Advanced Troubleshooting Topics

This chapter contains advanced troubleshooting topics. In most cases, the information found in this chapter will not be needed for troubleshooting Bridge Networking problems. When you are troubleshooting a problem, always begin with the "Troubleshooting Bridge Networking" chapter.

See the following sections:

Enabling and Interpreting Traces for the Brooktrout TR114 on the Bridge

When experiencing analog communication problems between the Bridge and remote Octel nodes, the sflogs on the Bridge are most useful in determining the activity that is occurring on the analog calls. However, it is sometimes helpful to begin by determining what is occurring right at the TR114 voice board, as this can help pinpoint the exact source of the problem. This section describes how to enable and interpret traces of activity on the TR114 board.

Enabling Debug Traces for the Brooktrout TR114

To Enable Debug Traces for the Brooktrout TR114

Step 1 Browse to the directory **<Drive>:\<Path>\Starfish\Bin**, where <Drive> and <Path> denote the drive and the topmost directory where the Bridge software is installed.

Step 2 Open the file **btcall.cfg** in Notepad.

Step 3 Add the following line at the bottom of the file:

debug ./traceinfo.log

Step 4 Save the changes to btcall.cfg and close the file.

Step 5 Restart the Unity Bridge service from the Services Control Panel.

Brooktrout TR114 activity will now be logged to <BridgePath>\Starfish\Bin\Traceinfo.log. Be aware that each time the Unity Bridge service is restarted, the old traceinfo.log will be deleted and a fresh log begun.

If you need to restart the Unity Bridge service and want to save the traceinfo.log from the previous test, use the following procedure.

To Save the traceinfo.log from a Previous Test

Step 1 Stop the Unity Bridge service from the Services Control Panel.

Step 2 Rename **<BridgePath>\Starfish\Bin\Traceinfo.log** to a filename of your choice.

Step 3 Start the Unity Bridge service from the Services Control Panel.

Note On Bridge servers where message traffic is heavy, the traceinfo.log file can grow very large. There is no mechanism for limiting the size of traceinfo.log, and no cycling occurs. When you have finished troubleshooting, you should disable the tracing.

Disabling Debug Traces for the Brooktrout TR114

To Disable Debug Traces for the Brooktrout TR114

Step 1 Browse to the directory **<BridgePath>\Starfish\Bin**.

Step 2 Open the file **btcall.cfg** in Notepad.

Step 3 Remove the **debug ./traceinfo.log** line from the bottom of the file.

Step 4 Save the changes to btcall.cfg and close the file.

Step 5 Restart the Unity Bridge service from the Services Control Panel.

Interpreting Debug Traces for the Brooktrout TR114

Because the Brooktrout TR114 traceinfo.log is run on the same server as the sflog.*.log files, the time stamps will match up. This allows you to use these logs together to get a good picture of what is going on. See the description of the Call Tracing Level field in the “System Settings” section for information about enabling the sflog.*.log files.

Following is an excerpt from a traceinfo.log on a 4-port Bridge server for an outgoing analog message delivery from the Bridge (serial number 80100) to a remote Octel node (serial number 80200). Descriptions of the events are included, as well as the associated events from sflog.log. The excerpt displays the context of actions the board is taking based on requests from the Unity Bridge service (starfish.exe) and vice versa. In the following example traces, the sflog traces begin with the word

"SFLOG" followed by a number. Only the sflog.log events for line 3 (TR114 channel 2) are listed. The traceinfo.log events begin with the date and time followed by the TR114 channel on which the event occurred.

Table 7-1 Brooktrout and SFLOG Example Trace

Trace	Description
11/26 22:59:08.33 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:08.34 1 cmd: 0x26 DIS_RING_DET 0x0 **11/26 22:59:08.34 2 cmd: 0x26 DIS_RING_DET 0x0** 11/26 22:59:08.35 3 cmd: 0x26 DIS_RING_DET 0x0 **11/26 22:59:08.37 2 await ring** **11/26 22:59:08.37 2 cmd: 0x1f NUM_RINGS 0x1** **11/26 22:59:08.37 2 cmd: 0x27 ENA_RING_DET 0x0** 11/26 22:59:08.39 1 await ring 11/26 22:59:08.39 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:08.39 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:08.41 0 await ring 11/26 22:59:08.41 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:08.41 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:08.41 3 await ring 11/26 22:59:08.41 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:08.41 3 cmd: 0x27 ENA_RING_DET 0x0 **11/26 22:59:18.37 2 cmd: 0x26 DIS_RING_DET 0x0** 11/26 22:59:18.39 1 cmd: 0x26 DIS_RING_DET 0x0	All of the analog ports on the Bridge are in an idle state. They repeatedly cycle through their ring detect processes to monitor for incoming calls until they get one, or until an outgoing call is initiated by the Unity Bridge service (starfish.exe). The number immediately following the time stamp is the channel number on the TR114. Note that it is zero based, that is, channels 0–3 correspond to lines 1–4 on the Bridge server. The traces for channel 2 (which is Bridge line 3) are in bold.
SFLOG 1396 1700 2002/11/26-22:59:18.384 00000008 Line 3: Call Out Process Initiated for Node 80200 Window Type 0	The Bridge initiates the call out process for node 80200 normal messages (type 0). There is a message to deliver, and therefore a request is sent to the TR114 to make the call.
11/26 22:59:18.39 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 22:59:18.39 2 dialing ,20 11/26 22:59:18.39 2 cmd: 0x37 DIAL_STRING 0x0	These three events initiate an outgoing analog call on channel 2 (Bridge line 3). The dial string is ",20" where the comma is a one second pause and 20 is the phone number being dialed.
11/26 22:59:18.41 1 await ring 11/26 22:59:18.41 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:18.41 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:18.41 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:18.44 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:18.44 0 await ring 11/26 22:59:18.44 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:18.44 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:18.45 3 await ring 11/26 22:59:18.45 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:18.45 3 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:20.15 2 intr: 0xa ISTRDNE 0x0 11/26 22:59:20.15 2 cmd: 0x2b ENA_CALL_PROG 0x23	Call progress detection is initiated on the TR114. The TR114 waits for detection of ringback, busy or reorder tone, or detection of a human voice.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
1/26 22:59:20.60 2 intr: 0x18 ILOW 0x40 11/26 22:59:20.89 2 intr: 0x19 IHI 0x3a 11/26 22:59:22.14 2 intr: 0x18 ILOW 0xfa 11/26 22:59:22.27 2 intr: 0x19 IHI 0x1a 11/26 22:59:22.77 2 intr: 0x18 ILOW 0x64 11/26 22:59:22.82 2 intr: 0x19 IHI 0xa 11/26 22:59:22.95 2 intr: 0x18 ILOW 0x1a 11/26 22:59:22.95 2 intr: 0x1b CALL_PROG 0x10 111/26 22:59:22.95 2 call progress: HUMAN	Voice detection has occurred. A connect event is sent to the Unity Bridge service.
11/26 22:59:22.95 2 cmd: 0x2a DIS_CALL_PROG 0x0 11/26 22:59:22.95 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:22.95 2 cmd: 0x24 ENABLE_TONE_DET 0x0 SFLOG 1396 1700 2002/11/26-22:59:22.960 00000100 Line 3: Call Status = Answer SFLOG 1396 1700 2002/11/26-22:59:22.960 00000100 Line 3: Playing BD	The Unity Bridge service acknowledges that the call has been answered and sends the tones “BD” to the Octel. DTMF tones are stored in.sph files in \Bridge\Starfish\Bin and played through the TR114 as audio files.
11/26 22:59:22.97 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:22.99 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of “BD” and the associated parameters, and begins to play them to the Octel.
11/26 22:59:22.99 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:23.02 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:23.29 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel. You will see only one pair of SPEECH_START/SPEECH_DONE events for each string of DTMF digits sent. Because the TR114 is simply playing recordings of DTMFs given to it by the Unity Bridge service, you will see the actual DTMF digits sent only in the sflogs.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
11/26 22:59:23.29 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:24.46 2 intr: 0xb TONE_DETECT 0x8f 11/26 22:59:24.53 2 intr: 0xb TONE_DETECT 0xf 11/26 22:59:24.60 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:24.67 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:24.75 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:24.82 2 intr: 0xb TONE_DETECT 0x0 **Description of Tones Received** The first tone received is "C" (because f=C). This tone began at 22:59:24.46 (0x8f) and ended at 22:59:24.53 (0xf); thus it was 70 milliseconds in duration. The second tone received is "D" (because 0=D). This tone began at 22:59:24.60 (0x80) and ended at 22:59:24.67 (0x0); thus it was 70 milliseconds in duration. The interdigit time between the "C" and the first "D" was 70 milliseconds (22:59:24.60 – 22:59:24.53 = .07 seconds, or 70 milliseconds). The third tone received is "D" (because 0=D). This tone began at 22:59:24.75 (0x80) and ended at 22:59:24.82 (0x0); thus it was 70 milliseconds in duration. The interdigit time between the first "D" and the second "D" was 80 milliseconds (22:59:24.75 – 22:59:24.67 = .08 seconds, or 80 milliseconds).	The TR114 detects incoming DTMF digits from the Octel. There are two TONE_DETECT events for each tone detected, one when the beginning of the tone is detected and one when the end is detected. The event for detection of the beginning of a tone will look like "intr: 0xb TONE_DETECT 0x8c," where the "8" indicates the beginning of the tone and the "c" indicates what tone was detected. The end of the tone will look just the same, but will not include the "8" (for example, "intr: 0xb TONE_DETECT 0xc"). Note the "c" here does not represent the actual DTMF "C" tone. Use the following table to translate what the digit signifies: 1–9 = 1–9 a = 0 b = * c = # d = A e = B f = C 0 = D
SFLOG 1396 1700 2002/11/26-22:59:25.835 00000100 Line 3: Received CDD	The "CDD" detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log.
SFLOG 1396 1700 2002/11/26-22:59:25.835 00000100 Line 3: Playing 12086D5082AC6AA897	The Unity Bridge services responds to the CDD packet by sending a request to the TR114 to play audio of the DTMF digit string "12086D5082AC6AA897."
11/26 22:59:25.82 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 22:59:25.82 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:25.83 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "12086D5082AC6AA897" and the associated parameters, and begins to play them to the Octel.
11/26 22:59:25.85 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:25.94 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:28.38 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
11/26 22:59:28.38 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:28.42 1 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:28.43 1 await ring 11/26 22:59:28.43 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:28.43 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:28.45 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:28.46 0 await ring 11/26 22:59:28.46 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:28.46 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:28.46 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:28.47 3 await ring 11/26 22:59:28.47 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:28.47 3 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:29.56 2 intr: 0xb TONE_DETECT 0x81 11/26 22:59:29.63 2 intr: 0xb TONE_DETECT 0x1 11/26 22:59:29.70 2 intr: 0xb TONE_DETECT 0x83 11/26 22:59:29.77 2 intr: 0xb TONE_DETECT 0x3 11/26 22:59:29.85 2 intr: 0xb TONE_DETECT 0x87 11/26 22:59:29.92 2 intr: 0xb TONE_DETECT 0x7 11/26 22:59:29.99 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:30.06 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:30.13 2 intr: 0xb TONE_DETECT 0x87 11/26 22:59:30.19 2 intr: 0xb TONE_DETECT 0x7 11/26 22:59:30.26 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:30.33 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:30.40 2 intr: 0xb TONE_DETECT 0x85 11/26 22:59:30.47 2 intr: 0xb TONE_DETECT 0x5 11/26 22:59:30.54 2 intr: 0xb TONE_DETECT 0x8d 11/26 22:59:30.61 2 intr: 0xb TONE_DETECT 0xd 11/26 22:59:30.68 2 intr: 0xb TONE_DETECT 0x88 11/26 22:59:30.75 2 intr: 0xb TONE_DETECT 0x8 11/26 22:59:30.82 2 intr: 0xb TONE_DETECT 0x8c 11/26 22:59:30.89 2 intr: 0xb TONE_DETECT 0xc 11/26 22:59:30.97 2 intr: 0xb TONE_DETECT 0x87 11/26 22:59:31.04 2 intr: 0xb TONE_DETECT 0x7 11/26 22:59:31.09 2 intr: 0xb TONE_DETECT 0x8a 11/26 22:59:31.16 2 intr: 0xb TONE_DETECT 0xa 11/26 22:59:31.24 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:31.31 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:31.38 2 intr: 0xb TONE_DETECT 0x88 11/26 22:59:31.45 2 intr: 0xb TONE_DETECT 0x8	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can map the TONE_DETECT events to DTMF digits, as follows: 1 3 7 0 7 0 5 d 8 c 7 a 0 8 1 3 7 D 7 D 5 A 8 # 7 0 D 8 (The first line contains the TONE_DETECT events, and the second line contains the DTMF digits.)
SFLOG 1396 1700 2002/11/26-22:59:32.464 00000100 Line 3: Received 137D7D5A8#70D8 SFLOG 1396 1700 2002/11/26-22:59:32.464 00000100 Line 3: Protocol Level = 3	The “137D7D5A8#70D8” detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log. From the received packet the Unity Bridge service can determine the analog Octel networking protocol level used by the Octel server.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-22:59:32.464 00000100 Line 3: Playing 2431#73258#51#4B	The Unity Bridge services responds to the received packet by sending a request to the TR114 to play audio of the DTMF digit string "2431#73258#51#4B."
11/26 22:59:32.45 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 22:59:32.45 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:32.46 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "2431#73258#51#4B" and the associated parameters, and begins to play them to the Octel.
11/26 22:59:32.49 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:32.52 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:34.74 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel.
11/26 22:59:34.74 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:35.92 2 intr: 0xb TONE_DETECT 0x8a 11/26 22:59:35.99 2 intr: 0xb TONE_DETECT 0xa 11/26 22:59:36.06 2 intr: 0xb TONE_DETECT 0x86 11/26 22:59:36.13 2 intr: 0xb TONE_DETECT 0x6 11/26 22:59:36.20 2 intr: 0xb TONE_DETECT 0x89 11/26 22:59:36.27 2 intr: 0xb TONE_DETECT 0x9 11/26 22:59:36.35 2 intr: 0xb TONE_DETECT 0x87 11/26 22:59:36.42 2 intr: 0xb TONE_DETECT 0x7 11/26 22:59:36.49 2 intr: 0xb TONE_DETECT 0x8c 11/26 22:59:36.56 2 intr: 0xb TONE_DETECT 0xc 11/26 22:59:36.63 2 intr: 0xb TONE_DETECT 0x87 11/26 22:59:36.69 2 intr: 0xb TONE_DETECT 0x7 11/26 22:59:36.76 2 intr: 0xb TONE_DETECT 0x86 11/26 22:59:36.83 2 intr: 0xb TONE_DETECT 0x6 11/26 22:59:36.90 2 intr: 0xb TONE_DETECT 0x8a 11/26 22:59:36.97 2 intr: 0xb TONE_DETECT 0xa 11/26 22:59:37.04 2 intr: 0xb TONE_DETECT 0x85 11/26 22:59:37.11 2 intr: 0xb TONE_DETECT 0x5 11/26 22:59:37.18 2 intr: 0xb TONE_DETECT 0x88 11/26 22:59:37.25 2 intr: 0xb TONE_DETECT 0x8 11/26 22:59:37.32 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:37.39 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:37.47 2 intr: 0xb TONE_DETECT 0x85 11/26 22:59:37.54 2 intr: 0xb TONE_DETECT 0x5 11/26 22:59:37.61 2 intr: 0xb TONE_DETECT 0x88 11/26 22:59:37.66 2 intr: 0xb TONE_DETECT 0x8 11/26 22:59:37.74 2 intr: 0xb TONE_DETECT 0x8d 11/26 22:59:37.81 2 intr: 0xb TONE_DETECT 0xd 11/26 22:59:37.88 2 intr: 0xb TONE_DETECT 0x82 11/26 22:59:37.95 2 intr: 0xb TONE_DETECT 0x2 11/26 22:59:38.02 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:38.09 2 intr: 0xb TONE_DETECT 0x0	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can see that the TONE_DETECT events correspond to the following DTMF digits: 0 6 9 7 # 7 6 0 5 8 D 5 8 A 2 D

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
1/26 22:59:38.43 1 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:38.44 1 await ring 11/26 22:59:38.44 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:38.44 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:38.46 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:38.47 0 await ring 11/26 22:59:38.47 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:38.47 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:38.47 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:38.48 3 await ring 11/26 22:59:38.48 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:38.48 3 cmd: 0x27 ENA_RING_DET 0x0 SFLOG 1396 1700 2002/11/26-22:59:39.104 00000100 Line 3: Received 0697#76058D58A2D SFLOG 1396 1700 2002/11/26-22:59:39.104 00000008 Line 3: Call established from Serial # 80100 to Serial # 80200	The “0697#76058D58A2D” detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log. From the received packet the Unity Bridge service is able to determine that Octel node #80200 has accepted our call and will allow us to proceed.
SFLOG 1396 1700 2002/11/26-22:59:39.114 00000100 Line 3: Playing 131A3C13#CAA7A*7AB72*C9D5A647*69D#1A	The Unity Bridge service responds to the received packet by sending a request to the TR114 to play audio of the DTMF digit string “131A3C13#CAA7A*7AB72*C9D5A647*69D#1A.”
11/26 22:59:39.10 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 22:59:39.10 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:39.12 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of “131A3C13#CAA7A*7AB72*C9D5A647*69D#1A” and the associated parameters, and begins to play them to the Octel.
11/26 22:59:39.15 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:41.09 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:44.19 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 finishes playing the audio to the Octel.
11/26 22:59:44.19 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:45.36 2 intr: 0xb TONE_DETECT 0x8a 11/26 22:59:45.43 2 intr: 0xb TONE_DETECT 0xa 11/26 22:59:45.51 2 intr: 0xb TONE_DETECT 0x82 11/26 22:59:45.58 2 intr: 0xb TONE_DETECT 0x2 11/26 22:59:45.65 2 intr: 0xb TONE_DETECT 0x84 11/26 22:59:45.72 2 intr: 0xb TONE_DETECT 0x4 11/26 22:59:45.79 2 intr: 0xb TONE_DETECT 0x83 11/26 22:59:45.86 2 intr: 0xb TONE_DETECT 0x3 11/26 22:59:45.92 2 intr: 0xb TONE_DETECT 0x80 11/26 22:59:45.99 2 intr: 0xb TONE_DETECT 0x0 11/26 22:59:46.06 2 intr: 0xb TONE_DETECT 0x81 11/26 22:59:46.13 2 intr: 0xb TONE_DETECT 0x1	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can see that TONE_DETECT events correspond to the following DTMF digits: 0 2 4 3 D 1
SFLOG 1396 1700 2002/11/26-22:59:47.145 00000100 Line 3: Received 0243D1	The “0243D1” detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-22:59:47.145 00000100 Line 3: Playing 67D59228#882313B*0AA*#C#1*DB4579B8D48D7D51A980	The Unity Bridge service responds to the received packet by sending a request to the TR114 to play audio of the DTMF digit string "67D59228#882313B*0AA*#C#1*DB4579B8D48D7D51A980."
11/26 22:59:47.13 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 22:59:47.13 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:47.14 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "67D59228#882313B*0AA*#C#1*DB4579B8D48D7D51A980" and the associated parameters, and begins to play them to the Octel.
11/26 22:59:47.17 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:48.48 1 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:48.48 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:48.49 1 await ring 11/26 22:59:48.49 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:48.49 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:48.49 0 await ring 11/26 22:59:48.49 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:48.49 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:48.49 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:48.50 3 await ring 11/26 22:59:48.50 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:48.50 3 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:50.51 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:53.61 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel.
11/26 22:59:53.61 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 22:59:54.82 2 intr: 0xb TONE_DETECT 0x82 11/26 22:59:54.89 2 intr: 0xb TONE_DETECT 0x2 11/26 22:59:54.95 2 intr: 0xb TONE_DETECT 0x83 11/26 22:59:55.02 2 intr: 0xb TONE_DETECT 0x3 11/26 22:59:55.09 2 intr: 0xb TONE_DETECT 0x8e 11/26 22:59:55.16 2 intr: 0xb TONE_DETECT 0xe 11/26 22:59:55.23 2 intr: 0xb TONE_DETECT 0x8d 11/26 22:59:55.30 2 intr: 0xb TONE_DETECT 0xd 11/26 22:59:55.37 2 intr: 0xb TONE_DETECT 0x8f 11/26 22:59:55.44 2 intr: 0xb TONE_DETECT 0xf 11/26 22:59:55.51 2 intr: 0xb TONE_DETECT 0x8e 11/26 22:59:55.58 2 intr: 0xb TONE_DETECT 0xe	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can see that the TONE_DETECT events correspond to the following DTMF digits: 2 3 B A C B

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-22:59:56.589 00000100 Line 3: Begin Mailbox Update SFLOG 1396 1700 2002/11/26-22:59:56.589 00000100 Line 3: End Mailbox Update SFLOG 1396 1700 2002/11/26-22:59:56.589 00000008 Line 3: Sending Message from mailbox 57100@JDC1 to mailbox 47100@jdc6a.ecsbu-lab-sea.cisco.com SFLOG 1396 1700 2002/11/26-22:59:56.589 00000008 Line 3: Playing Voice SFLOG 1396 1700 2002/11/26-22:59:56.589 00000100 Line 3: Playing c:\Bridge\Starfish\In\80200\80100\20021127065817606-443f73dd-da28-40ad-ae0a-26f2c388ff4f	The Unity Bridge service responds to the received packet by sending a request to the TR114 to play the actual voice message.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
11/26 22:59:56.58 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 22:59:56.59 2 play 1024 11/26 22:59:56.59 2 play 2048 11/26 22:59:56.59 2 play 3072 11/26 22:59:56.59 2 play 4096 11/26 22:59:56.59 2 play 5120 11/26 22:59:56.59 2 play 6144 11/26 22:59:56.59 2 play 7168 11/26 22:59:56.59 2 play 8192 11/26 22:59:56.59 2 cmd: 0x5c SPEECH_START 0x0 11/26 22:59:56.59 2 play 9216 11/26 22:59:56.60 2 play 10240 11/26 22:59:56.60 2 play 11264 11/26 22:59:56.60 2 play 12288 11/26 22:59:56.61 2 play 13312 11/26 22:59:56.61 2 play 14336 11/26 22:59:56.61 2 play 15360 11/26 22:59:56.62 2 play 16384 11/26 22:59:56.62 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 22:59:56.63 2 play 17408 11/26 22:59:56.65 2 play 18432 11/26 22:59:56.68 2 play 19456 11/26 22:59:56.70 2 play 20480 11/26 22:59:56.73 2 play 21504 11/26 22:59:56.75 2 play 22528 11/26 22:59:56.78 2 play 23552 11/26 22:59:56.80 2 play 24576 11/26 22:59:56.83 2 play 25600 11/26 22:59:56.85 2 play 26624 11/26 22:59:56.94 2 play 27648 11/26 22:59:57.07 2 play 28672 11/26 22:59:57.20 2 play 29696 11/26 22:59:57.32 2 play 30720 11/26 22:59:57.45 2 play 31744 11/26 22:59:57.58 2 play 32768 11/26 22:59:57.71 2 play 33792 11/26 22:59:57.84 2 play 34816 11/26 22:59:57.96 2 play 35840 11/26 22:59:58.09 2 play 36864 11/26 22:59:58.22 2 play 37888 11/26 22:59:58.35 2 play 38912 11/26 22:59:58.48 2 play 39614 11/26 22:59:58.48 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 22:59:58.49 1 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:58.49 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:58.50 1 await ring 11/26 22:59:58.50 1 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:58.50 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:58.50 0 await ring 11/26 22:59:58.50 0 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:58.50 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 22:59:58.50 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 22:59:58.51 3 await ring 11/26 22:59:58.51 3 cmd: 0x1f NUM_RINGS 0x1 11/26 22:59:58.51 3 cmd: 0x27 ENA_RING_DET 0x0 11/26 23:00:01.57 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel.

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-23:00:01.586 00000008 Line 3: Playing completed	The Unity Bridge service acknowledges that the playing of the voice message has finished.
SFLOG 1396 1700 2002/11/26-23:00:01.586 00000100 Line 3: Playing #	The Unity Bridge service sends a request to the TR114 to play the audio of the DTMF digit "#." Playing of this digit indicates to the Octel server the end of the voice message.
11/26 23:00:01.58 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 23:00:01.58 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 23:00:01.59 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "#" and the associated parameters, and begins to play it to the Octel.
11/26 23:00:01.59 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 23:00:01.62 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 23:00:01.75 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 finishes playing the audio to the Octel.
11/26 23:00:01.75 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 23:00:02.93 2 intr: 0xb TONE_DETECT 0x88 11/26 23:00:03.00 2 intr: 0xb TONE_DETECT 0x8	The TR114 detects an incoming DTMF digit from the Octel. By using the conversion table provided earlier in this example, we can see that the TONE_DETECT event corresponds to DTMF digit "8."
SFLOG 1396 1700 2002/11/26-23:00:04.009 00000100 Line 3: Received 8	The "8" detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log.
SFLOG 1396 1700 2002/11/26-23:00:04.009 00000100 Line 3: Playing 9	The Unity Bridge service responds to the received "8" by sending a request to the TR114 to play audio of the DTMF digit "9."
11/26 23:00:04.00 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 23:00:04.00 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 23:00:04.01 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "9" and the associated parameters, and begins to play it to the Octel.
11/26 23:00:04.01 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 23:00:04.04 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 23:00:04.17 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 completes playing the audio to the Octel.
11/26 23:00:04.17 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 23:00:05.34 2 intr: 0xb TONE_DETECT 0x82 11/26 23:00:05.41 2 intr: 0xb TONE_DETECT 0x2 11/26 23:00:05.48 2 intr: 0xb TONE_DETECT 0x83 11/26 23:00:05.55 2 intr: 0xb TONE_DETECT 0x3 11/26 23:00:05.63 2 intr: 0xb TONE_DETECT 0x87 11/26 23:00:05.70 2 intr: 0xb TONE_DETECT 0x7 11/26 23:00:05.77 2 intr: 0xb TONE_DETECT 0x86 11/26 23:00:05.84 2 intr: 0xb TONE_DETECT 0x6 11/26 23:00:05.91 2 intr: 0xb TONE_DETECT 0x83 11/26 23:00:05.98 2 intr: 0xb TONE_DETECT 0x3 11/26 23:00:06.04 2 intr: 0xb TONE_DETECT 0x8d 11/26 23:00:06.11 2 intr: 0xb TONE_DETECT 0xd	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can see that the TONE_DETECT events correspond to the following DTMF digits: 2 3 7 6 3 A

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-23:00:07.124 00000100 Line 3: Received 23763A SFLOG 1396 1700 2002/11/26-23:00:07.124 00000008 Line 3: Message Delivered SFLOG 1396 1700 2002/11/26-23:00:07.124 00000008 Line 3: Completed delivering Messages	The "23763A" detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log. From the received packet, the Unity Bridge service is able to determine that Octel node #80200 has successfully received and processed the message. The Bridge has no more messages in the queue to deliver to this node.
SFLOG 1396 1700 2002/11/26-23:00:07.124 00000100 Line 3: Playing 045AC0BA9*84#D002A991C367C8ABDB0C2A4	The Unity Bridge service sends a request to the TR114 to play audio of the DTMF digit string "045AC0BA9*84#D002A991C367C8ABDB0C2A4."
11/26 23:00:07.11 2 cmd: 0x24 ENABLE_TONE_DET 0x0 11/26 23:00:07.11 2 cmd: 0x5e SPEECH_PARAMS 0x0 11/26 23:00:07.13 2 cmd: 0x5c SPEECH_START 0x0	The TR114 receives the audio of "045AC0BA9*84#D002A991C367C8ABDB0C2A4" and the associated parameters, and begins to play them to the Octel.
11/26 23:00:07.16 2 intr: 0x2c ECHO_ADAPT 0xa 11/26 23:00:08.51 1 cmd: 0x26 DIS_RING_DET 0x0 11/26 23:00:08.51 0 cmd: 0x26 DIS_RING_DET 0x0 11/26 23:00:08.52 1 await ring 11/26 23:00:08.52 1 cmd: 0x1f NUM_RINGS 0x1 11/26 23:00:08.52 1 cmd: 0x27 ENA_RING_DET 0x0 11/26 23:00:08.52 0 await ring 11/26 23:00:08.52 0 cmd: 0x1f NUM_RINGS 0x1 11/26 23:00:08.52 0 cmd: 0x27 ENA_RING_DET 0x0 11/26 23:00:08.52 3 cmd: 0x26 DIS_RING_DET 0x0 11/26 23:00:08.53 3 await ring 11/26 23:00:08.53 3 cmd: 0x1f NUM_RINGS 0x1 11/26 23:00:08.53 3 cmd: 0x27 ENA_RING_DET 0x0 11/26 23:00:09.10 2 cmd-sync: 0x5d SPEECH_ALTER 0x8 11/26 23:00:12.20 2 intr: 0x38 SPEECH_DONE 0x0	The TR114 finishes playing the audio to the Octel.
11/26 23:00:12.20 2 cmd: 0x24 ENABLE_TONE_DET 0x3 11/26 23:00:13.39 2 intr: 0xb TONE_DETECT 0x81 11/26 23:00:13.46 2 intr: 0xb TONE_DETECT 0x1 11/26 23:00:13.54 2 intr: 0xb TONE_DETECT 0x85 11/26 23:00:13.61 2 intr: 0xb TONE_DETECT 0x5 11/26 23:00:13.68 2 intr: 0xb TONE_DETECT 0x8e 11/26 23:00:13.75 2 intr: 0xb TONE_DETECT 0xe 11/26 23:00:13.82 2 intr: 0xb TONE_DETECT 0x8a 11/26 23:00:13.89 2 intr: 0xb TONE_DETECT 0xa 11/26 23:00:13.95 2 intr: 0xb TONE_DETECT 0x89 11/26 23:00:14.02 2 intr: 0xb TONE_DETECT 0x9 11/26 23:00:14.09 2 intr: 0xb TONE_DETECT 0x8e 11/26 23:00:14.16 2 intr: 0xb TONE_DETECT 0xe	The TR114 detects incoming DTMF digits from the Octel. By using the conversion table provided earlier in this example, we can see that the TONE_DETECT events correspond to the following DTMF digits: 1 5 B 0 9 B

Table 7-1 Brooktrout and SFLOG Example Trace (continued)

Trace	Description
SFLOG 1396 1700 2002/11/26-23:00:15.176 00000100 Line 3: Received 15B09B	The "15B09B" detected by the TR114 is passed to the Unity Bridge service and reported as received in sflog.log.
SFLOG 1396 1700 2002/11/26-23:00:16.918 00000008 Line 3: Call Out Completed.	The Unity Bridge service is done. It sends a request to the TR114 to hang up the call.

The TR114 begins the process of reinitializing the channel to be ready for the next incoming or outgoing call.

Using the Call Traces to Troubleshoot Why Messages Are Not Delivered from the Bridge to the Octels

We recommend that you use the Bridge Analog Network And Node Analyzer (BANANA) to troubleshoot analog problems between the Bridge and the Octels rather than using the call trace information in this section. We recommend use of BANANA, as it saves you from having to dig through the call traces to determine why messages were not delivered from the Bridge to the Octels.

However, in cases where it is necessary to evaluate the call traces rather than using BANANA, you will find in this section all of the details you need for using call traces to track down message delivery problems between the Bridge and the Octels.

- Are Calls Attempted?, page 7-14
- Are Dialouts Successful?, page 7-16
- Are Handshakes Successful?, page 7-17
- Are Any Messages Transmitted Successfully?, page 7-18

Are Calls Attempted?

To Determine Whether Calls Are Attempted by Viewing the Call Traces

Step 1 On the Bridge server, browse to the **Bridge\Starfish\Log** directory.

Step 2 Open the log file for the time period in question.

Step 3 Look for messages with the syntax **Line <Line number>: Call Out Process Initiated for Node <Node#> Window Type <WindowType>**, where:

- Line <Line number> is the line on which the call was attempted
- <Node#> is the destination Octel node serial number
- <WindowType> is 0, 1, or 2
  - 0=normal priority voice messages
  - 1=urgent priority voice messages
  - 2=administrative (text/voice name retrieval) calls

In Bridge 3.0(1) and later, these messages appear only when the Bridge is about to initiate a call to an Octel server.

Step 4 Look for the next line that has the same "Line <Line number>" notation you found in Step 3. The appearance of the "Line <Line number>" notation again, following the first appearance of the message, indicates whether the call is answered successfully.

Step 5 If lines similar to those described in Step 3 and Step 4 do appear in the log for the expected destination serial number and message type, an analog call is being attempted to the specified node. In this case, go to the "To Determine Whether Dialouts Are Successful by Viewing the Call Traces" procedure on page 7-16.

Step 6 If no lines similar to those described in Step 3 and Step 4 appear in the log, or if the only lines present in the log indicate "No Callout Activity was started," calls are not being attempted. In this case, continue with the following "Troubleshooting Why Calls Are Not Attempted" section.

Troubleshooting Why Calls Are Not Attempted

- Is the Octel node delivery schedule active?—In the Bridge Administrator, go to the Octel Node configuration page for each node. Confirm that the settings in the Message Delivery Windows section of the page indicate that the delivery schedule is active.
- Is the Unity Bridge service running?—On the Bridge server, open the Services Control Panel and confirm that the Unity Bridge service is running.
- Are any lines enabled?—In the Bridge Administrator, go to the Line Status page to view the status for each line.
- Is only one line enabled?—In the Bridge Administrator, go to the Line Status page to view the status for each line. The Bridge will not dial out when only one line is enabled.
- Are all ports busy with incoming calls?—In the Bridge Administrator, go to the Line Status page to view the status for each line.
- Is there a problem with the Bridge analog card(s) or drivers?—On the Bridge server, open the Windows Event Viewer Application log, and look for warnings and errors related to the cards and drivers.
- Are lines retired?—In the Bridge Administrator, go to the Line Status page to view the status for each line. On the Bridge server, you can also open the Windows Event Viewer Application log, and look for warnings and errors related to retired lines (for example, "Retired for callouts"). If line retirements occur, plug an analog phone into the lines going to the Bridge. Confirm that you get dial tone when you go off hook.

  When a problem occurs that prevents the Bridge from initiating an outgoing analog call on a particular analog port—for example, a line cord is not plugged in or there is no dial tone from the phone system—and when the same problem occurs on the same port four times in succession, the Bridge will retire that port and log the following warning in the Windows Event Viewer Application log: "Line <CmdArg>X<NoCmdArg>: Retired for callouts." This port will then be unavailable for outgoing calls. However, if the same port receives an incoming call and the connection is successful, the port will be put back into service for both incoming and outgoing calls, and another warning will appear in the Application Event Viewer: "Line <CmdArg>X<NoCmdArg>: Callouts re-started." This allows the Bridge to resolve the situation automatically if the condition clears up, or at the minimum allows the port to continue to receive incoming calls even if the problem initiating outgoing calls persists.

If all enabled analog lines on the Bridge server become retired due to these conditions, another warning will appear in the Application Event Viewer: "No lines are available for placing outgoing callouts." As soon as at least one port receives an incoming call and becomes available, another warning will appear in the log: "Line(s) are once again available for outgoing calls."

If these warnings appear frequently in the Application Event Viewer log, the analog lines connected to the Bridge server should be checked to see what problems may be occurring. After resolving any issues with the lines, any ports currently retired can be returned to service either by calling into the retired ports to trigger an automatic return to service, or by restarting the Unity Bridge service from the Services Control Panel.

Are Dialouts Successful?

To Determine Whether Dialouts Are Successful by Viewing the Call Traces

Step 1 Look at the same **Call Out Process Initiated for Node <Node#> Window Type <WindowType>** lines in the sflog file that you opened in the "To Determine Whether Calls Are Attempted by Viewing the Call Traces" procedure on page 7-14. If the call is attempted, this line will be followed by another line indicating the status of the call. There are four possibilities:

- Call Status=Answer—The dialout was successful, and the call was successfully answered by the destination Octel node.
- Call Status=Ring No Answer—The dialout was successful, but the destination Octel node did not answer.
- Call Status=Busy—The dialout was successful, but the destination Octel node was busy.
- Call Status=Line Error—The dialout was not successful. The Bridge encountered an error with the analog port or line prior to successfully dialing the phone number.

Step 2 If the Call Status is "Answer," go to the "To Determine Whether Handshakes Are Successful by Viewing the Call Traces" procedure on page 7-17.

If the Call Status is not "Answer," continue with the following "Troubleshooting Why Dialouts Are Not Successful" section.

Troubleshooting Why Dialouts Are Not Successful

- Is the correct phone number defined?—In the Bridge Administrator, go to the Octel Node configuration page for each node. Confirm that the Phone Number, Extension, and Dial Sequence fields contain correct information.
- Are long distance calls blocked?—If the call to the Octel node is long distance, confirm that long distance calls are not blocked by the phone system on the lines the Bridge is using.
- Does Octel answer when you dial the Octel node manually?—Plug an analog phone into the lines going to the Bridge, and dial the Octel node manually.

Are Handshakes Successful?

To Determine Whether Handshakes Are Successful by Viewing the Call Traces

Step 1 In the sflog file that you opened in the "To Determine Whether Calls Are Attempted by Viewing the Call Traces" procedure on page 7-14, look for the status line **Call Status=Answer**. This indicates that the dialout was successful, and that the call was successfully answered by the destination Octel node.

Step 2 Look for the next event for this line, which should be **Playing BD**. "Playing BD" indicates that the Bridge has detected the prompt on the Octel system and is now playing the analog DTMF digits B and D to indicate to the Octel that this is an Octel analog networking call.

Step 3 Following the Playing BD line, look for a line that contains **Received CDD**. By transmitting the CDD DTMF string, the Octel system indicates that it recognizes this call as an Octel analog networking call and is ready to receive the next DTMF packet.

If instead you see a line that contains **Received CC**, this indicates that the Octel node is using the VOICENET protocol, which is not supported. The Octel servers must be running Octel analog networking.

Step 4 Look for the line **Call Established From Serial # <Unity/BridgeSerial#> to Serial # <OctelSerial#>**, which should appear after two more packets are exchanged in each direction. The presence of this line indicates that the handshake was successful.

Note It is not uncommon to occasionally see the Playing BD line two or three times before the Octel system responds with CDD.

Step 5 If the handshake was successful, go to the "To Determine Whether Any Messages Are Transmitted Successfully by Viewing the Call Traces" procedure on page 7-18.

If the handshake was not successful, continue with the following "Troubleshooting Why Handshakes Are Not Successful" section.

Troubleshooting Why Handshakes Are Not Successful

- Is the destination Octel node serial number correct?—In the Bridge Administrator, go to the applicable Octel Node configuration page. Confirm that the number in the Serial Number field matches the serial number for the destination Octel node.
- Is the serial number for the Unity node correct?—In the Bridge Administrator, go to the Unity Node configuration page. Confirm that the number in the Serial Number field matches the serial number entered in the Octel system. The serial number must be entered on each Octel node.
- Is the correct phone number defined?—In the Bridge Administrator, go to the Octel Node configuration page for each node. Confirm that the Phone Number, Extension, and Dial Sequence fields contain correct information. If the Bridge is not receiving a response to the handshake, a possible cause is that whoever or whatever answered is not an Octel server.

Are Any Messages Transmitted Successfully?

To Determine Whether Any Messages Are Transmitted Successfully by Viewing the Call Traces

Step 1 In the sflog file that you opened in the "To Determine Whether Calls Are Attempted by Viewing the Call Traces" procedure on page 7-14, look for **Message Delivered**. This line indicates that a message was successfully delivered to the Octel system.

There can be multiple messages delivered on the same call, so each Message Delivered line indicates the successful delivery of the message identified on the Sending Message line that precedes the Message Delivered line.

Following is an excerpt from an sflog of a successfully delivered message:

```
...
Line 11: Sending Message from mailbox 60032@NYUnityServer5 to mailbox
70044@BridgeServer1.paris.cisco.com
Line 11: Playing Voice
Line 11: Playing D:\Bridge\Starfish\In\80007\86364\935105e7-c584-4167-a0ce-d5b42def44fe
Line 11: Playing completed
Line 11: Playing #
Line 11: Received 8
Line 11: Playing 9
Line 11: Received 13D*C8
Line 11: Message Delivered
...
```

The excerpt shows that the message was successfully delivered to the Octel system. If the call abnormally terminates subsequent to these events, it will not affect the delivery of this particular message, and the delivery for this message will not be retried.

Step 2 If messages are transmitted successfully to the Octel node, but are not delivered to the Octel subscriber, troubleshoot the problem on the Octel node.

If messages are not transmitted successfully, continue with the following "Troubleshooting Why Messages Are Not Transmitted Successfully" section.

Troubleshooting Why Messages Are Not Transmitted Successfully

- Does the target mailbox ID of the message correspond to a valid user mailbox on the Octel system?—In the sflog file, confirm that the target mailbox ID on the Sending Message line corresponds to a valid mailbox number on the Octel node.
- Were there line problems?—In the sflog file, search for the phrase "Encountered communication problems with this node." This phrase can indicate poor line quality or DTMF protocol miscommunications. Confirm the quality of the line. Use an analog digit grabber to monitor the DTMF digit durations and inter-digit delays—they should be between 60ms and 120ms.
- Are the messages in the correct format?—In the sflog file, confirm that there is no line indicating the message was in an unsupported format. The Bridge can play messages sent from Cisco Unity formatted with either the G.711 or G.729a codec.

Using the Call Traces to Troubleshoot Why Messages Are Not Delivered from the Octel Node to the Bridge Server

We recommend that you use the Bridge Analog Network And Node Analyzer (BANANA) to troubleshoot analog problems between the Octels and the Bridge rather than using the call trace information in this section. We recommend use of BANANA, as it saves you from having to dig through the call traces to determine why messages were not delivered from the Bridge to the Octels.

However, in cases where it is necessary to evaluate the call traces rather than using BANANA, you will find in this section the information that you need for using call traces to track down message delivery problems between the Octels and the Bridge.

- The Windows Event Viewer on the Bridge server should be the first place to look when troubleshooting voice message flow from an Octel node to a Cisco Unity node. The Bridge services record errors and warnings to the Windows Event Viewer application log, and you can troubleshoot any errors or warnings you find in the Event logs.
- The call log traces can be used to obtain information about messages coming from Octel servers through the Bridge voice-fax card(s). The log records actions that the Bridge service attempts, notes whether those actions are completed successfully, and records the reasons for failed actions. Within the log directory are files named SFLOG.mmddtttt.LOG. Each file contains log entries for one hour of the day; the filename indicates which hour.
- Go to the Bridge\Starfish\Log directory and open the SFLOG.log that corresponds to the time the message was sent. Look for the following lines in the log file. The presence of these lines shows a successful transmission from the Octel node to the Bridge server.

  ```
  SFLOG 1888  664 2002/09/24-17:02:05.968 00000100
  Line 7: Received 252B605C794162D39B1C328D83CA153A79C4
  SFLOG 1888  664 2002/09/24-17:02:05.968 00000008
  Line 7: New Message from mailbox 10006 to mailbox 30022
  SFLOG 1888  664 2002/09/24-17:02:05.968 00000100
  Line 7: Playing 021#31
  SFLOG 1888  664 2002/09/24-17:02:06.890 00000008
  Line 7: Recording Voice
  SFLOG 1888  664 2002/09/24-17:02:06.890 00000100
  Line 7: Recording c:\Bridge\Starfish\Out\6c37c321-533e-45fc-bdf7-1676c6ed3046
  SFLOG 1888  664 2002/09/24-17:03:07.859 00000008
  Line 7: Recording completed
  SFLOG 1888  664 2002/09/24-17:03:08.859 00000100
  Line 7: Received #
  SFLOG 1888  664 2002/09/24-17:03:08.859 00000100
  Line 7: Playing 8
  SFLOG 1888  664 2002/09/24-17:03:11.296 00000100
  Line 7: Received 9
  SFLOG 1888  664 2002/09/24-17:03:11.296 00000100
  Line 7: Playing 0282C6
  SFLOG 1888  664 2002/09/24-17:03:12.203 00000008
  Line 7: Message Saved
  ```

  If you do not see the above lines in the log, troubleshoot any errors or discrepancies that may be displayed in the SFLOG. Verify that your phone lines are plugged in and that the Octel node and Cisco Unity node serial numbers and node IDs are configured correctly.

If the calling Octel server does not have the Cisco Unity node serial number defined in its node configuration, the Bridge hangs up immediately when it receives a call from the Octel node. In the following example, the SFLOGs show that the calling Octel sent an initial handshake packet of "CD" instead of "BD":

```
SFLOG 1732  704 2003/01/10-22:04:49.711 00000008
Line 1: Call Received.
SFLOG 1732  704 2003/01/10-22:04:50.713 00000100
Line 1: Playing 1.sph
SFLOG 1732  704 2003/01/10-22:04:56.531 00000100
Line 1: Received CD
SFLOG 1732  704 2003/01/10-22:04:57.533 00000100
Line 1: Received
SFLOG 1732  704 2003/01/10-22:04:58.083 00000008
Line 1: Incoming Call Completed
```

Additionally, the Bridge logs the following warning to the Windows Event Log:

```
Event Type: Warning
Event Source: Bridge
Event Category: None
Event ID: 108
Bridge received an incoming call that could not be processed. The calling server does not
have a Serial Number defined in its Bridge node profile.
Verify that all remote servers configured to communicate with Bridge have Serial Numbers
for all Bridge nodes.
```

Because the Bridge requires the Cisco Unity node serial number to be configured on the Octel server, you will have to define the serial number for the Cisco Unity node in the node profile on the Octel server. When the serial number for the Cisco Unity node is properly configured in the node profile on the Octel server, the Octel system will send the expected "BD" handshake packet which the Bridge will successfully respond to with "CDD," and the call will proceed as in the following example:

```
SFLOG  736  752 2002/12/31-21:41:45.017 00000008
Line 3: Call Received.
SFLOG  736  752 2002/12/31-21:41:46.019 00000100
Line 3: Playing 1.sph
SFLOG  736  752 2002/12/31-21:41:51.436 00000100
Line 3: Received
SFLOG  736  752 2002/12/31-21:41:51.436 00000100
Line 3: Playing 1.sph
SFLOG  736  752 2002/12/31-21:41:56.974 00000100
Line 3: Received BD
SFLOG  736  752 2002/12/31-21:41:56.974 00000100
Line 3: Playing CDD
SFLOG  736  752 2002/12/31-21:42:02.052 00000100
Line 3: Received 12##C#5D82AC6AA897
SFLOG  736  752 2002/12/31-21:42:02.062 00000100
Line 3: Protocol Level = 3
SFLOG  736  752 2002/12/31-21:42:02.062 00000100
Line 3: Playing 012444C54331CA
```

If the Message Was Received by the Bridge

- If the SFLOGs indicate that the handshake was successful, and that the Bridge received the message, a copy of the message should be saved to the Bridge\VPIM\Internet\Out and Bridge\VPIM\Internet\Out\Tmp directories.

Note The message will stay in the Bridge\VPIM\Internet\Out\Tmp directory only for the number of days that is set in the Retention Days for Temporary SMTP Messages setting. The message will stay in the Bridge\VPIM\Internet\Out directory until it is delivered to the Voice Connector.

- Verify that the voice message appears in the Bridge\VPIM\Internet\Out\Tmp directory.

BANANA admin Call States, and Reasons for Communication Errors

When the Bridge receives or makes a call, the call goes through various states. If a call fails for some reason, BANANA admin displays the state the call was in when it failed. Table 7-2 lists the possible call state codes and text that may be displayed in BANANA admin. BANANA admin also displays a reason the call failed. Table 7-3 lists the reason codes and text for communication errors. If a call results in an error, BANANA admin assigns an error code by combining the call state and reason codes.

Table 7-2 Call State Code and Text Displayed in BANANA admin

Call State Code	Call State Text
0	Complete
1	Dialing
2	Wake-up
3	Wake-up Response
4	Line Sync
5	Line Sync Response
6	Session Header
7	Session Header Response
8	Message Header
9	Message Header Response
10	Additional Recipient Header
11	Additional Recipient Header Response
12	Text Name Confirmation
13	Audio Transmission
14	Audio Terminator
15	Fax Request
16	Fax Transmission
17	Save Request

Table 7-2 Call State Code and Text Displayed in BANANA admin (continued)

Call State Code	Call State Text
18	Save Response
19	Message Response
20	Admin Request
21	Admin Mailbox Response
22	Admin Response
23	Admin Recording Recd Confirmation
24	Text Name Response
25	Text Name Transmission
26	Recording
27	Recording Voice Name
28	Playing
29	Add Recipients
30	Call Out Process Initiated
31	Failure After Outbound Message Rejection
32	Request For Voice Name
998	Other
999	Unknown

Table 7-3 BANANA admin Reason Code and Reason Text for Communication Errors

Reason Code	Reason Text
0	Successful
1	Line Error
2	Line State Unknown
3	Busy
4	Ring No Answer
5	Expected Data Not Received
6	Received DTMF String Longer Than Expected
7	Received DTMF String Shorter Than Expected
8	Received Data Invalid
9	Silence Timeout
10	No End-of-Fax Received
11	Bridge Rejected Admin Name Push
12	Calling Ser# Not Configured as Octel Node
13	Target Unity Node Not Configured; or Invalid DTMFs
998	Other

Table 7-3 BANANA admin Reason Code and Reason Text for Communication Errors (continued)

Reason Code	Reason Text
999	Unknown

Bridge Compensates for Echoed Digits in Wakeup Packets

When viewing the starfish logs on the Unity Bridge server, or using BANANA to view call detail, the wakeup (BD) and wakeup response (CDD) packets contain additional B, C, and/or D digits however the call may proceed normally to completion or to some other point in the protocol.

Echoed digits that occasionally appear in the log do not indicate a problem. However, if echo is consistently a problem during transmissions and calls consistently fail later in the protocol despite the successful completion of the wakeup portion, the gateway settings and other environmental conditions should be investigated to resolve any underlying problems.

When an inbound call is received by the Bridge, the Bridge requires the first inbound DTMF sequence to be the Octel analog networking wakeup sequence of 'BD.' Exceptions to this are as follows:

1. If 'B' is received, without receiving the 'D' within the configured Inbound DTMF - Inter-Digit Timeout, the Bridge will return to a state waiting for 'BD.'
2. If 'BD' is any part of the sequence received, the entire sequence is accepted as the wakeup. In regular expression terms, the acceptable wakeup sequence would be represented as .*BD.*

Any sequence other than 1 and 2 above received while awaiting the wakeup sequence after answering an inbound call will be considered invalid and the call will end at this point. This includes a sequence of 'CD,' which is a valid wakeup sequence in the Octel analog networking protocol, but not supported by the Bridge (see CSCea35313).

When the Bridge places an outbound call, upon detection of a connected state, the Bridge sends the 'BD' wakeup sequence to the remote system. The Bridge requires the response from the remote system to be the Octel analog networking wakeup response of 'CDD.' Exceptions to this are as follows:

1. If 'CDD' is any part of the response sequence received, the entire sequence is accepted as the valid wakeup response. In regular expression terms, the acceptable wakeup sequence would be represented as .*CDD.*
2. If the response received begins with 'B,' but does not include 'CDD,' the Bridge will return to a state waiting for 'CDD.' This is to account for any echo conditions that may occur. The regular expression representation of sequences falling into this category is B.* (Note that anything matching the regular expression B.*CDD.* would fall into category a and be accepted immediately).

Any response sequence other than 1 and 2 above received while awaiting the wakeup response sequence after sending the 'BD' wakeup sequence will be considered invalid and the call will end at this point.