000000079 Fetch:
  http://localhost/voicemail/vxmlscripts/getAddrParts.jsp
```

The user's interaction with the voice mail system continues like this until the end of the call.

# Troubleshooting the Database, LDAP, and Mailbox Activities

This section describes troubleshooting the Cisco Unity Express voice mail system's interaction with its back-end databases. Here you learn how to interpret SQL database traces, voice mail's interaction with the LDAP database for user information, and message and mailbox activities. These topics are described by looking at an example of a subscriber addressing a voice message to another subscriber and sending it. The **trace** commands needed for this section include the following:

- **trace voicemail database all**
- **trace voicemail ldap all**
- **trace voicemail message all**
- **trace voicemail mailbox all**
- **show trace buffer tail**

In the trace segment shown in the following example, a call arrives at the voice mail system with a calling number of 6001. As you can see from the traces, voice mail accesses the LDAP database to resolve the calling number to a user configured on the system to check whether the caller is a valid subscriber on the system or an outside (PSTN) caller calling into the voice mail system. Depending on the outcome of this query, the voice mail system plays different prompts.

In this example, the calling number 6001 is resolved to the user useru3. Following that action, the system retrieves the user's preferred language (English only in the system used to generate these traces) to select the appropriate prompts to play.

```
Cue# show trace buffer tail
6082 09/30 18:39:48.614 voicemail ldap "getUserByPhoneNo" 6001
6082 09/30 18:39:48.617 voicemail ldap "getUserByPhoneNo: userDn." /sw/local/
  users/useru3
6082 09/30 18:39:48.617 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/Language/preferredLanguage
6082 09/30 18:39:48.618 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/TelephoneNumbers/primaryExtension
6082 09/30 18:39:48.618 voicemail database 0 Got connection: 2, inUse: 3, active:
  2
```

After the calling user has been identified, the voice mail system verifies that the user has a mailbox, because it is not necessary for every user defined in the system to have an associated voice mailbox.

The SQL statement shown in the following example is executed and returns the mailbox ID of the given user (for example, the mailbox ID returned here is PERSONAL_00000000000000000000022).

```
Cue# show trace buffer tail
6082 09/30 18:39:48.619 voicemail database "SQL: " select mailboxid from
  vm_mbxusers where owner=true and userdn='/sw/local/users/useru3';
6082 09/30 18:39:48.621 voicemail database "Database query results"
  PERSONAL_00000000000000000000022
6082 09/30 18:39:48.621 voicemail database 0 Freed connection: 2, inUse: 3,
  active: 2
```

Next, the voice mail system retrieves the details of the mailbox for useru3 by executing the SQL statement shown in the trace output in the following example. The database returns a row of data. To interpret this data, you need the database schema. The schema-related trace output is shown in the following example.

```
create table vm_mailbox
(
    MailboxId            varchar(64)          not null,
    MailboxType          integer              not null default 0,
    Description          varchar(64)          ,
    MailboxSize          integer              ,
    MessageSize          integer              ,
    Tutorial             boolean              not null default true,
    TotalMessageTime     integer              ,
    MessageExpiryTime    integer              ,
    Enabled              boolean              not null default true,
    GreetingType         integer              not null default 10,
    OrphanedTime         bigint               not null default 0,
    primary key         (MailboxId)
);
```

```
Cue# show trace buffer tail
6082 09/30 18:39:53.085 voicemail database "SQL: " select * from vm_mailbox where
  mailboxid='PERSONAL_00000000000000000000022';
6082 09/30 18:39:53.090 voicemail database "Database query results"
  PERSONAL_00000000000000000000022,0,useru3 mailbox, 3000, 60, f,0,30, t,10,0,
  1064946956019
```

From the trace output and schema definition, the following are the characteristics of useru3's mailbox:

- *Mailbox ID*—PERSONAL_00000000000000000000022
- *MailboxType*—0, personal mailbox
- *Description*—useru3 mailbox
- *Mailboxsize*—3000 seconds
- *MessageSize*—60 seconds
- *Tutorial*—0, tutorial flag is OFF
- *TotalMessageTime*—0, meaning that the mailbox is empty
- *MessageExpiryTime*—30, messages expire after 30 days
- *Enabled*—t, True, the mailbox is enabled
- *GreetingType*—10, the standard greeting is active
- *OrphanedTime*—0, not an orphaned mailbox

The data returned from the back-end SQL database affects the prompts played to the caller via VXML. The JSP create dynamic VXML scripts for the TUI depending on the values returned from the back-end databases. For example, if the tutorial flag is set for the mailbox, the prompts played to the user start with the tutorial prompt "Welcome to Cisco Unity Express messaging system. To get the most from the system...."

After checking that the mailbox is enabled, the voice mail system lets the user log in to the mailbox. At the same time, the voice mail system retrieves the user's spoken name from the LDAP database to play it after the login. The traces in the following example show this exchange.

```
Cue# show trace buffer tail
6082 09/30 18:39:53.104 voicemail mailbox "User login" /sw/local/users/useru3
6082 09/30 18:39:53.104 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
6082 09/30 18:39:53.106 voicemail database "SQL: " update vm_mailbox set
  lastaccessed=1064947193104 where mailboxid='PERSONAL_00000000000000000000022';
6082 09/30 18:39:53.111 voicemail database "Commiting transaction"
6082 09/30 18:39:53.114 voicemail database 0 Freed connection: 3, inUse: 3,
  active: 2
6082 09/30 18:39:53.160 voicemail ldap "getSpokenNameByName: userDn."
  /sw/local/users/useru3
6082 09/30 18:39:53.160 voicemail ldap "normalizeDN" /sw/local/users/useru3
6082 09/30 18:39:53.160 voicemail ldap "getSpokenName: dn."
  uid=useru3,ou=users,ou=branch123,o=cisco.com
```

After having logged into the mailbox, user useru3 chooses to send a voice message to another subscriber at extension 6003. At this time, the voice mail system resolves extension 6003 to a subscriber called useru4. User useru4's spoken name is retrieved, if recorded, from LDAP and is played as a confirmation to useru3. Finally, the voice mail system finds useru4's mailbox ID (PERSONAL_00000000000000000000023), as shown in the following example.

```
Cue# show trace buffer tail
6081 09/30 18:40:07.825 voicemail ldap "getUserByPhoneNo" 6003
6081 09/30 18:40:07.828 voicemail ldap "getUserByPhoneNo: userDn." /sw/local/
  users/useru4
6081 09/30 18:40:07.828 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/Language/preferredLanguage
6081 09/30 18:40:07.829 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/TelephoneNumbers/primaryExtension
6081 09/30 18:40:07.829 voicemail database 0 Got connection: 3, inUse: 3, active:
  2
6081 09/30 18:40:07.830 voicemail database "SQL: " select mailboxid from
  vm_mbxusers where owner=true and userdn='/sw/local/users/useru4';
6081 09/30 18:40:07.832 voicemail database "Database query results"
  PERSONAL_00000000000000000000023
```

User useru4's spoken name, if recorded, is retrieved from LDAP and is played so that the sender can recognize the extension by spoken-name confirmation, as shown in the following example.

```
Cue# show trace buffer tail
6082 09/30 18:40:07.846 voicemail ldap "getSpokenNameByName: userDn." /sw/local/
  users/useru4
6082 09/30 18:40:07.846 voicemail ldap "normalizeDN" /sw/local/users/useru4
6082 09/30 18:40:07.847 voicemail ldap "getSpokenName: dn." uid=useru4,ou=users,
  ou=branch123,o=cisco.com
```

As soon as the recipient of the voice message is identified, the caller records a message. The voice mail system creates a message, assigns it a message ID, and sets the message's size. As shown in the trace output in the following example, a message with ID 1064947237095_0, 43258 bytes long, is created. This message is 5.2 seconds long (5281 ms).

```
Cue# show trace buffer tail
6082 09/30 18:40:37.096 voicemail message "Creating Message" 1064947237095_0
6082 09/30 18:40:37.096 voicemail message "Message Length" 5281, Message Size:
  43258
```

The next few lines of trace output (as given in the following example) show that a message is sent from user useru3 to extension 6003 and that a message with ID 1064947237095_0 is received for mailbox ID PERSONAL_00000000000000000000023. The total usage time for the destination mailbox is updated in the database. After sending the message, the user logs out of the mailbox.

```
Cue# show trace buffer tail
6082 09/30 18:40:37.096 voicemail mailbox "Sending message(s) from"
  0x000000037e11d61a /sw/local/users/useru3
6082 09/30 18:40:37.096 voicemail mailbox "Sending message to" 0x000000037e11d61a
  6003
.......
6082 09/30 18:40:37.096 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
6082 09/30 18:40:37.097 voicemail mailbox "Message received"
  0x0000000000000000 PERSONAL_00000000000000000000023,1064947237095_0
............
6082 09/30 18:40:37.099 voicemail database "SQL: " update vm_message
  set messageid='1064947237095_0',messagetype=1,sender='/sw/local/users/useru3',
  urgent=false,private=false,attachedmsgid=null where messageId='OID_16693';
6082 09/30 18:40:37.104 voicemail database "SQL: " insert into vm_usermsg
  values('PERSONAL_00000000000000000000023','1064947237095_0',1,1064947237095);
6082 09/30 18:40:37.108 voicemail database "SQL: " select totalmessagetime from vm
 _mailbox where mailboxid='PERSONAL_00000000000000000000023' for update;
6082 09/30 18:40:37.111 voicemail database "Database query results" 2409
6082 09/30 18:40:37.111 voicemail database "SQL: " update vm_mailbox set
  totalmessagetime=7690 where mailboxid='PERSONAL_00000000000000000000023';
6082 09/30 18:40:37.115 voicemail database "Commiting transaction"
6082 09/30 18:40:37.118 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/TelephoneNumbers/primaryExtension
6082 09/30 18:40:37.131 voicemail database 0 Freed connection: 2, inUse: 3,
  active: 2
6081 09/30 18:40:43.994 voicemail mailbox "User logout" 0x000000037e11d61a
  /sw/local/users/useru3
```

The following example explores what happens when User U4 logs in to her mailbox and listens to the message just sent by User U3. First, the calling number (6003) is resolved to a subscriber on the system, and the subscriber's details are retrieved. The calling subscriber's mailbox ID is checked, and the details of that mailbox are retrieved from the vm_mailbox table. Then the voice mailbox entity logs the subscriber into the mailbox, updates the last access time for the mailbox, and retrieves the user's spoken name.

```
Cue# show trace buffer tail
6082 09/30 18:40:48.912 voicemail ldap "getUserByPhoneNo" 6003
6082 09/30 18:40:48.915 voicemail ldap "getUserByPhoneNo: userDn."
  /sw/local/users/useru4
6082 09/30 18:40:48.916 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/Language/preferredLanguage
6082 09/30 18:40:48.916 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/TelephoneNumbers/primaryExtension
6082 09/30 18:40:48.917 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
.......
```

```
6082 09/30 18:40:48.918 voicemail database "SQL: " select mailboxid from vm
 _mbxusers where owner=true and userdn='/sw/local/users/useru4';
6082 09/30 18:40:48.920 voicemail database "Database query results"
  PERSONAL_00000000000000000000023
6082 09/30 18:40:48.921 voicemail database 0 Freed connection: 3, inUse: 3,
  active: 2
6081 09/30 18:40:57.002 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
6081 09/30 18:40:57.003 voicemail database "SQL: " select * from vm_mailbox where
  mailboxid='PERSONAL_00000000000000000000023';
6081 09/30 18:40:57.009 voicemail database "Database query results"
  PERSONAL_00000000000000000000023,0,useru4
  mailbox,3000,60,f,7690,30,t,10,0,1064947025206
6081 09/30 18:40:57.009 voicemail database "SQL: " select userdn from vm_mbxusers
  where owner=true and mailboxid='PERSONAL_00000000000000000000023';
6081 09/30 18:40:57.012 voicemail database 0 Freed connection: 2, inUse: 3,
  active: 2
6081 09/30 18:40:57.013 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
6081 09/30 18:40:57.013 voicemail database "SQL: " 0x000000037e11d61c select
  mailboxid from vm_mbxusers where owner=true and userdn='/sw/local/users/
  useru4';
6081 09/30 18:40:57.016 voicemail database "Database query results"
  0x000000037e11d61c PERSONAL_00000000000000000000023
6081 09/30 18:40:57.017 voicemail database "SQL: " 0x000000037e11d61c select
  distinct vm_mbxusers.mailboxid, orphanedtime from vm_mbxusers, vm_mailbox where
  vm_mailbox.mailboxid=vm_mbxusers.mailboxid and (userdn='/sw/local/users/
  useru4') and orphanedtime=0 and owner=false;
6081 09/30 18:40:57.022 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 3, active: 2
6081 09/30 18:40:57.022 voicemail mailbox "User login" /sw/local/users/useru4
6081 09/30 18:40:57.023 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
6081 09/30 18:40:57.024 voicemail database "SQL: " update vm_mailbox set
  lastaccessed=1064947257023 where mailboxid='PERSONAL_00000000000000000000023';
6081 09/30 18:40:57.028 voicemail database "Commiting transaction"
6081 09/30 18:40:57.030 voicemail database 0 Freed connection: 2, inUse: 3,
  active: 2
6081 09/30 18:40:57.165 voicemail ldap "getSpokenNameByName: userDn."
  /sw/local/users/useru4
6081 09/30 18:40:57.165 voicemail ldap "normalizeDN" /sw/local/users/useru4
6081 09/30 18:40:57.165 voicemail ldap "getSpokenName: dn."
  uid=useru4,ou=users,ou=branch123,o=cisco.com
```

The voice mail system now gets details about the messages for this particular mailbox from the *vm_usermsg* and *vm_message* tables in the SQL database, as shown in the following example. You can see as part of this SQL transaction that all the message attributes are returned, including the following:

- The message's sender (/sw/local/users/useru3)
- The message's length (5281 seconds)
- The message's size (43258 bytes)
- When the message was left (30-Sep-03 18:40:25 UTC)

Also note the value 16693, which is the object ID of the message stored as an object in the SQL database. References to this appear in the later traces also. Right now the messages in the mailbox are ordered according to urgency and secondarily by the time when they were left. Urgent messages are played out first, and then the remaining messages are played in the order in which they were received.

```
Cue# show trace buffer tail
5047 09/30 18:40:57.207 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
```

```
5047 09/30 18:40:57.208 voicemail database "SQL: " 0x000000037e11d61c select *
  from vm_usermsg,vm_message where mailboxid='PERSONAL_00000000000000000023'
  and vm_usermsg.messageid=vm_message.messageid;
5047 09/30 18:40:57.225 voicemail database "Database query results"
  0x000000037e11d61c PERSONAL_00000000000000000023,1064947237095_0,1,
  1064947237095,1064947237095_0,1,1,/sw/local/users/useru3,f,f,5281,43258,
  1064947225155,null,16693,30-Sep-03 18:40:25 UTC
5047 09/30 18:40:57.225 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 3, active: 2
6081 09/30 18:41:01.912 voicemail database 0 Got connection: 2, inUse: 3, active:
  2
6081 09/30 18:41:01.913 voicemail database "SQL: " 0x000000037e11d61c select
  mailboxid, vm_usermsg.messageid, urgent, messagetime from vm_usermsg, vm_message
  where vm_usermsg.messageid=vm_message.messageid and state=1 and mailboxid=
  'PERSONAL_00000000000000000023' and messagetype<10 order by urgent desc,
  messagetime asc;
6081 09/30 18:41:01.918 voicemail database "Database query results"
  0x000000037e11d61c PERSONAL_00000000000000000023,1064947237095_0,f,
  1064947225155
```

Note that the message ID was received in the last SQL transaction. In the traces shown in the following example, the voice mail system creates a message to be played using that message ID.

```
Cue# show trace buffer tail
6081 09/30 18:41:01.919 voicemail database 0 0x000000037e11d61c Freed connection:
  2, inUse: 3, active: 2
6081 09/30 18:41:01.919 voicemail database 0 Got connection: 3, inUse: 3, active:
  2
6081 09/30 18:41:01.920 voicemail message 0 1064947237095_0
6081 09/30 18:41:01.924 voicemail message "Creating Message" 1064947237095_0
6081 09/30 18:41:01.925 voicemail database "Database query results"
  0x000000037e11d61c 1064947237095_0,1,/sw/local/users/useru3,f,f,5281,43258,
  1064947225155,null,16693,30-Sep-03 18:40:25 UTCh
6081 09/30 18:41:01.925 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 3, active: 2
```

Because User U3 is the sender of the message, the voice mail system now must retrieve details such as his spoken name and extension to play as part of the message's envelope information, as shown in the following example.

```
Cue# show trace buffer tail
6081 09/30 18:41:01.926 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/TelephoneNumbers/primaryExtension
6081 09/30 18:41:01.926 voicemail ldap "getUserByPhoneNo" 6001
6081 09/30 18:41:01.929 voicemail ldap "getUserByPhoneNo: userDn."
  /sw/local/users/useru3
6081 09/30 18:41:01.929 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/Language/preferredLanguage
6081 09/30 18:41:01.929 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/TelephoneNumbers/primaryExtension
6081 09/30 18:41:01.930 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
6081 09/30 18:41:01.931 voicemail database "SQL: " select mailboxid from vm
 _mbxusers where owner=true and userdn='/sw/local/users/useru3';
6081 09/30 18:41:01.934 voicemail database "Database query results"
  PERSONAL_00000000000000000022
6081 09/30 18:41:01.934 voicemail database 0 Freed connection: 2, inUse: 3,
  active: 2
6081 09/30 18:41:01.934 voicemail ldap "getUserByDn" /sw/local/users/useru3
6081 09/30 18:41:01.934 voicemail ldap "normalizeDN" /sw/local/users/useru3
6081 09/30 18:41:01.934 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/TelephoneNumbers/primaryExtension
6081 09/30 18:41:01.935 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru3/Language/preferredLanguage
```

```
6081 09/30 18:41:01.935 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
6081 09/30 18:41:01.936 voicemail database "SQL: " select mailboxid from vm
 _mbxusers where owner=true and userdn='/sw/local/users/useru3';
6081 09/30 18:41:01.938 voicemail database "Database query results"
  PERSONAL_00000000000000000000022
6081 09/30 18:41:01.938 voicemail database 0 Freed connection: 3, inUse: 3,
  active: 2
6081 09/30 18:41:01.938 voicemail ldap "getSpokenNameByName: userDn."
  /sw/local/users/useru3
6081 09/30 18:41:01.939 voicemail ldap "normalizeDN" /sw/local/users/useru3
6081 09/30 18:41:01.939 voicemail ldap "getSpokenName: dn."
  uid=useru3,ou=users,ou=branch123,o=cisco.com
5047 09/30 18:41:12.757 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
```

In the trace shown in the following example you can see the message entity of the voice mail system retrieving message object 16693 from the database. After User U4 listens to the message, she opts to delete it. In the traces shown in the following example, you see that the message and large object are deleted, and the mailbox properties (for example, mailbox time remaining) are updated in the database. Then the user logs out of the mailbox.

```
Cue# show trace buffer tail
5047 09/30 18:41:12.757 voicemail message 16693 get
5047 09/30 18:41:12.757 voicemail database "Large object open" 16693 0
  0x000000037e11d61c
5047 09/30 18:41:12.758 voicemail database "Large object close" 0x000000037e11d61c
5047 09/30 18:41:12.758 voicemail database 0 0x000000037e11d61c Freed connection:
  2, inUse: 3, active: 2
6081 09/30 18:41:14.970 voicemail mailbox "Message deleted" 0x000000037e11d61c
  PERSONAL_00000000000000000000023,1064947237095_0
6081 09/30 18:41:14.970 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
6081 09/30 18:41:14.971 voicemail database "SQL: " 0x000000037e11d61c delete from
  vm_usermsg where messageid='1064947237095_0' and
  mailboxid='PERSONAL_00000000000000000000023';
6081 09/30 18:41:14.974 voicemail database "SQL: " 0x000000037e11d61c select
  usecount,messagelength,attachedmsgid, messageoid from vm_message where
  messageId='1064947237095_0' for update;
6081 09/30 18:41:14.977 voicemail database "Database query results"
  0x000000037e11d61c 1,5281,null,16693
6081 09/30 18:41:14.978 voicemail message "Deleting Message" 1064947237095_0
6081 09/30 18:41:14.978 voicemail database "SQL: " 0x000000037e11d61c delete from
  vm_message where messageid='1064947237095_0';
6081 09/30 18:41:14.982 voicemail database 16693 0x000000037e11d61c Deleted large
  object id:
6081 09/30 18:41:14.982 voicemail database "Commiting transaction"
  0x000000037e11d61c
6081 09/30 18:41:14.985 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 3, active: 2
6081 09/30 18:41:14.985 voicemail database 0 Got connection: 2, inUse: 3,
  active: 2
6081 09/30 18:41:14.986 voicemail database "SQL: " 0x000000037e11d61c select
  totalmessagetime from vm_mailbox where mailboxid='PERSONAL
 _00000000000000000000023' for update;
6081 09/30 18:41:14.989 voicemail database "Database query results"
  0x000000037e11d61c 7690
6081 09/30 18:41:14.989 voicemail database "SQL: " 0x000000037e11d61c update vm
 _mailbox set totalmessagetime=2409 where mailboxid='PERSONAL
 _00000000000000000000023';
6081 09/30 18:41:14.993 voicemail database "Commiting transaction"
  0x000000037e11d61c
6081 09/30 18:41:14.995 voicemail database 0 Got connection: 3, inUse: 4,
```

```
  active: 2
6081 09/30 18:41:14.996 voicemail database "SQL: " 0x000000037e11d61c select
  count(*) from vm_usermsg where state=1 and mailboxid='PERSONAL
 _00000000000000000000023';
6081 09/30 18:41:14.999 voicemail database "Database query results"
  0x000000037e11d61c 0
6081 09/30 18:41:15.000 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 4, active: 2
6081 09/30 18:41:15.000 voicemail ldap 0 getAttributeValue:
  /sw/local/users/useru4/TelephoneNumbers/primaryExtension
6081 09/30 18:41:15.031 voicemail database 0 0x000000037e11d61c Freed connection:
  2, inUse: 3, active: 2
6082 09/30 18:41:15.108 voicemail ldap "getUserByPhoneNo" 80016003
6082 09/30 18:41:15.112 voicemail ldap "getUserByPhoneNo: No entry found."
6082 09/30 18:41:15.200 voicemail database 0 Got connection: 3, inUse: 3,
  active: 2
6082 09/30 18:41:15.201 voicemail database "SQL: " 0x000000037e11d61c select *
  from vm_usermsg,vm_message where mailboxid='PERSONAL_00000000000000000000023'
  and vm_usermsg.messageid=vm_message.messageid;
6082 09/30 18:41:15.206 voicemail database 0 0x000000037e11d61c Freed connection:
  3, inUse: 3, active: 2
6082 09/30 18:41:20.357 voicemail mailbox "User logout" 0x000000037e11d61c
  /sw/local/users/useru4
```

In a similar way, you can isolate any other voice mail back-end-related issues using the preceding trace commands and referring to the database schema. The Cisco Unity Express Database Schema changes from one release to another as more features are added, but the basic nature of the schema and troubleshooting using the schema remain the same. The key to effective troubleshooting in this area is understanding what data is stored in which database.

# Troubleshooting the Message Waiting Indicator

The message waiting indicator (MWI) is one of the most important and basic features of any voice mail system. Sometimes, because of many factors, the subscriber does not see the correct MWI changes on the phone. This could be because somehow Cisco CME and Cisco Unity Express got out of sync with each other concerning the MWI status for subscribers. (For example, perhaps Cisco CME was reloaded and it does not keep track of MWI state across reloads.) This section describes MWI configuration and troubleshooting MWI-related issues. The following sections MWI-related troubleshooting notes:



## MWI Operation

With Cisco CME deployments, Cisco Unity Express uses what is called a callout mechanism to change MWI status on the phone. In this mechanism, Cisco CME defines two special extensions: the MWI ON and MWI OFF DNs. These extension definitions have a very specific number format.

For example, the pattern can be xx22nnnn for the MWI ON DN and xx21nnnn for the MWI OFF DN, where xx22 and xx21 are prefixes, and the number of dots is equal to the length of the extensions in your Cisco CME dial plan. If Cisco Unity Express must turn on MWI for extension 6001, it places an outgoing call to number xx226001. Cisco CME terminates the call on the MWI ON DN (because the called number matches the dial peer for the MWI ON DN). Cisco CME processes the called number of the call

and extracts extension 6001 from it, which is matched by the four dots defined on the MWI ON DN. It then finds all the phones with appearances of extension 6001 on them (there can be several if 6001 is a shared line), and then sends an SCCP MWI ON message to each of the phones. A similar sequence of steps is followed to turn off an MWI lamp by Cisco Unity Express outcalling to the MWI OFF DN. For example, xx216001 is used to turn off MWI for extension 6001.

You can test the operation of your MWI DNs on Cisco CME by dialing the MWI DN prefix followed by the extension from any phone. For example, if you dial xx226001 from any phone, you will see the MWI light activated on phones with extension 6001. This verifies that your MWI DNs are correctly configured and operational.

When you choose the patterns for MWI DNs, keep in mind that such a pattern must not overlap with any other extension or dial peer patterns configured on Cisco CME. Those configurations can interfere with the MWI calls that come from Cisco Unity Express, effectively breaking MWI operation. Also look for any Cisco IOS translation rules that might translate the called numbers for the MWI DN call. This also breaks the MWI operation.

Now that you understand the mechanism between Cisco CME and Cisco Unity Express for turning MWI on or off, the next step is to investigate what happens inside the Cisco Unity Express software components to affect MWI when a voice message is left in or deleted from a mailbox.

When a new message is left in a mailbox, or a subscriber deletes the last new message in the mailbox, the voice mail system must change the MWI state on the subscriber's phone. However, the voice mail back-end component does not have access to any call control mechanism to place the outcall to Cisco CME. Instead, it depends on the CRS component, which contains a SIP stack for call control. However, there is no direct interface between the voice mail back end and the CRS component.

The CRS component can start applications on three kinds of triggers:

- E-mail
- HTTP
- Call

To affect MWI changes, the voice mail system (internally) sends an HTTP trigger to the CRS software. A system-level CRS application called ciscomwiapplication, with an associated script called setMWI.aef, exists and is configured to respond to this HTTP request.

When voice mail sends an HTTP request specifying the extension for MWI operation and the state to be set (ON or OFF), this CRS application starts up and places a call to the appropriate Cisco CME MWI ON or OFF DN using the parameters passed.

# Verifying MWI Configuration

For MWI to work correctly, it is important that the configurations on Cisco CME and Cisco Unity Express are correct and match. To verify the configuration, use the commands shown in the following example:

```
Ccme# show telephony-service ephone-dn
ephone-dn 20
number 7701
preference 0 secondary 9
huntstop
call-forward busy 5800
call-forward noan 5800 timeout 10
hold-alert 30 originator

ephone-dn 21
number 7010
```

```
preference 0 secondary 9
huntstop

ephone-dn 22
number 7011
preference 0 secondary 9
huntstop
call-forward busy 5800
call-forward noan 5800 timeout 10

ephone-dn 23
number 7012
preference 0 secondary 9
huntstop

ephone-dn 50
number xx22....
preference 0 secondary 9
huntstop
mwi on

ephone-dn 51
number xx21....
preference 0 secondary 9
huntstop
mwi off

ephone-dn 61
number 6001
preference 0 secondary 9
huntstop
call-forward all 5800
call-forward busy 5800
call-forward noan 5800 timeout 10
```

On Cisco Unity Express, ensure that the ciscomwiapplication CRS application is created and configured properly, as shown inthe following example:

```
Cue# show ccn application
Name:                                   ciscomwiapplication
Description:                            ciscomwiapplication
Script:                                 setmwi.aef
ID number:                              0
Enabled:                                yes
Maximum number of sessions:             1
strMWI_OFF_DN:                          2221
strMWI_ON_DN:                           2222
CallControlGroupID:                     0

Name:                                   voicemail
Description:                            voicemail
Script:                                 voicebrowser.aef
ID number:                              1
Enabled:                                yes
Maximum number of sessions:             1
logoutUri:                              http://localhost/voicemail/vxmlscripts/m
bxLogout.jsp
uri:                                    http://localhost/voicemail/vxmlscripts/l
ogin.vxml

Name:                                   autoattendant
Description:                            autoattendant
Script:                                 aa.aef
ID number:                              2
```

```
Enabled:                                yes
Maximum number of sessions:             1
MaxRetry:                               3
operExtn:                               7701
welcomePrompt:                          AAWelcome.wav

Name:                                   promptmgmt
Description:                            promptmgmt
Script:                                 promptmgmt.aef
ID number:                              3
Enabled:                                yes
Maximum number of sessions:             1
```

You can also verify the current MWI DN settings in the Cisco Unity Express sysDB settings, as shown in the following example. Note that in the output of both the Cisco Unity Express **show ccn application** and **show sysdb** commands, MWI DNs appear without the dots. This does not mean that they are without dots on Cisco CME.

```
Cue# show sysdb /sw/apps/ccn/wf/applications/craAesop/applications/ciscomwiapplication
app/applicationId                       0
app/applicationType                     Cisco Script Application
app/description                         ciscomwiapplication
app/enabled                             1
app/maxSessions                         8
app/script                              setmwi.aef
app/aeAccess                            0
app/aeEnd                               1
cfgVars/strMWI_OFF_DN                   2221
cfgVars/strMWI_ON_DN                    2222
cfgVars/CallControlGroupID              0
cfgVarsType/strMWI_OFF_DN               java.lang.String
cfgVarsType/strMWI_ON_DN                java.lang.String
cfgVarsType/CallControlGroupID          java.lang.Integer
```

There can be some inconsistency between the Cisco CME and Cisco Unity Express MWI DN configurations if the Cisco CME configuration is changed directly through the Cisco CME GUI or via Cisco IOS CLI. If you see that an inconsistency exists, synchronize both configurations. You can achieve this by navigating to the **Administration > Synchronize Information** window in the GUI, as shown in Figure 54.

If there are any inconsistencies between the Cisco CME and Cisco Unity Express configurations with respect to MWI DNs or other fields, these are listed on the **Administration > Synchronize Information** window. You must synchronize the information on this window for the MWI mechanism to work correctly. After you synchronize the information, the MWIs are not automatically refreshed at that time.

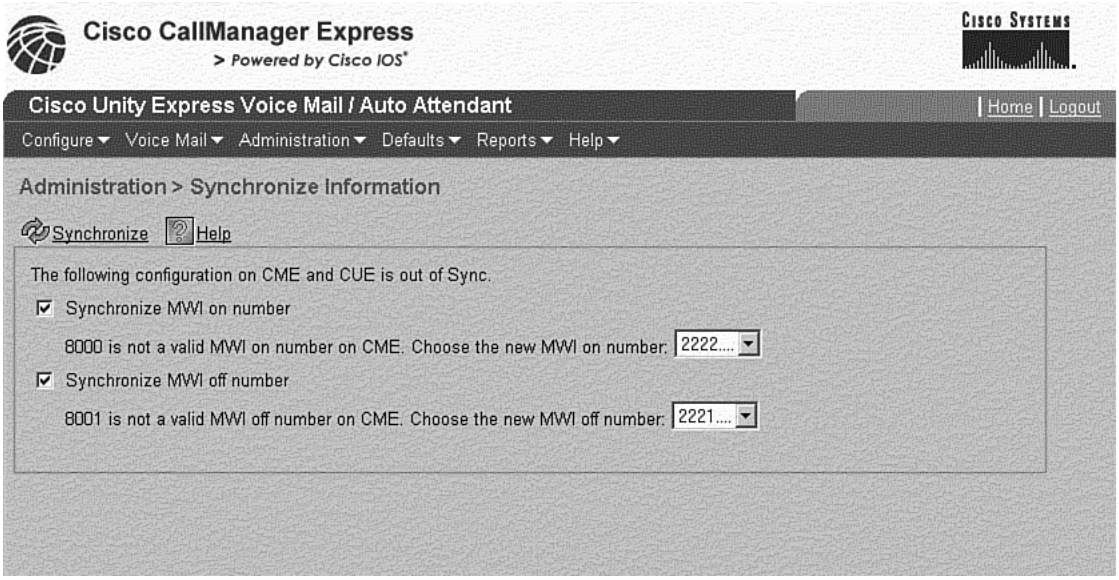
You can refresh the MWI states on subscribers' phones in two ways:

- You can refresh a particular subscriber's MWI using the **mwi refresh telephonenumber number** command in the Cisco Unity Express CLI.
- You can refresh MWI for all the subscribers in your Cisco CME system using the **mwi refresh all** command.

You can perform both operations via the GUI by navigating to the **Voice Mail > Message Waiting Indicators > Refresh** window.

**Note** Cisco Unity Express sends an MWI update every time there is a new message in a subscriber's mailbox, not just when there is a state change in a mailbox. Also, Cisco Unity Express does not have the capability to synchronize MWI states automatically at a configured time (or midnight routines), so manual synchronization is a must in case the states get out of synchronization.

*Figure 54 Synchronizing MWI Information*




One other configuration that can affect MWI operation is the SIP gateway address configuration in Cisco Unity Express. This should point to the Cisco CME with which Cisco Unity Express is integrated. You can verify this setting by using the **show ccn subsystem sip** Cisco Unity Express CLI command.

# Tracing MWI

This section investigates which traces to turn on to troubleshoot MWI-related problems, as well as how to interpret the traces to verify that all the subsystems involved in MWI operation are working correctly. The following trace commands are used to troubleshoot problems in this area:

- **trace ccn stacksip dbug**
- **trace voicemail mwi all**
- **trace ccn managerappl dbug**
- **show trace buffer tail**

In the traces shown in the following example, extension 5010 has received a new voice message, so the voice mail system sets the MWI state for this extension to true (the ON state). This action by the voice mail system results in an HTTP trigger being sent to the CRS software component, as described in the previous sections. In the third line in the following example, you can see the HTTP message going to port 8080, passing the extension number (5010 in this example) and the desired state of the MWI (1 for ON).

```
Cue# show trace buffer tail
2487 10/01 14:26:54.309 voicemail mwi "setMessageWaiting" 0x0000000000000000 5010,
  true
2487 10/01 14:26:54.309 voicemail mwi " job state" adding job
1792 10/01 14:26:54.309 voicemail mwi "job state" http://localhost: 8080/
  mwiapp?extn=5010&state=1

2317 10/01 14:26:54.316 ACCN APMG 0 TASK_CREATED:Application task created:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Application
  Trigger=ContactApplicationTrigger[time=1065018436314,contact=HttpContact[id=54,
```

```
  type=Cisco Http Contact,implId=null,active=true,state=CONTACT_RECEIVED,
  inbound=true,handled=false,
  locale=en_US,aborting=false,app=App[name=ciscomwiapplication,type=Cisco Script
  Application,id=0,desc=ciscomwiapplication,enabled=true,max=8,valid=true,
  optional=[cfgVars=[Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=
  setmwi.aef]],task=null,session=null,seqNum=-1,time=1065018436314,req=org.apache.
  tomcat.facade.HttpServletRequestFacade@7ce36091]],Task id=16,000,000,031,Task
  Class=com.cisco.wfframework.engine.core.WFEngineWorkflowDebugTask,New Task
  Class=com.cisco.app.impl.WFWorkflowAppDebugTaskWrapper
2317 10/01 14:26:54.317 ACCN APMG 0 APP_SESSION_ACTIVE:Active application session:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Application
  Trigger=ContactApplicationTrigger[time=1065018436314,contact=HttpContact[id=54,
  type=Cisco Http
```

As shown in the preceding trace, CRS starts its outcalling application in response to the HTTP trigger received from the voice mail component. the following example is the step-by-step trace of the CRS application executing this operation.

```
Cue# show trace buffer tail
2317 10/01 14:26:54.318 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=0,Step Class=com.cisco.wfframework.steps.core.
  StepStart,Step Description=Start
2317 10/01 14:26:54.319 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=1,Step Class=com.cisco.wf.steps.ivr.AcceptStep,Step
  Description=Accept (contact: --Triggering Contact--)
2317 10/01 14:26:54.319 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=111,Step Class=com.cisco.wf.steps.http.
  GetHttpHeaderStep,Step Description=Get Http Contact Info (contact: --Triggering
  Contact--)
2317 10/01 14:26:54.320 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=142,Step Class=com.cisco.wfframework.steps.core.
  StepIf,Step Description=If ( mwistatus != “0” ) Then
2317 10/01 14:26:54.320 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=143,Step Class=com.cisco.wfframework.steps.core.
  StepAssign,Step Description=Set strMWI_DN = strMWI_ON_DN
2317 10/01 14:26:54.321 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=144,Step Class=com.cisco.wfframework.steps.core.
  StepAssign,Step Description=Set mwiOn = true
2317 10/01 14:26:54.321 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=78,Step Class=com.cisco.wfframework.steps.core.StepIf,
  Step Description=If ( mwistatus != ”” ) Then
```

```
2317 10/01 14:26:54.322 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=83,Step Class=com.cisco.wfframework.steps.core.StepIf,
  Step Description=If ( strDeviceNum != "" ) Then
2317 10/01 14:26:54.322 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=84,Step Class=com.cisco.wfframework.steps.core.
  StepAssign,Step Description=Set strMWI_CompleteNum = strMWI_DN +  strDeviceNum
2317 10/01 14:26:54.322 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=189,Step Class=com.cisco.wf.steps.ivr.MWIStep,Step
  Description=Set Message Waiting Indicator (DN: strDeviceNum)
2317 10/01 14:26:54.323 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=193,Step Class=com.cisco.wfframework.steps.core.
  StepComment,Step Description=/* Don't return fail since ... */
2317 10/01 14:26:54.323 ACCN APMG 0 EXECUTING_STEP:Executing a step:
  Application=App[name=ciscomwiapplication,type=Cisco Script Application,id=0,
  desc=ciscomwiapplication,enabled=true,max=8,valid=true,optional=[cfgVars=
  [Lcom.cisco.wfapi.util.WFNameValuePair;@6786a090,script=setmwi.aef]],Task
  id=16,000,000,031,Step id=175,Step Class=com.cisco.wf.steps.ivr.CreateCallStep,
  Step Description=Place Call (to strMWI_CompleteNum )
2317 10/01 14:26:54.329 ACCN SIPL 0 --- send message --- to a.3.6.29:5060
```

As a result of the last step, which sets up a call to the value contained in strMWI_CompleteNum, you can see in the traces shown in the following example that a SIP INVITE message is sent from Cisco Unity Express to Cisco CME to change the state of the MWI lamp.

```
Cue# trace ccn stacksip dbug
Cue# show trace buffer tail
INVITE sip:xx225010@a.3.6.29;user=phone SIP/2.0
Via: SIP/2.0/UDP a.3.6.129:5060
From: "Cisco SIP Channel5" <sip:outbound-0@a.3.6.29>;tag=5bed1e99-272
To: <sip:xx225010@a.3.6.29;user=phone>
Call-ID: c91b8616-270@a.3.6.129:5060
CSeq: 51 INVITE
Contact: sip:outbound-0@a.3.6.129:5060
User-Agent: Jasmin UA / ver 1.1
Accept: application/sdp
Content-Type: application/sdp
Content-Length: 216

v=0
o=CiscoSystemsSIP-Workflow-App-UserAgent 2792 2792 IN IP4 a.3.6.129
s=SIP Call
c=IN IP4 a.3.6.129
t=0 0
m=audio 16906 RTP/AVP 0 111
a=rtpmap:0 pcmu/8000
a=rtpmap:111 telephone-event/8000
a=fmtp:111 0-11

2272 10/01 14:26:54.335 ACCN SIPL 0 receive 357 from a.3.6.29:5060
2273 10/01 14:26:54.336 ACCN SIPL 0 not found header for Date
2273 10/01 14:26:54.336 ACCN SIPL 0 not found header for Allow-Events
2273 10/01 14:26:54.336 ACCN SIPL 0 -------
```

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP a.3.6.129:5060
From: "Cisco SIP Channel5" <sip:outbound-0@a.3.6.29>;tag=5bed1e99-272
To: <sip:xx225010@a.3.6.29;user=phone>;tag=EFCE8-E51
Date: Fri, 01 Mar 2002 00:16:22 GMT
Call-ID: c91b8616-270@a.3.6.129:5060
Server: Cisco-SIPGateway/IOS-12.x
CSeq: 51 INVITE
Allow-Events: telephone-event
Content-Length: 0
```

In the traces shown in the following example, Cisco CME sends a 1 "80 Ringing" message to Cisco Unity Express, indicating that the call terminated on a valid MWI DN and that the MWI request is being processed.

```
Cue# trace ccn stacksip dbug
Cue# show trace buffer tail
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP a.3.6.129:5060
From: "Cisco SIP Channel5" <sip:outbound-0@a.3.6.29>;tag=5bed1e99-272
To: <sip:xx225010@a.3.6.29;user=phone>;tag=EFCE8-E51
Date: Fri, 01 Mar 2002 00:16:22 GMT
Call-ID: c91b8616-270@a.3.6.129:5060
Server: Cisco-SIPGateway/IOS-12.x
CSeq: 51 INVITE
Allow: UPDATE
Allow-Events: telephone-event
Contact: <sip:xx225010@a.3.6.29:5060>
Content-Length: 0

.........
CANCEL sip:xx225010@a.3.6.29;user=phone SIP/2.0
Via: SIP/2.0/UDP a.3.6.129:5060
From: "Cisco SIP Channel5" <sip:outbound-0@a.3.6.29>;tag=5bed1e99-272
To: <sip:xx225010@a.3.6.29;user=phone>
Call-ID: c91b8616-270@a.3.6.129:5060
CSeq: 51 CANCEL
Max-Forwards: 50
Content-Length: 0
```

At this point, Cisco Unity Express has finished notifying Cisco CME about a change in MWI status. It is now the responsibility of the Cisco CME call control software to complete the MWI notification to the IP phone to turn the lamp on or off. To troubleshoot this portion of the transaction, turn on the **debug ephone mwi** or **debug ephone detail** traces in Cisco IOS software. The traces in the following example show the SCCP messages sent by Cisco CME to the phone(s) for turning MWI on and off.

```
Ccme# debug ephone state
Ccme# debug ephone mwi
.............
Jun 11 05:51:57.663: calling [private] called [xx225010]
Jun 11 05:51:57.663: SkinnyTryCall to 5010 instance 1 start at 0SkinnyTryCall to
  5010 instance 1 match DN 1
Jun 11 05:51:57.663: Phone 0 DN 1 MWI on 0 messages
Jun 11 05:51:57.667: ephone-1[1]:Set MWI line 1 to ON count 0
Jun 11 05:51:57.667: ephone-1[1]:Set MWI line 0 to ON count 0
Jun 11 05:51:57.667: UpdateCallState DN 10 chan 1 operating in mode 1
Jun 11 05:52:01.874: ephone-1[1]:ONHOOK (internal)
Jun 11 05:52:01.874: ephone-1[1]:call clean up this DN 1 chan 1 was calling other
  DN -1 chan 1
Jun 11 05:52:01.874: this ephone-1 other ephone-(-1) other DN state UNKNOWN
Jun 11 05:52:01.878: ephone-1[1]:CloseReceive
Jun 11 05:52:01.878: ephone-1[1]:StopMedia
Jun 11 05:52:01.878: DN 1 chan 1 End Voice_Mode
```

```
Jun 11 05:52:01.878: ephone-1[1]:SetCallState line 1 DN 1 chan 1 ref 13 TsOnHook
Jun 11 05:52:01.882: ephone-1[1]:SpeakerPhoneOnHook
Jun 11 05:52:01.886: ephone-1[1]:SpeakerPhoneOnHook
```

# Troubleshooting Voice Profile for Internet Mail Networking

Cisco Unity Express voice mail networking using Voice Profile for Internet Mail (VPIM) can be configured to interoperate with other Cisco Unity Express systems and Cisco Unity systems in a network. The following sections describe some common problems faced in deploying this feature and how to troubleshoot and fix them:



## Cannot Send and Receive Network Messages from a Location

If you notice that you cannot send or receive any messages from a particular location, a probable cause is that the local location ID is not configured in the system. Until you configure this on a Cisco Unity Express system, voice mail networking is not enabled. If you are trying to send a network message from a system where a local location ID is not configured, the system plays the prompt “Sorry. The extension you requested is not available.” You can verify whether you have configured a local location by using the Cisco Unity Express CLI shown inthe following example:

```
Cue# show network locations
ID        NAME                                ABBREV DOMAIN
303       'Boston'                            BOS    cueunity.cisco.com
401       'Bangalore'                         BAN    bang.cue.domain-name
201       'Los Angeles'                       LAX    lax.cue.domain-name

Local location id:
```

In the preceding example, the local location ID is empty, which indicates that voice mail networking is disabled. You can configure this parameter using the following CLI to set the local location ID to 401:

```
Cue(config)# network local location id 401
```

## Cannot Send Messages to Cisco Unity

Assume that you have a network with a centralized Cisco Unity at the main campus site, smaller offices with Cisco Unity Expresss, and voice mail networking configured between all the sites. You might be able to send messages between the Cisco Unity Express sites and send a message from Cisco Unity to Cisco Unity Express, but you are unable to send a message from Cisco Unity Express to Cisco Unity.

The reason for this problem might be that the sending Cisco Unity Express site is configured in the same domain as that of the Cisco Unity system. The Microsoft Exchange server used by Cisco Unity does not allow other VPIM locations to be in the same domain as itself. For example, the network location configuration shown in the following example does not work if iptel.cisco.com points to a Cisco Unity server.

```
Cue# show network locations
ID        NAME                                ABBREV DOMAIN
303       'Boston'                            BOS    iptel.cisco.com
401       'Bangalore'                         BAN    bang.iptel.cisco.com
201       'Los Angeles'                       LAX    lax.iptel.cisco.com
```

Separating the domains for the Cisco Unity Express and Cisco Unity systems fixes the problem, as shown in the following example:

```
Cue# show network locations
ID        NAME                                ABBREV DOMAIN
303       'Boston'                            BOS    bos.iptel.cisco.com
401       'Bangalore'                         BAN    bang.cue.domain-name
201       'Los Angeles'                       LAX    lax.cue.domain-name
```

When deploying Cisco Unity Express and Cisco Unity voice mail networking, remember that DNS is mandatory in such a network. Cisco Unity is not supported in a network where DNS service is unavailable. It is important that all the network locations on Cisco Unity Express and Cisco Unity be configured using host names and domain names, not explicit IP addresses. If you have a network of Cisco Unity Expresss only, you can use explicit IP addresses or DNS host names in network location configurations. This alternative configuration eliminates the dependency on DNS service or DNS cache consistency for voice mail networking.

# Unable to Address Messages to Network Locations

There can be occasions when users are unable to address messages to network locations. The Cisco Unity Express system plays a message informing the user that the extension he or she dialed is unavailable. The probable reason for this issue is a misconfiguration in the network location setting with respect to extension lengths. For example, if you have accidentally configured an extension length of 3 for a location where the actual extension length is 4, the system responds with this message when a user addresses a message with a four digit extension. You can verify the configuration as shown in the following example.

```
Bang# show network detail location id 201
Name:                                Los Angeles
Abbreviation:                        LAX
Email domain:                        lax.cue.domain-name
Minimum extension length:            4
Maximum extension length:            4
Phone prefix:
VPIM encoding:                       G711ulaw
Send spoken name:                    enabled
Sent msg count:                      1
Received msg count:                  0
```

# Troubleshooting Nondelivery Receipts

When you send voice mail messages to different networked locations, you might receive nondelivery receipts (NDRs) for many reasons. When you listen to the NDR, it describes why that particular message could not be delivered.

The following sections discuss the most common reasons why a message cannot be delivered. It also covers how to identify the underlying problem and correct it.

## Invalid Extension at the Receiving System

The invalid extension error at the receiving system occurs when a message is sent to an extension that does not exist on the destination system's configuration. You can verify whether the extension the message is addressed to is available on the destination system by checking the extension in the mailbox page of the Cisco Unity Express GUI.

Another reason you might run into this situation is that in your network location configuration for the recipient location, you might have configured **voicemail extension-length min** and **voicemail extension-length max**. If this is the configuration, and the user dials the wrong length for a location and extension when addressing the message, Cisco Unity Express cannot detect that until the message reaches the destination system. So the message is sent anyway, resulting in an invalid extension NDR from the destination system. For example, assume that you have a location called Boston where all the extensions are four digits in length. At another location in the network, such as Los Angeles, you have a configuration for the Boston location, as shown in the following example.

```
network location id 303
 abbreviation "BOS"
 email domain cueunity.cisco.com
 name "Boston"
 voicemail extension-length min 3 max 10
 end location
```

This configuration allows users to address a network message to location 303 (Boston) with any strings of digits between three and ten digits in length, even though all the extensions in Boston are four digits long. So if a user makes a mistake in addressing the message, the sending system cannot detect it. Use the minimum and maximum extension length configuration if you have mixed-length extensions. Otherwise, it is recommended that you use a fixed-extension-length value to minimize addressing errors.

## Remote Location Is Unavailable

The destination location might be unavailable for a few reasons, including the following:

- The Cisco Unity Express system is offline for an extended period of time.
- The network link serving the location might be down.

A simple IP **ping** test can confirm the network status.

## DNS Service Is Unavailable and Local DNS Cache Is Inconsistent

If you are using host and domain names in your network location configurations, the availability of DNS service is compulsory to send messages across the network. Cisco Unity Express has a DNS cache, so even if the DNS server on the network is unavailable for a brief period, Cisco Unity Express should still be able to send messages to remote locations as long as the cache is consistent and has entries for the location the message is addressed to.

You can verify the availability of a DNS server using IP **ping**. Cisco Unity Express checks the Mail Exchange (MX) DNS record for a remote location, so it is important that an MX record and a normal DNS record are available for the remote location. You can check Cisco Unity Express's DNS cache by using the **show ip dns cache** command, shown in the following example, to check the consistency of the entries.

```
Bang# show ip dns cache
Bang.localdomain.       2xx7yy3647 IN A         a.4.13.90
a.0.0.127.in-addr.arpa. 2xx7yy3647 IN PTR       localhost.
localhost.              2xx7yy3647 IN A         127.0.0.1
90.13.4.1.in-addr.arpa. 2xx7yy3647 IN PTR       BANG.localdomain.
lax.cue.domain-name.          3600 IN MX        10 lax.cue.domain-name.
lax.cue.domain-name.          3600 IN A         a.4.14.134
```

If the IP address of the remote Cisco Unity Express has changed since this cache was last updated, voice mail messages cannot be sent to this location. As soon as the DNS server is again available, you can use the **clear ip dns cache** command to clear the cache, and build a new one with updated values.

### Recipient Mailbox at the Remote Location Is Full or Disabled

If the recipient's mailbox is full or disabled, an NDR is generated by the destination Cisco Unity Express system. You can check mailbox usage by using the CLI described in the "Common Voice Mail show Commands" section on page 189.

### Recipient Location Has No Configuration for the Sending Location

For successful operation of Cisco Unity Express voice mail networking, it is important that each location know about every other location in the network. For security reasons, Cisco Unity Express does not accept any incoming messages from an IP address or host name that is not present in its network location configuration. You can verify the location configuration by using the **show network locations** command.

## Tracing Voice Mail Networking

The preceding sections described common problems and their solutions. This section demonstrates how to debug voice mail networking with tracing. Voice mail networking is implemented using VPIM, which in turn uses Simple Mail Transfer Protocol (SMTP) for message transfer.

When you troubleshoot, a basic understanding of SMTP is helpful. The following example shows the output of a networking trace in which a message was sent to a nonexistent mailbox and an NDR was received by the sending system. The first half of the trace shows a voice mail message being sent from extension 5501 at the sending location to extension 9008 at the receiving side. The second half of the trace shows an NDR coming back from the remote location to 5501 with a "mailbox full" reason. The important information to look for while troubleshooting is as follows (also highlighted in the trace output):

- The EHLO SMTP message sent and received by the systems.
- The MAIL FROM and RCPT TO headers, which identify the sender and recipient's information.
- The From and To VPIM headers.
- The VPIM message ID. This is a globally unique message ID that can be useful in troubleshooting on the receiving side.

- The vCard information sent and received. This is useful in Cisco Unity Express 2.1 and later when the vCard information is cached on the receiving system to identify remote users.
- The encoding format for the audio message. This can be G.711 or G.726 (32-KB Adaptive Differential Pulse Code Modulation [ADPCM]).
- In the NDR message delivery, the NDR status that gives the reason why the message could not be delivered.

```
Cue# trace networking smtp all
Cue# trace networking vpim send
Cue# trace networking vpim receive
Cue# show trace buffer tail
Press <CTRL-C> to exit...
6018 12/26 22:56:50.091 netw smtp 3 192.168.0.200
6018 12/26 22:56:50.106 netw smtp 4
6018 12/26 22:56:50.129 netw smtp 6 220 192.168.0.200 Simple Mail Transfer Service
  Ready
6018 12/26 22:56:50.129 netw smtp 5 EHLO
6018 12/26 22:56:50.137 netw smtp 6 250-192.168.0.200 (Cisco Unity Express)
6018 12/26 22:56:50.137 netw smtp 6 250-X-VPIM-Wave
6018 12/26 22:56:50.139 netw smtp 6 250-DSN NOTIFY
6018 12/26 22:56:50.141 netw smtp 6 250 SIZE
6018 12/26 22:56:50.714 netw smtp 5 MAIL FROM 5501@192.168.0.100
6018 12/26 22:56:50.729 netw smtp 6 250 ok
6018 12/26 22:56:50.730 netw smtp 5 RCPT TO 9008@192.168.0.200
6018 12/26 22:56:50.736 netw smtp 6 250 ok
6018 12/26 22:56:50.737 netw smtp 5 DATA
6018 12/26 22:56:50.743 netw smtp 6 354 Start mail input; end with <CRLF>.<CRLF>
6018 12/26 22:56:50.750 netw vpim 3 VPIM
6018 12/26 22:56:50.793 netw vpim 3 VPIM: To: <9008@192.168.0.200>
6018 12/26 22:56:50.811 netw vpim 3 VPIM: From: Auto SubOne<5501@192.168.0.100>
6018 12/26 22:56:50.867 netw vpim 3 VPIM: Date: Sun, 26 Dec 2004 22:56:49 -0800
  (PST)
6018 12/26 22:56:50.867 netw vpim 3 VPIM: MIME-Version: 1.0 (Voice 2.0)
6018 12/26 22:56:50.867 netw vpim 3 VPIM: Content-Type: Multipart/Voice-Message;
  Version=2.0;
6018 12/26 22:56:50.867 netw vpim 3 VPIM:       Boundary="==VpimMsg==1104130610745"
6018 12/26 22:56:50.868 netw vpim 3 VPIM: Content-Transfer-Encoding: 7bit
6018 12/26 22:56:50.868 netw vpim 3 VPIM: Message-ID:
  <JAB054980L7-NM-JAD06390I66-1104130214514>
6018 12/26 22:56:50.869 netw vpim 3 VPIM:
6018 12/26 22:56:50.872 netw vpim 3 VPIM: --==VpimMsg==1104130610745
6018 12/26 22:56:50.872 netw vpim 3 VPIM: Content-Type: text/directory;
  charset=us-ascii; profile=vCard
6018 12/26 22:56:50.872 netw vpim 3 VPIM: Content-Transfer-Encoding: 7bit
6018 12/26 22:56:50.873 netw vpim 3 VPIM: Content-Disposition: attachment;
  filename="Auto SubOne.vcf"
6018 12/26 22:56:50.873 netw vpim 3 VPIM:
6018 12/26 22:56:50.873 netw vpim 3 VPIM: BEGIN:vCard
6018 12/26 22:56:50.873 netw vpim 3 VPIM: FN:Auto SubOne
6018 12/26 22:56:50.874 netw vpim 3 VPIM: EMAIL;TYPE=INTERNET;
  TYPE=VPIM:5501@192.168.0.100
6018 12/26 22:56:50.874 netw vpim 3 VPIM: TEL:5501
6018 12/26 22:56:50.874 netw vpim 3 VPIM: VERSION: 3.0
6018 12/26 22:56:50.874 netw vpim 3 VPIM: END:vCard
6018 12/26 22:56:50.874 netw vpim 3 VPIM:
6018 12/26 22:56:50.909 netw vpim 3 VPIM: --==VpimMsg==1104130610745
6018 12/26 22:56:50.909 netw vpim 3 VPIM: Content-Type: Audio/x-wav
6018 12/26 22:56:50.909 netw vpim 3 VPIM: Content-Transfer-Encoding: Base64
6018 12/26 22:56:50.910 netw vpim 3 VPIM: Content-Description: VPIM Message
6018 12/26 22:56:50.910 netw vpim 3 VPIM: Content-Disposition: inline;
  voice=Voice-Message
6018 12/26 22:56:50.910 netw vpim 3 VPIM: Content-ID:
```

```
  JAB054980L7-NM-JAD06390I66-1104130214514
6018 12/26 22:56:50.910 netw vpim 3 VPIM:
6018 12/26 22:56:51.124 netw vpim 3 VPIM:
6018 12/26 22:56:51.125 netw vpim 3 VPIM: ----VpimMsg==1104130610745--
6018 12/26 22:56:51.143 netw smtp 5 End of DATA
6018 12/26 22:56:54.230 netw smtp 6 250 2.6.0 Message queued for delivery
5762 12/26 22:56:55.242 netw smtp 2
5762 12/26 22:56:55.247 netw smtp 3 socket hostName: 192.168.0.200,
  hostAddress: 192.168.0.200
5762 12/26 22:56:55.248 netw smtp 3 hostname: 192.168.0.200 found in good address
  cache
5762 12/26 22:56:55.255 netw smtp 1
6023 12/26 22:56:55.257 netw smtp 5 Initial connection message
6023 12/26 22:56:55.269 netw smtp 6 UNKNOWN: EHLO 192.168.0.200
6023 12/26 22:56:55.269 netw smtp 5 250-192.168.0.100
6023 12/26 22:56:55.774 netw smtp 6 EHLO : MAIL FROM: <9008@192.168.0.200>
6023 12/26 22:56:55.777 netw smtp 5 250 ok
6023 12/26 22:56:55.791 netw smtp 6 MAIL FROM:: RCPT TO: <5501@192.168.0.100>
6023 12/26 22:56:55.792 netw smtp 5 250 ok
6023 12/26 22:56:55.797 netw smtp 6 RCPT TO:: DATA
6023 12/26 22:56:55.797 netw smtp 5 354 Start data
6023 12/26 22:56:55.895 netw vpim 4 VPIM: To: <5501@192.168.0.100>
6023 12/26 22:56:55.896 netw vpim 4 VPIM: From: <9008@192.168.0.200>
6023 12/26 22:56:55.897 netw vpim 4 VPIM: Date: Sun, 26 Dec 2004 22:56:49 -0800 (PST)
6023 12/26 22:56:55.900 netw vpim 4 VPIM: MIME-Version: 1.0 (Voice 2.0)
6023 12/26 22:56:55.902 netw vpim 4 VPIM: Content-Type: Multipart/report;
  report-type=delivery-status;Boundary="==VpimNdrMsg==1104187978982"
6023 12/26 22:56:55.906 netw vpim 4 VPIM: Message-ID:
  <JAB054980L7-NM-JAD06390I66-1104130214514@192.168.0.200>
6023 12/26 22:56:55.907 netw vpim 4 VPIM:
6023 12/26 22:56:55.910 netw vpim 4 VPIM: ----VpimNdrMsg==1104187978982
6023 12/26 22:56:55.919 netw vpim 4 NDR: Content-Type: text/plain
6023 12/26 22:56:55.920 netw vpim 4 NDR:
6023 12/26 22:56:55.921 netw vpim 4 NDR: Your message could not be delivered
6023 12/26 22:56:55.922 netw vpim 4 NDR:
6023 12/26 22:56:55.924 netw vpim 4 NDR: ----VpimNdrMsg==1104187978982
6023 12/26 22:56:55.925 netw vpim 4 NDR: Content-Type: message/delivery-status
6023 12/26 22:56:56.029 netw vpim 4 NDR:
6023 12/26 22:56:56.031 netw vpim 4 NDR: Reporting-MTA: dns; CUE
6023 12/26 22:56:56.033 netw vpim 4 NDR: Original-Recipient: rfc822; 9008@192.168.0.200
6023 12/26 22:56:56.034 netw vpim 4 NDR: Final-Recipient: rfc822; 9008@192.168.0.200
6023 12/26 22:56:56.036 netw vpim 4 NDR: Action: failed
6023 12/26 22:56:56.038 netw vpim 4 NDR: Status: 5.1.1 (Mailbox does not exist)
6023 12/26 22:56:56.043 netw vpim 4 NDR: Last-Attempt-Date: Sun, 26 Dec 2004
  22:56:49 -0800 (PST)
6023 12/26 22:56:56.044 netw vpim 4 NDR:
6023 12/26 22:56:56.046 netw vpim 4 NDR: ----VpimNdrMsg==1104187978982
6023 12/26 22:56:56.047 netw vpim 4 NDR: Content-type: Message/RFC822
6023 12/26 22:56:56.048 netw vpim 4 VPIM: Content-Transfer-Encoding: 7bit
6023 12/26 22:56:56.050 netw vpim 4 VPIM:
6023 12/26 22:56:56.051 netw vpim 4 VPIM: To: <9008@192.168.0.200>
6023 12/26 22:56:56.053 netw vpim 4 VPIM: From: <5501@192.168.0.100>
6023 12/26 22:56:56.054 netw vpim 4 VPIM: Date: Sun, 26 Dec 2004 22:56:49 -0800
  (PST)
6023 12/26 22:56:56.058 netw vpim 4 VPIM: MIME-Version: 1.0 (Voice 2.0)
6023 12/26 22:56:56.059 netw vpim 4 VPIM: Content-Type: Multipart/Voice-Message;
  Version=2.0;Boundary="==VpimMsg==1104187979064"
6023 12/26 22:56:56.061 netw vpim 4 VPIM: Content-Transfer-Encoding: 7bit
6023 12/26 22:56:56.063 netw vpim 4 VPIM: Message-ID:
  <JAB054980L7-NM-JAD06390I66-1104130214514>
6023 12/26 22:56:56.064 netw vpim 4 VPIM:
6023 12/26 22:56:56.066 netw vpim 4 VPIM: ----VpimMsg==1104187979064
6023 12/26 22:56:56.070 netw vpim 4 VPIM: Content-Type: Audio/32KADPCM
6023 12/26 22:56:56.071 netw vpim 4 VPIM: Content-Transfer-Encoding: Base64
```

```
6023 12/26 22:56:56.073 netw vpim 4 VPIM: Content-Description: VPIM Message
6023 12/26 22:56:56.084 netw vpim 4 VPIM: Content-Disposition: inline;
  voice=Voice-Message
6023 12/26 22:56:56.085 netw vpim 4 VPIM: Content-ID:
  JAB054980L7-NM-JAD06390I66-1104130214514
6023 12/26 22:56:56.144 netw vpim 5 83787
6023 12/26 22:56:56.144 netw vpim 8
6023 12/26 22:56:56.496 netw vpim 6 16074
6023 12/26 22:56:57.228 netw vpim 6 15446
6023 12/26 22:56:57.457 netw vpim 10
6023 12/26 22:56:57.602 netw vpim 4 VPIM: --==VpimMsg==1104187979064--
6023 12/26 22:56:57.603 netw vpim 4 VPIM: --==VpimNdrMsg==1104187978982--
6023 12/26 22:56:57.621 netw vpim 4 NDR: .
6023 12/26 22:56:58.000 netw smtp 5 260 Message queued
6023 12/26 22:56:58.016 netw smtp 6 DATA: RSET
6023 12/26 22:56:58.017 netw smtp 6 RSET: QUIT
6023 12/26 22:56:58.017 netw smtp 5 221 closing channel
```