Troubleshoot CVP Courtesy Callback (CCB) Gateway Capacity Validation Failure

Contents

Introduction
Prerequisites
Requirements
Components Used
Background Information
Symptoms
Troubleshoot
Solution
Permanent Solution
Test Final Solution

Introduction

This document describes how to troubleshoot a Customer Voice Portal (CVP) CCB problem when the caller does not get a CCB offer because the trunk gateway capacity has exceeded.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- CVP
- Cisco CVP Courtesy Callback

Components Used

The information in this document is based on these software versions:

- CVP Server 10.5
- Unified Contact Center Enterprise (UCCE) 10.5

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Background Information

Before the gateway capacity problem is troubleshoot, it is important to understand the trunk validation process in CCB. Basically, the process first determines the number of calls from the
Callback_current table with EventTypeID in (21,22,23); Pending, Inprogress, Tentative for specific gateways and locations.

Second, from the same Callback_current table, determine, the number of calls completed with cause connected: EventTypeID = 24 (Completed), and CauseID = 27 (Connected).

Finally the process adds these two values and compares with the number of trunks configured under the Survivability.tcl service.

If the result is over the trunks threshold configured, the process sends back a failure (return 1), otherwise sends back ok (return 0).

In summary, the formula to validate the trunks used for CCB is:

CCB Trunks < (Callback_current table with EventTypeID in (21,22,23); Pending, Inprogress, Tentative for specific gateways) + Callback_current table of EventTypeID = 24 (Completed), and CauseID = 27 (Connected)

If CCB Trunks value is lower validation fails.

Symptoms

An inbound call does not get CCB offer. The call goes directly to queue regardless the Estimated Wait Time (EWT)

Troubleshoot

Step 1. Collect Activity logs from the CallbackEntry application from the Voice Extensible Markup Language (VXML) Server.

Step 2. Search within the Activity logs for any call where validation is none: Validate_02,data,result,none

Which means that validation did not pass. Obtain the GUID for this call. Filter the call by the activity callid and look for a callid like this example:

start,parameter,callid=BBBBAAAAACCCDDEEEFFFFFAAABB

Step 3. Collect CVP reporting logs for the Reporting Server. Find the same callid in the CVP Reporting logs.

ValidateHandler:ValidateHandler.exec: ValidateHandler GUID=BBBBAAAAACCCDDEEEFFFFFAAABB results:none validation status bitmask=0x00000103

Step 4. Convert the bitmask number to binary. Use a programmer calculator: 0001 00000011

Step 5. Check the CVP Reporting guide bitmask for CCB tables. You should see that validation fails because of "EXCEED_CAPACITY_GW".
00000000 00000001 OK
00000000 00000010 ICM_NO_SCHEDULED_ALLOWED
00000000 00000100 ICM_NO_PREEMPTIVE_ALLOWED
00000000 00001000 NOT_IN_QUEUE
00000000 00010000 TOD
00000000 01000000 EWT
00000000 01000000 PROBE_FAILED_NO_RESPONSE
00000000 10000000 PROBE_FAILED_NO_CONFIG
00000001 00000000 EXCEED_CAPACITY_GW
00000010 00000000 EXCEED_CAPACITY_QUEUE

Note: ICM_NO_SCHEDULED_ALLOWED and the OK bit are always set

Step 6. Narrow the issue down to a specific queue. Check the CCB Servelet from the CVP Reporting server in order to determine if the there is any specific queue(s) where CCB is not offered. Open a web browser and type.

http://{Reporting Server IP Address}:8000/cvp/CallbackServlet?method=Diag

This is an example of a queue where CCB is offered:

![Servlet In-Memory Queue](image)

This is an example of a queue where CCB is not offered:

![Servlet In-Memory Queue](image)

Step 7. Check if the queue(s) are served by a specific gateway. Check the gateway configuration (Survivability application parameters).

```application
service new-call flash:bootstrap.vxml!
service survivability flash:survivability.tcl
paramspace callfeature med-inact-det enable
param ccb id:10.201.198.21;loc:CALO;trunks:512
```
Step 8. If the configuration is correct, check the information stored in the Reporting Server database (Informix) to determine the number of calls on this specific gateway and location. You can check by the CCB id (10.201.198.21 in this case) or locattion (CALO in this example).

Step 9. On the reporting server, access Informix database.

Open a CMD prompt and type: \texttt{dbacces}

Navigate to \texttt{connection > connect}

Select \texttt{cvp instance}

type username \texttt{cvp_dbadmin}

type password

select \texttt{callback@cvp database}

exit and navigate to Query Languages

Step 10. Run the query:

\begin{verbatim}
Select count(*) from callback_current where location == "CALO";
\end{verbatim}

Step 11. If the value is the same or higher than the trunk value configured in the gateway for the location(s), this is the reason why validation fails, since the maximum numbers of trunks allowed have been reached in the Callback_Current table.

\textbf{Note:} As referenced in the CVP Reporting guide, the Callback table is a view of two tables: Callback_Current and Callback_Historical. The two tables are identical. Every 30 minutes, data for completed calls is pulled from Callback_Pending and moved to Callback_Historical.

Step 12. If the trunk value per location has reached its limits in the Callback_Current table and there are no callbacks in the queue this indicates that there is a problem in moving the callback records from Callback_Current to the Callback_Historical table.

Step 13. Ensure that CVPCallbackArchive is running under the Schedule Tasks (CVP Reporting Server). Navigate to \texttt{Start -> Programs -> Accessories -> System Tools -> Scheduled Task}.
Step 14. If this task **CVPCallbackArchive** completes ensure the exit code is (0x0).

Step 15. If step 13 and 14 are fine, but still no data in the Callback_Historical table, you will need to determine why the information is not added in the database. Check the integrity of the information stored in the current and historical table. Run this query on the informix dbaccess CMD window:

```sql
Select count(*) from callback_current where surrogateid in (select surrogateid from callback_historical);
```

Step 16. If the count is 1 or higher, it means that the primary key on the current table already exist in the historical table and the information is not added into the database. In most of these scenarios, a race condition causes duplicate records to enter into callback_current table.

GUID to surrogateid mapping happens on queue table. In situations where the call moves from callback wait to callback queue script, there seems to be a window where archive job moves the records from current to history and the application enters a new record in the current table with the same surrogateid. This issue is related to this CDETS [CSCuq86400](https://example.com/cscuq86400).

**Solution**

Step 1. Access Informix database. Open a CMD prompt and type: **dbacces**

Step 2. Navigate to **connection > connect** select cvp instance. Type username cvp_dbadmin and type password

Step 3. Select callback@cvp database exit and navigate to **Query Languages**

Step 4. Run these commands:

```sql
delete from callback_current where surrogateid in (select surrogateid from callback_historical);
```

If there is a temporary table error do:

```sql
drop table t1;
```

Step 5. Run the sp procedure that moves the information from Current to historical callback table from the query language window dbaccess.

**EXECUTE PROCEDURE sp_arch_callback();**

Step 6. Check that there are not as many records in the current table as before.

```sql
Select count(*) from callback_current where location == "CALO";
```
Permanent Solution

Step 1. Navigate to Cisco\CVP\informix_frag and open sp_arch_callback.sql in a text editor.

Step 2. Uncomment this line at the beginning of the file: --drop procedure sp_arch_callback; (remove -- at beginning of the line).

Step 3. Add this line: delete from callback_current where surrogateid in (select surrogateid from callback_historical) ; after create procedure sp_arch_callback() line.

Step 4. Save the file.

Step 5. This is an example on how the first part of the file should look like.

{*********************************************************************************
* Stored procedure to move completed calls out of the active table into the historical table.  
*********************************************************************************}
drop procedure sp_arch_callback;
create procedure sp_arch_callback()
DEFINE p_ageoff INTEGER;
-- delete any duplicates found in current table.

delete from callback_current where surrogateid in (select surrogateid from callback_historical);

Test Final Solution

Step 1. Open a CMD prompt and run the command: dbschema
dbschema -d callback -f sp_arch_callback

Note: If you have an authorization issue when running the dbschema command, login as cvp_dbadmin into the reporting server and try one more time.

Step 2. From the output, ensure that the Delete from command is executed.
create procedure "Administrator".sp_arch_callback()
DEFINE p_ageoff INTEGER;

-- delete any duplicates found in current table.
delete from callback_current where surrogateID in (select surrogateID from callback_historical);

SELECT surrogateID
  FROM callback_current
 WHERE EventTypeID in (24,29) -- Completed, Too many callbacks
   AND CauseID in (27,28) -- Connected, Cancelled
 INTO TEMP t1 WITH NO LOG;