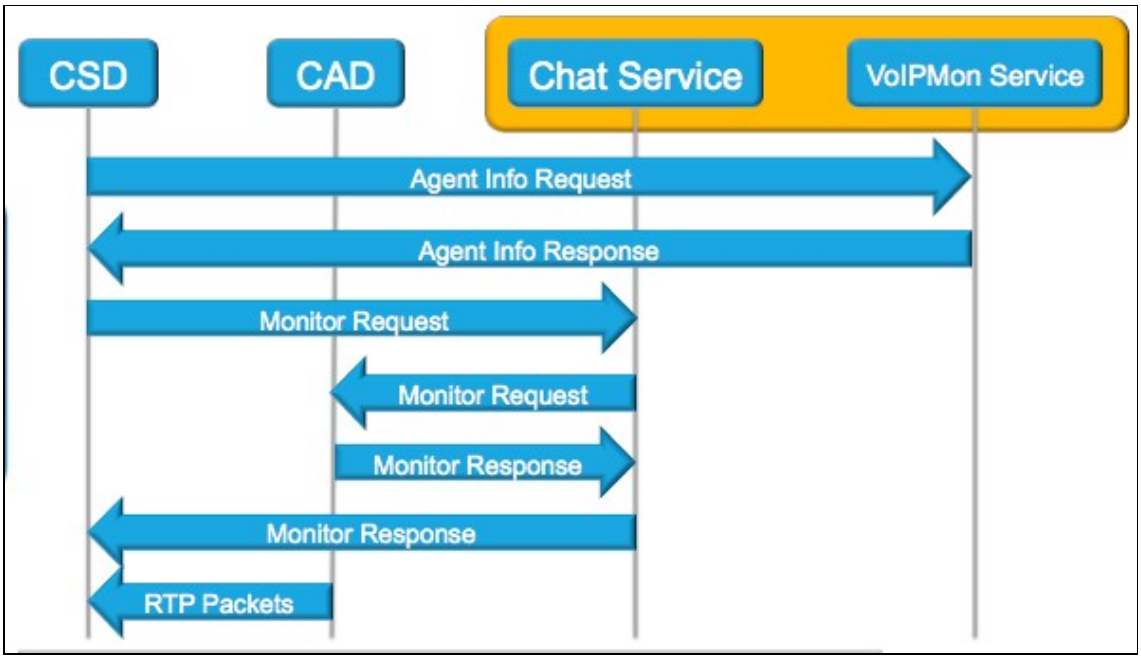
Contents

CAD Desktop Monitoring Log Analysis

Purpose

This document serves as a sample log analysis from a working UCCX CAD Desktop Monitoring Session. The goal is to increase knowledge in the field of troubleshooting CAD Desktop Monitoring by providing good, working examples to compare to production environments.

Communication Flow Diagram



Log Information

When tracing communication through CAD Desktop Monitoring logs, use the following logs. In most cases DEBUG level tracing is sufficient except where specified in the following log snippets. When using TRACE level settings, be sure to consult Cisco TAC. TRACE level may have a negative impact on performance and should not be enable during high production hours.

When retrieving CCX Service logs, collect them from the service's active side.

- Agent.dbg
  - C:\Program Files\Cisco\Desktop\log on Agent?s PC

- Supervisor.dbg
    - ♦ C:\Program Files\Cisco\Desktop\log on Supervisor?s PC
- FCCServer.dbg
    - ♦ Collected via RTMT (Call/Chat Service)

Keywords

When tracing communication through CAD Desktop Monitoring logs, the following keywords serve as unique identifiers for transactions.

- Agent Extension
- ConnectionCallID
- Message Type
- IP Addresses

Log Communication Analysis

Start with the **`CALL_ESTABLISHED_EVENT`** in the Agent.dbg log to find the **`ConnectionCallID`**. Ensure that **`IsBusinessCall`** is **`true`** otherwise the agent is not considered applicable for Desktop Monitoring.

`Agent.dbg`

```
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2141] GetDebugInfo:
 GetDebugInfo ---------------------- Begin: CALL_ESTABLISHED_EVENT ----------------------
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2263] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2264] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2265] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2266] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2267] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2268] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2269] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2270] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2271] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2272] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2273] GetDebugInfo: CALL_ESTABLISHED_EVENT:
2012-03-24 13:41:16:458 DEBUG [0x8fc] EventHandler.cpp[2458] GetDebugInfo:
 GetDebugInfo ---------------------- End: CALL_ESTABLISHED_EVENT ----------------------
```

The following entry will be printed repeatedly until both **`address1`** and **`address2`** fields are filled in, indicating the IP addresses of the incoming and outgoing RTP streams. Note that **`callCanBeMonitored`** will return **`false`** until such IP addresses are obtained.

`Agent.dbg`

```
2012-03-24 13:41:16:459 DEBUG [0x8fc] AgentMonitor.cpp[539] AgentMonitor::callCanBeMonitored:
 isTheActiveCall false, isABusinessCall true, forwardOnlyBusinessCalls true, address1 [], address2
2012-03-24 13:41:16:459 TRACE [0x8fc] AgentMonitor.cpp[547] AgentMonitor::callCanBeMonitored:
 Call can't be monitored because it is not the active call
2012-03-24 13:41:16:459 DEBUG [0x8fc] AgentMonitor.cpp[556] AgentMonitor::callCanBeMonitored: Retu
```

Look for the **`OnRTPStartedEvent`** for both the out going (**`RTP Direction OUTPUT`**) and in coming (**`RTP Direction INPUT`**) streams. CAD uses this events to determine the IP addresses needed to capture RTP traffic.

`Agent.dbg`

```
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1977] OnRtpStartEvent: ========== OnRTPStar
```

```
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1978] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1982] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1983] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1984] OnRtpStartEvent: OnRTPStartedEvent:
 Direction of Audio Stream: RTP Direction OUTPUT
2012-03-24 13:41:16:583 DEBUG [0x8fc] EventHandler.cpp[1991] OnRtpStartEvent: ========== End OnRTP
```

Agent.dbg

```
2012-03-24 13:41:16:589 DEBUG [0x8fc] EventHandler.cpp[1977] OnRtpStartEvent: ========== OnRTPStar
2012-03-24 13:41:16:589 DEBUG [0x8fc] EventHandler.cpp[1978] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:589 DEBUG [0x8fc] EventHandler.cpp[1982] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:589 DEBUG [0x8fc] EventHandler.cpp[1983] OnRtpStartEvent: OnRTPStartedEvent: C
2012-03-24 13:41:16:589 DEBUG [0x8fc] EventHandler.cpp[1984] OnRtpStartEvent: OnRTPStartedEvent:
 Direction of Audio Stream: RTP Direction INPUT
2012-03-24 13:41:16:590 DEBUG [0x8fc] EventHandler.cpp[1991] OnRtpStartEvent: ========== End OnRTP
```

Having both IP addresses, CAD will report **`callCanBeMonitored`** as **`true`**.

Agent.dbg

```
2012-03-24 13:41:16:591 DEBUG [0x8fc] AgentMonitor.cpp[539] AgentMonitor::callCanBeMonitored:
 isTheActiveCall true, isABusinessCall true, forwardOnlyBusinessCalls true, address1 [10.77.85.104
2012-03-24 13:41:16:591 DEBUG [0x8fc] AgentMonitor.cpp[556] AgentMonitor::callCanBeMonitored: Retu
```

In the case where the supervisor initiated CAD Desktop Monitoring through CSD, we will see an entry in the Supervisor.dbg log showing that the supervisor clicked the start monitor button.

Supervisor.dbg

```
2012-03-24 13:41:38:611 CALL [0x5d4] AgentView.cpp[3418] CAgentView::OnSupervisorStartVoicemonitor
 AV3418 User clicked START VOICE MONITOR button
2012-03-24 13:41:38:611 DEBUG [0x5d4] AgentView.cpp[2870] CAgentView::GetDeviceIDForSelectedAgent:
 AV2870 GetDeviceIDForSelectedAgent returned: 85104
```

CSD will attempt to open the supervisors PC audio device for the RTP stream playback.

Supervisor.dbg

```
2012-03-24 13:41:38:727 CALL [0x5d4] SplkPSoundChannel.cpp[205] SplkPSoundChannel::OpenDevice: Att
2012-03-24 13:41:38:727 CALL [0x5d4] SplkPSoundChannel.cpp[127] SplkPSoundChannel::OpenDefaultDevi
2012-03-24 13:41:38:746 TRACE [0x5d4] SplkPSoundChannel.cpp[133] SplkPSoundChannel::OpenDefaultDev
2012-03-24 13:41:38:746 TRACE [0x5d4] SplkPSoundChannel.cpp[134] SplkPSoundChannel::OpenDefaultDev
2012-03-24 13:41:38:746 CALL [0x5d4] SplkPSoundChannel.cpp[143] SplkPSoundChannel::OpenDefaultDevi
2012-03-24 13:41:38:746 DEBUG [0x5d4] OpenH323Wrapper.cpp[145] OpenSoundChannel: Sound Channel ope
2012-03-24 13:41:38:746 TRACE [0x5d4] FCVoIPMonClientMon.cpp[940] FCVoIPMonClient::StartMonitoring
```

CSD must query the VoIPMon Service for agent information. Here we see the UCCX servers port information being obtained for the query. This is the only entry indicating this action has started successfully.

Supervisor.dbg

```
2012-03-24 13:41:38:746 CALL [0x5d4] CFcvmsMonitorServerList.cpp[2268] CFcvmsMonitorServerList::ge
2012-03-24 13:41:38:746 CALL [0x5d4] CFcvmsMonitorServerList.cpp[2327] CFcvmsMonitorServerList::ge
```

CSD will print the response from the VoIPMon Service, which includes the agents IP address. Once the agents IP address is obtained, it will start the **`Voice Monitor Thread`**.

Supervisor.dbg

```
2012-03-24 13:41:38:746 DEBUG [0x5d4] FCVoIPMonClientMon.cpp[950] FCVoIPMonClient::StartMonitoring
 Successfully got the local address over the VPN: 10.77.85.103.
2012-03-24 13:41:38:746 DEBUG [0x5d4] FCVoIPMonClientMon.cpp[968] FCVoIPMonClient::StartMonitoring
 Attempting to start RTP Voice Monitor thread
2012-03-24 13:41:38:746 DEBUG [0x5d4] FCVoIPMonClientMon.cpp[971] FCVoIPMonClient::StartMonitoring
 127.0.0.1, m_ToAgentPort: 59010, m_FromAgentPort: 59012, m_JitterBuffer: 400, m_SoundBuffers: 30,
2012-03-24 13:41:38:746 DEBUG [0x5d4] FCVoIPMonClientMon.cpp[988] FCVoIPMonClient::StartMonitoring
```

CSD will create a **genericMessage** to send to CAD, through the Chat Service, to request CAD to start forwarding the RTP packets to CSD. The agents IP address is stored in the **dataStringMap[0]** field.

Supervisor.dbg

```
2012-03-24 13:41:38:770 DEBUG [0x5d4] MonitoringAndRecording.cpp[37] MonitoringAndRecording::start
2012-03-24 13:41:38:770 CALL [0x5d4] MonitoringAndRecording.cpp[100] MonitoringAndRecording::sendM
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1609] FCCClientAPI::genericMessage: Begin.
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1623] FCCClientAPI::genericMessage:
 Begin. source Ext: 1000, sourceType: Supervisor, destinationType: To Agent, destinationId: 85104,
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1631] FCCClientAPI::genericMessage:   dataS
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1642] FCCClientAPI::genericMessage:   dataI
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1642] FCCClientAPI::genericMessage:   dataI
2012-03-24 13:41:38:770 DEBUG [0x5d4] FCCClientAPI.cpp[1679] FCCClientAPI::genericMessage: Calling
2012-03-24 13:41:38:772 DEBUG [0x5d4] FCCClientAPI.cpp[1730] FCCClientAPI::genericMessage: End.  r
2012-03-24 13:41:38:772 CALL [0x5d4] MonitoringAndRecording.cpp[100] MonitoringAndRecording::sendM
2012-03-24 13:41:38:772 CALL [0x5d4] MonitoringAndRecording.cpp[40] MonitoringAndRecording::startM
 AV0040 Supervisor has started voice monitor on agent 85104
2012-03-24 13:41:38:772 DEBUG [0x5d4] MonitoringAndRecording.cpp[44] MonitoringAndRecording::start
```

When tracing **genericMessage** events, note the **sourceType**, **destinationType**, and **messageType**. These fields indicate the direction of the message and which messages match each other in the various logs. Next, the Chat Service will receive the message, as indicated by these three fields, to pass to CAD. The agents IP address is stored in the **stringArgs[0]** field.

FCCServer.dbg

```
2012-03-24 13:41:38:774 CALL [0x35e3ba0] FCC_ServerImpl.cpp[297] genericMessage: Begin.
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3179] genericMessage:
 Begin.  senderUserType: Supervisor, senderExtension: 1000, destType: To Agent, destinationId: 851
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3189] genericMessage: <Supervisor_1000>
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3220] genericMessage: Sender is a superv
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3248] genericMessage: find agent.
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3300] genericMessage: stringArgIndexes[0
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3312] genericMessage: stringArgs[0] = 10
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3323] genericMessage: ulongArgIndexes[0]
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3323] genericMessage: ulongArgIndexes[1]
2012-03-24 13:41:38:774 DEBUG [0x35e3ba0] CChatServer.cpp[3333] genericMessage: ulongArgs[0] = 590
2012-03-24 13:41:38:775 DEBUG [0x35e3ba0] CChatServer.cpp[3333] genericMessage: ulongArgs[1] = 590
2012-03-24 13:41:38:775 DEBUG [0x35e3ba0] CChatServer.cpp[3382] genericMessage: End.
```

Next CAD will receive this **genericMessage** from the Chat Service. This is matched up by the **sourceType**, **destinationType**, and **messageType**. The agents IP address is stored in the **enericMessage.dataStringMap[0]** field.

Agent.dbg

```
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[233] FCC_Client_impl::eventList: FCC_ELI
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[800] FCC_Client_impl::genericMessage:
 Begin. destType: Agent, destID: 85104, senderUserType: Supervisor, senderId: 1000, messageType: 2
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[804] FCC_Client_impl::genericMessage: st
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[809] FCC_Client_impl::genericMessage: ul
```

```
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[809] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:771 DEBUG [0xa04] FCC_Client_impl.cpp[814] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:772 DEBUG [0xa04] FCC_Client_impl.cpp[814] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:772 DEBUG [0xa04] FCC_Client_impl.cpp[843] FCC_Client_impl::genericMessage:
 genericMessage.dataStringMap[0] = 10.77.85.103.
2012-03-24 13:41:38:772 DEBUG [0xa04] FCC_Client_impl.cpp[851] FCC_Client_impl::genericMessage: ge
2012-03-24 13:41:38:772 DEBUG [0xa04] FCC_Client_impl.cpp[851] FCC_Client_impl::genericMessage: ge
```

Once CAD receives this message, it will immediately begin to start copying and forwarding packets to the supervisors CSD client. The following entry confirms CAD is starting the process.

Agent.dbg

```
2012-03-24 13:41:38:773 DEBUG [0x8fc] InterventionManager.cpp[359] CInterventionManager::startMoni
 IM0359 Start Monitoring for remote agent: 85104, Client Address 10.77.85.103, SupervisorID: 1000,
```

CAD will identify the two RTP streams, indicated by **`Stream ONE`** and **`Stream TWO`**, to begin to capture the RTP traffic. The IP addresses were obtained from the **`OnRTPStartedEvent`** mentioned previously. The Agent.dbg log will print the **`ConnectionCallID`** of the active call for the agent.

Agent.dbg

```
2012-03-24 13:41:38:774 TRACE [0x8fc] MonitoringBase.cpp[166] MonitoringBase::startMonitoringOrRec
 Stream ONE: AgentExt<85104> callid<59010>, port<24578>, action<1738005680>, IPAddr<10.77.85.104>
2012-03-24 13:41:38:774 TRACE [0x8fc] MonitoringBase.cpp[167] MonitoringBase::startMonitoringOrRec
 Stream TWO: AgentExt<85104> callid<59010>, port<24578>, action<1179648>, IPAddr<10.77.85.105>
2012-03-24 13:41:38:776 DEBUG [0x8fc] AgentMonitor.cpp[761] AgentMonitor::findActiveCall: The acti
```

Next CAD begins to open the NIC identified for CAD Desktop Monitoring in postinstall.exe. The logs will display which NIC adapter.

Agent.dbg

```
2012-03-24 13:41:38:776 TRACE [0xad4] CFCDMSnifferSession.cpp[215] CFCDMSnifferSession::_snifferSe
 Calling splk_pcap_open_live() passing adapter [\Device\Splkpc_{671D2F1B-F54F-431B-8603-B17E8973C4
```

CAD will send a **`genericMessage`** to CSD indicating that it will being to capture and forward packets.

Agent.dbg

```
2012-03-24 13:41:38:777 DEBUG [0x8fc] ChatAPI.cpp[380] CChatAPI::GenericMessage:
 Ext <85104>, Team <Default>, Name < agent2>, GenericMessage <GM_MONITOR_STATUS_NOTIFY>
2012-03-24 13:41:38:777 DEBUG [0x8fc] FCCClientAPI.cpp[1609] FCCClientAPI::genericMessage: Begin.
2012-03-24 13:41:38:777 DEBUG [0x8fc] FCCClientAPI.cpp[1623] FCCClientAPI::genericMessage:
 Begin. source Ext: 85104, sourceType: Agent, destinationType: To Supervisor, destinationId: 1000,
2012-03-24 13:41:38:777 DEBUG [0x8fc] FCCClientAPI.cpp[1642] FCCClientAPI::genericMessage:   dataI
2012-03-24 13:41:38:777 DEBUG [0x8fc] FCCClientAPI.cpp[1642] FCCClientAPI::genericMessage:   dataI
2012-03-24 13:41:38:777 DEBUG [0x8fc] FCCClientAPI.cpp[1679] FCCClientAPI::genericMessage: Calling
```

The Chat Service will relay this message from CAD to CSD.

FCCServer.dbg

```
2012-03-24 13:41:38:783 CALL [0x1e83ba0] FCC_ServerImpl.cpp[297] genericMessage: Begin.
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3179] genericMessage:
 Begin.  senderUserType: Agent, senderExtension: 85104, destType: To Supervisor, destinationId: 10
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3189] genericMessage: <AGENT_DESKTOP_age
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3206] genericMessage: Sender is an agent
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3263] genericMessage: find supervisor.
```

```
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3323] genericMessage: ulongArgIndexes[0]
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3323] genericMessage: ulongArgIndexes[1]
2012-03-24 13:41:38:783 DEBUG [0x1e83ba0] CChatServer.cpp[3333] genericMessage: ulongArgs[0] = 1.
2012-03-24 13:41:38:784 DEBUG [0x1e83ba0] CChatServer.cpp[3333] genericMessage: ulongArgs[1] = 0.
2012-03-24 13:41:38:784 DEBUG [0x1e83ba0] CChatServer.cpp[3382] genericMessage: End.
```

CSD receives the **genericMessage** from CAD via the Chat Service. At this point CSD will start looking for the RTP streams.

Supervisor.dbg

```
2012-03-24 13:41:38:781 DEBUG [0xe20] FCC_Client_impl.cpp[233] FCC_Client_impl::eventList: FCC_ELI
2012-03-24 13:41:38:781 DEBUG [0xe20] FCC_Client_impl.cpp[800] FCC_Client_impl::genericMessage:
 Begin. destType: Supervisor, destID: 1000, senderUserType: Agent, senderId: 85104, messageType: 2
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[809] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[809] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[814] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[814] FCC_Client_impl::genericMessage: ul
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[851] FCC_Client_impl::genericMessage: ge
2012-03-24 13:41:38:782 DEBUG [0xe20] FCC_Client_impl.cpp[851] FCC_Client_impl::genericMessage: ge
```

CAD begins to capture RTP traffic. The Agent.dbg log shows the NIC being used as well as the packet filter.

Agent.dbg

```
2012-03-24 13:41:38:781 INFO [0xad4] VOIP2042 _snifferSessionThread: The NIC adapter was opened su
2012-03-24 13:41:38:781 TRACE [0xad4] CFCDMSnifferSession.cpp[229] CFCDMSnifferSession::_snifferSe
 Calling splk_pcap_setmintocopy().
2012-03-24 13:41:38:781 TRACE [0xad4] CFCDMSnifferSession.cpp[232] CFCDMSnifferSession::_snifferSe
2012-03-24 13:41:38:781 TRACE [0xad4] SnifferSession.cpp[191] SnifferSession::PacketFilter_lookupr
 Calling splk_pcap_lookupnet() with \Device\Splkpc_{671D2F1B-F54F-431B-8603-B17E8973C423}.
2012-03-24 13:41:38:782 DEBUG [0xad4] SnifferSession.cpp[252] SnifferSession::PacketFilter_setFilt
 filter set to (udp and ((ip host 10.77.85.104 and port 24578) or (ip host 10.77.85.105 and port 2
2012-03-24 13:41:38:782 TRACE [0xad4] CFCDMSnifferSession.cpp[245] CFCDMSnifferSession::_snifferSe
 Capturing and forwarding packets.
```

With CAD tracing level set to TRACE, the Agent.dbg log will show every RTP packet being captured. The log will fill quickly, roughly 10MB in 45 seconds.

Agent.dbg

```
2012-03-24 13:41:38:782 TRACE [0xad4] filterPacket.cpp[67] FilterPacket::parsePacket: Begin.
2012-03-24 13:41:38:783 DUMP [0xad4] filterPacket.cpp[96] FilterPacket::parsePacket: hexPacket: 6E
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[106] FilterPacket::parsePacket: destMacAddr
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[115] FilterPacket::parsePacket: sourceMacAd
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[143] FilterPacket::parsePacket: ipHeaderLen
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[163] FilterPacket::parsePacket: sourceIpAdd
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[164] FilterPacket::parsePacket: sourceIpAdd
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[179] FilterPacket::parsePacket: destination
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[180] FilterPacket::parsePacket: destination
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[188] FilterPacket::parsePacket: source port
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[192] FilterPacket::parsePacket: destination
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[196] FilterPacket::parsePacket: UDP packet
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[203] FilterPacket::parsePacket: Bytes to se
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[216] FilterPacket::parsePacket: Payload typ
2012-03-24 13:41:38:783 TRACE [0xad4] filterPacket.cpp[67] FilterPacket::parsePacket: End.
```

If CAD is successful in sending the captured packet to CSD, the following entry will be present after the packet print out above. The syntax **forwardPacketToRecipients: send (outbound)** and

forwardPacketToRecipients: Outgoing: Sent 172 bytes indicates that CAD is sending an INBOUND stream RTP packet. Corresponding OUTBOUND stream RTP packets will have the syntax **forwardPacketToRecipients: send (inbound)** and **forwardPacketToRecipients: Incoming: Sent 172 bytes**.

Agent.dbg

```
2012-03-24 13:41:38:784 TRACE [0xad4] filterPacket.cpp[360] FilterPacket::forwardPacketToRecipient
 send (outbound) [10.77.85.105 24578] to [85104 [ destination [1000, 10.77.85.103, 59010,59012, fa
2012-03-24 13:41:38:784 TRACE [0xad4] filterPacket.cpp[361] FilterPacket::forwardPacketToRecipient
```

As mentioned, CSD started looking for the incoming RTP streams before CAD started sending the RTP streams. This entry shows the transition from CSD not receiving the RTP packets (**Both packets null sleep 10ms**) to CSD receiving them (**Payload size is 160** and **Payload type is 0:PCMU (G711u)**).

Supervisor.dbg

```
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[395] PlaybackThread::Main: Add codec Fram
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[227] PlaybackThread::Main: size1 = 0
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[271] PlaybackThread::Main: ReadBufferdata
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[281] PlaybackThread::Main: size2 = 0
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[288] PlaybackThread::Main:
 After assignment prev_rtpTimestamp2=rtpTimestamp2 rtpTimestamp2 = 17100
2012-03-24 13:41:39:785 TRACE [0xb7c] PlaybackThread.cpp[380] PlaybackThread::Main: Both packets n
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[395] PlaybackThread::Main: Add codec Fram
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[227] PlaybackThread::Main: size1 = 160
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[271] PlaybackThread::Main: ReadBufferdata
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[281] PlaybackThread::Main: size2 = 160
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[288] PlaybackThread::Main:
 After assignment prev_rtpTimestamp2=rtpTimestamp2 rtpTimestamp2 = 17200
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[292] PlaybackThread::Main: Getting timest
2012-03-24 13:41:39:791 DEBUG [0xb7c] PlaybackThread.cpp[403] PlaybackThread::Main: Started gettin
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[436] PlaybackThread::Main: Payload size i
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[445] PlaybackThread::Main: Payload type i
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[473] PlaybackThread::Main:
 Frame rate is 80, Bit per sample is 16, Bytes per frame is 80
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[227] PlaybackThread::Main: size1 = 160
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[271] PlaybackThread::Main: ReadBufferdata
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[281] PlaybackThread::Main: size2 = 160
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[288] PlaybackThread::Main:
 After assignment prev_rtpTimestamp2=rtpTimestamp2 rtpTimestamp2 = -1101501915
2012-03-24 13:41:39:791 TRACE [0xb7c] PlaybackThread.cpp[292] PlaybackThread::Main: Getting timest
```

Lastly, Supervisor.dbg will print an entry indicating that the supervisor stopped VoIP Monitoring via CSD.

Supervisor.dbg

```
2012-03-24 13:42:23:743 CALL [0x5d4] AgentView.cpp[3506] CAgentView::OnSupervisorStopVoicemonitor:
 AV3506 User clicked STOP VOICE MONITOR button
2012-03-24 13:42:23:822 CALL [0x5d4] MonitoringAndRecording.cpp[57] MonitoringAndRecording::stopMo
 AV0057 Supervisor has stopped voice monitor on agent 85104
```