**Shared CC DB PATCHING INSTRUCTIONS**

Table of Contents

[1.0 PRE-PATCH Validation Steps 3](#_Toc55167885)

[1.1 Access Requirements 3](#_Toc55167886)

[1. 2 Check for any running jobs 3](#_Toc55167887)

[1.3 Turn-off all CRONS 3](#_Toc55167888)

[2.0 POST-PATCH validation Steps 4](#_Toc55167889)

[2.1 Turn-On all CRONS 4](#_Toc55167890)

[2.2 Check all scheduled interval jobs are running 4](#_Toc55167891)

[2.3 Validate ETL Job Management Tables 4](#_Toc55167892)

[2.3.1 Check if ETL job management scripts are getting executed 4](#_Toc55167893)

[2.3.2 Check for stuck jobs in ETL Job Management tables 5](#_Toc55167894)

[2.3.3 Reset stuck jobs in ETL Job Management tables 5](#_Toc55167895)

[2.3.4 Verify reset jobs are running fine 5](#_Toc55167896)

[2.4 Validate scheduled ***Contact Center*** job run 5](#_Toc55167897)

[2.4.1 Check if scheduled ***Contact Center*** job ran successfully 5](#_Toc55167898)

[2.4.2 Schedule an adhoc ***Contact Center*** job 6](#_Toc55167899)

[2.4.3 Check if adhoc ***Contact Center*** job is unsuccessful 6](#_Toc55167900)

[2.4.4 Add an entry in scheduled job table to avoid duplicate run 7](#_Toc55167901)

[2.5 Validate scheduled ***load\_ivr\_menu\_group*** job run 7](#_Toc55167902)

[2.5.1 Check if scheduled ***load\_ivr\_menu\_group*** job ran successfully 7](#_Toc55167903)

[2.5.2 Schedule an adhoc ***load\_ivr\_menu\_group*** job 7](#_Toc55167904)

[2.5.3 Check if adhoc ***load\_ivr\_menu\_group*** job is unsuccessful 8](#_Toc55167905)

[2.5.4 Add an entry in scheduled job table to avoid duplicate run 8](#_Toc55167906)

[2.6 Validate scheduled ***load\_call\_back*** job run 9](#_Toc55167907)

[2.6.1 Check if scheduled ***load\_call\_back*** job ran successfully 9](#_Toc55167908)

[2.6.2 Schedule an adhoc ***load\_call\_back*** job 9](#_Toc55167909)

[2.6.3 Check if adhoc ***load\_call\_back*** job is unsuccessful 10](#_Toc55167910)

[2.6.4 Add an entry in scheduled job table to avoid duplicate run 10](#_Toc55167911)

[3.0 CA HCO FILE FEED MISSING 12](#_Toc55167912)

[3.1 Schedule an adhoc ***cahco\_file\_feed*** job 12](#_Toc55167913)

[3.2 Verify adhoc ***cahco\_file\_feed*** job run was successful 12](#_Toc55167914)

[4.0 Updates to CC\_HCO\_F\_V2\_CALL 13](#_Toc55167915)

[4.1 Update CC\_A\_LIST\_LKUP table 13](#_Toc55167916)

[4.2 Execute UPDATE\_HCO\_V2\_CALL procedure 13](#_Toc55167917)

[4.3 Validate the data in Semantic view 13](#_Toc55167918)

# 1.0 PRE-PATCH Validation Steps

## 1.1 Access Requirements

* None as we assume DBA is running validation and has all requisite table accesses.
* Shared CC Production database url is uvacpmmora01mxd.maxcorp.maximus; port 1531

## 1. 2 Check for any running jobs

Check if all jobs have completed processing – run the following query to check.

**SELECT** *\**

**FROM** cisco\_enterprise\_cc.cc\_a\_scheduled\_job

**WHERE is\_running** = 1;

**SELECT \***

**FROM cisco\_enterprise\_cc.ETL\_JOB\_STATUS**

**WHERE JOB\_RUNNABLE = 'N';**

If the query returns no rows, then no jobs are running and we can proceed.

## 1.3 Turn-off all CRONS

Work with the app admin to make sure all crons are turned off. The app admins are provided a pre-patching document with detailed steps on what to check/verify before turning the crons off.

# 2.0 POST-PATCH validation Steps

## 2.1 Turn-On all CRONS

Work with app admin to ensure crons are turned back on after patching is completed.

## 2.2 Check all scheduled interval jobs are running

After confirmation of crons being turned back on - Verify if the scheduled interval jobs (that run at the 10th and 40th min of the hour) are **running** and **completing** successfully as expected.

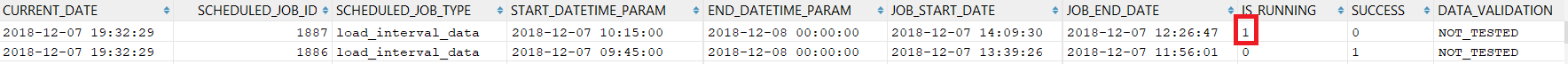
**SELECT** *CURRENT\_DATE*, **a**.*\**

**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB **a**

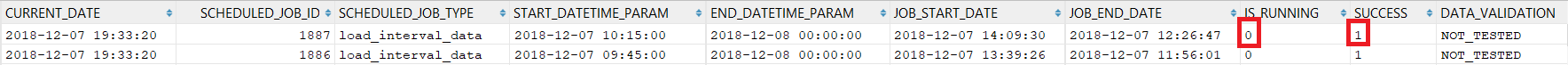
**WHERE SCHEDULED\_JOB\_TYPE** = **'load\_interval\_data'**

**ORDER BY SCHEDULED\_JOB\_ID DESC**;

IS\_RUNNING should == 1 when the job is **running**.



After the job successfully completes, IS\_RUNNING == 0 and SUCCESS == 1



If load\_interval\_data entries aren’t being created every 30 minutes, then check section 2.3.

If the load\_interval\_data fails with entries of IS\_SUCCESS = 0 and IS\_RUNNING = 0 (jobs have completed but not successfully) – this needs to be looked into for any database errors.

## 2.3 Validate ETL Job Management Tables

### 2.3.1 Check if ETL job management scripts are getting executed

Check in the ETL job management run table if any jobs started running after CRONs are enabled, by executing the below query.

If not, reach out to App Admins to check if the script for ETL job management jobs (etl\_job\_control.bash) is enabled in the CRON. Once App admins confirms the jobs are running, re-execute this step 2.3.1. If the jobs are still not running after 30 minutes, check 2.3.2.

**SELECT \***

**FROM cisco\_enterprise\_cc.ETL\_JOB\_RUN**

**WHERE RUN\_START\_DT >= '2020-11-02 14:00:00';**

**-- The above date/time should be corresponding to when CRONs were enabled after the patch**

### 2.3.2 Check for stuck jobs in ETL Job Management tables

If the ETL job management script is enabled in the CRON, but the jobs are still not running, check if any jobs in ETL Job management tables are stuck by running the below query. This query will return the JOB\_ID for any jobs stuck on that particular day. If this query returned no rows and jobs are still not running, reach out to a developer. If there are any stuck jobs, go to step 2.3.3.

**SELECT JOB\_ID**

**FROM cisco\_enterprise\_cc.ETL\_JOB\_LOG**

**WHERE LOG\_DESC = 'JOB STUCK - RESET MAY BE REQUIRED'**

**AND LOG\_UPD\_DT >= '2020-11-02 14:00:00';**

**-- The above date/time should be corresponding to when CRONs were enabled after the patch**

### 2.3.3 Reset stuck jobs in ETL Job Management tables

Reset each stuck job by executing the RESET\_ETL\_JOB procedure in CISCO\_ENTERPRISE\_CC schema for each JOB\_ID returned in the previous step. Replace “?” below with each JOB\_ID returned in the previous step.

**BEGIN**

**ETL\_JOB.RESET\_ETL\_JOB (?);**

**END;**

### 2.3.4 Verify reset jobs are running fine

ETL Job management shell script is usually scheduled to run every 5 or 10 minutes. So, wait for the next execution of the ETL job management scripts and check if the stuck jobs are started running. If not, reach out to the developer to have a look at the issue.

## 2.4 Validate scheduled ***Contact Center*** job run

### 2.4.1 Check if scheduled ***Contact Center*** job ran successfully

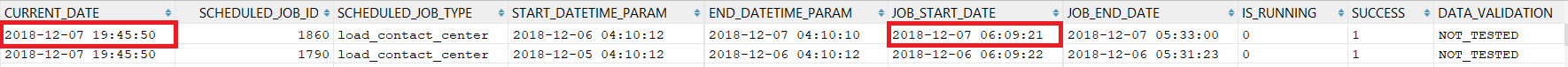
Verify if the scheduled contact center job (that runs once daily) is **ran that day** and **completed** successfully as expected – Please check if the date stamp matches to current date.

**SELECT** *CURRENT\_DATE*, **a**.*\**

**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB **a**

**WHERE SCHEDULED\_JOB\_TYPE** = **'load\_contact\_center'**

**ORDER BY SCHEDULED\_JOB\_ID DESC**;



### 2.4.2 Schedule an adhoc ***Contact Center*** job

If the job was not run that day, then insert the job entry into adhoc table for adhoc run using **yesterday’s** date as **START\_DATETIME\_PARAM** and **today’s** date as **END\_DATETIME\_PARAM**.

For example, if today is **12/07/2018** and load\_contact\_center missed its run today, then the insert query will be

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB (

**adhoc\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**is\_pending**,

**ACD\_SOURCE**,

**WFM\_SOURCE**)  
**VALUES** (

**'load\_contact\_center'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

1,

**'CISCO'**,

**'NA'**);

**COMMIT**;

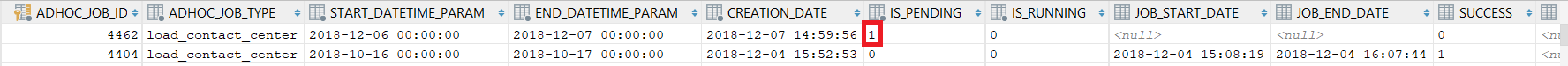
Check for the adhoc job to run (the job kicks off at the 35th minute of the hour):

**SELECT** *\**

**FROM** CISCO\_ENTERPRISE\_CC. CC\_A\_ADHOC\_JOB

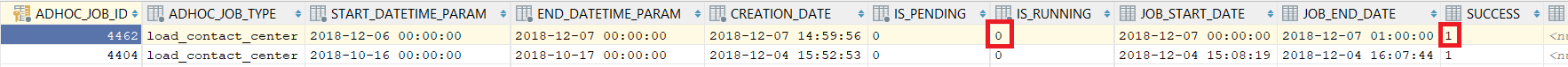
**WHERE ADHOC\_JOB\_TYPE** = **'load\_contact\_center'**

**ORDER BY ADHOC\_JOB\_ID DESC**;



Once the job kicks off, IS\_PENDING == 0 and IS\_RUNNING == 1

When the job finishes successfully the entry would be IS\_RUNNING == 0 and SUCCESS == 1



### 2.4.3 Check if adhoc ***Contact Center*** job is unsuccessful

If the job is unsuccessful where IS\_RUNNING == 0 and SUCCESS == 0, please reach out to app admin to investigate logs. App admin has instructions on how to check adhoc logs for errors. If required to resolve the error, please reach out to available MAXDAT developer.

### 2.4.4 Add an entry in scheduled job table to avoid duplicate run

Add a new entry in scheduled table for the date we just ran, so the job doesn’t run again next day for the same day. If today is **12/07/2018**, the **START\_DATETIME\_PARAM** will be of yesterday’s date and **END\_DATETIME\_PARAM** will be of today’s date and **JOB\_START\_DATE** and **JOB\_END\_DATE** will come from the previous query’s **JOB\_START\_DATE** and **JOB\_END\_DATE**.

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB (

**scheduled\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**job\_start\_date**,

**job\_end\_date**,

**success**)  
**VALUES** (

**'load\_contact\_center'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

*to\_date*(**'2018-12-07 00:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),   
 *to\_date*(**'2018-12-07 01:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),

1);

**COMMIT**;

## 2.5 Validate scheduled ***load\_ivr\_menu\_group*** job run

### 2.5.1 Check if scheduled ***load\_ivr\_menu\_group*** job ran successfully

Verify if the scheduled load\_ivr\_menu\_group job (that runs once daily) is **running** and **completing** successfully as expected.

**SELECT** *CURRENT\_DATE*, **a**.*\**

**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB **a**

**WHERE SCHEDULED\_JOB\_TYPE** = **'load\_ivr\_menu\_group'**

**ORDER BY SCHEDULED\_JOB\_ID DESC**;



### 2.5.2 Schedule an adhoc ***load\_ivr\_menu\_group*** job

If the job ran successfully after patching, move on to step 10. If the job was scheduled to be run and is not showing up, then insert the job entry into adhoc table for adhoc run using **yesterday’s** date as **START\_DATETIME\_PARAM** and **today’s** date as **END\_DATETIME\_PARAM**.

For example, if today is **12/07/2018** and load\_ivr\_menu\_group missed its run today, then the insert query will be

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB (

**adhoc\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**is\_pending**,

**ACD\_SOURCE**,

**WFM\_SOURCE**)  
**VALUES** (

**'load\_ivr\_menu\_group'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

1,

**'CISCO'**,

**'NA'**);

**COMMIT**;

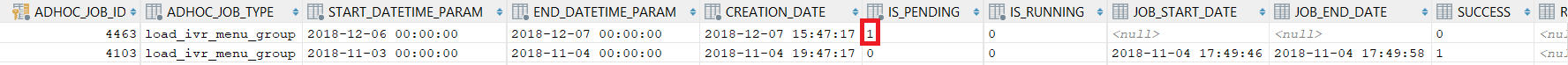
Check for the adhoc job to run (the job runs at the 35th minute of the hour):

**SELECT** *\**

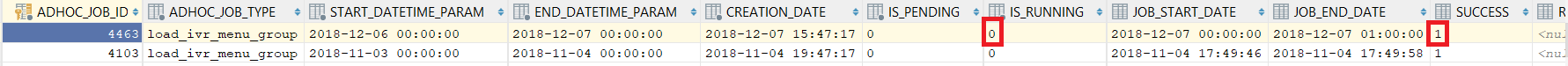
**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB

**WHERE ADHOC\_JOB\_TYPE** = **'load\_ivr\_menu\_group'**

**ORDER BY ADHOC\_JOB\_ID DESC**;



Once the job kicks off, IS\_PENDING == 0 and IS\_RUNNING == 0. The job finished successfully when IS\_RUNNING == 0 and SUCCESS == 1.



### 2.5.3 Check if adhoc ***load\_ivr\_menu\_group*** job is unsuccessful

If the job is unsuccessful where IS\_RUNNING == 0 and SUCCESS == 0, please reach out to app admin to investigate logs. App admin has instructions on how to check adhoc logs for errors. If required to resolve the error, please reach out to available MAXDAT developer.

### 2.5.4 Add an entry in scheduled job table to avoid duplicate run

Add a new entry in scheduled table for the adhoc entry we just ran, so the job doesn’t run again next day for the same day. If today is **12/07/2018**, the **START\_DATETIME\_PARAM** will be of yesterday’s date and **END\_DATETIME\_PARAM** will be of today’s date and **JOB\_START\_DATE** and **JOB\_END\_DATE** will come from the previous query’s **JOB\_START\_DATE** and **JOB\_END\_DATE**.

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB (

**scheduled\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**job\_start\_date**,

**job\_end\_date**,

**success**)  
**VALUES** (

**'load\_ivr\_menu\_group'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

*to\_date*(**'2018-12-07 00:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),   
 *to\_date*(**'2018-12-07 01:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),

1);

**COMMIT**;

## 2.6 Validate scheduled ***load\_call\_back*** job run

### 2.6.1 Check if scheduled ***load\_call\_back*** job ran successfully

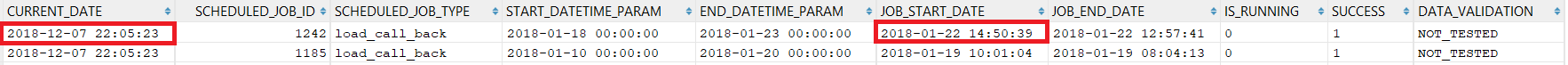
Verify if the scheduled load\_call\_back job (that runs once daily) is **running** and **completing** successfully as expected.

**SELECT** *CURRENT\_DATE*, **a**.*\**

**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB **a**

**WHERE SCHEDULED\_JOB\_TYPE** = **'load\_call\_back'**

**ORDER BY SCHEDULED\_JOB\_ID DESC**;



### 2.6.2 Schedule an adhoc ***load\_call\_back*** job

If the job ran successfully after patching, move on to step 12. If the job was scheduled to be run and is not showing up, then insert the job entry into adhoc table for adhoc run using **yesterday’s** date as **START\_DATETIME\_PARAM** and **today’s** date as **END\_DATETIME\_PARAM**.

For example, if today is **12/07/2018** and load\_call\_back missed its run today, then the insert query will be

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB (

**adhoc\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**is\_pending**,

**ACD\_SOURCE**,

**WFM\_SOURCE**)  
**VALUES** (

**'load\_call\_back'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

1,

**'CISCO'**,

**'NA'**);

**COMMIT**;

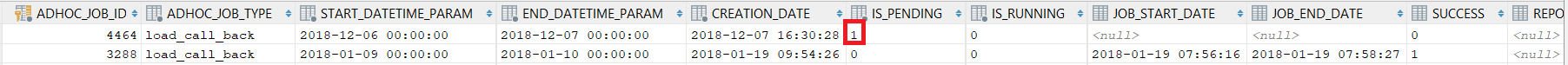
Check for the adhoc job to run (the job runs at the 35th minute of the hour):

**SELECT** *\**

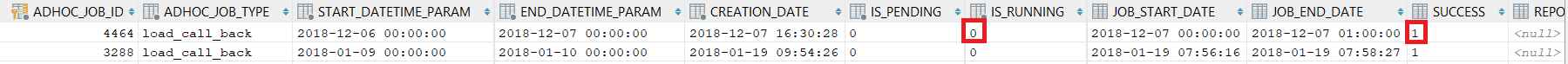
**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB

**WHERE ADHOC\_JOB\_TYPE** = **'load\_call\_back'**

**ORDER BY ADHOC\_JOB\_ID DESC**;



Once the job kicks off, IS\_PENDING == 0 and IS\_RUNNING == 0. The job finished successfully when IS\_RUNNING == 0 and SUCCESS == 1.



### 2.6.3 Check if adhoc ***load\_call\_back*** job is unsuccessful

If the job is unsuccessful where IS\_RUNNING == 0 and SUCCESS == 0, please reach out to app admin to investigate logs. App admin has instructions on how to check adhoc logs for errors. If required to resolve the error, please reach out to available MAXDAT developer.

### 2.6.4 Add an entry in scheduled job table to avoid duplicate run

Add a new entry in scheduled table for the adhoc entry we just ran, so the job doesn’t run again next day for the same day. If today is **12/07/2018**, the **START\_DATETIME\_PARAM** will be of yesterday’s date and **END\_DATETIME\_PARAM** will be of today’s date and **JOB\_START\_DATE** and **JOB\_END\_DATE** will come from the previous query’s **JOB\_START\_DATE** and **JOB\_END\_DATE**.

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_SCHEDULED\_JOB (

**scheduled\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**job\_start\_date**,

**job\_end\_date**,

**success**)  
**VALUES** (

**'load\_call\_back'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

*to\_date*(**'2018-12-07 00:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),   
 *to\_date*(**'2018-12-07 01:00:00'**, **'YYYY-MM-DD HH24:MI:SS'**),

1);

**COMMIT**;

# 3.0 CA HCO FILE FEED MISSING

## 3.1 Schedule an adhoc ***cahco\_file\_feed*** job

If the app admin reaches out regarding missing CA HCO files, we need to insert an adhoc entry in the adhoc table with **yesterday’s** date as **START\_DATETIME\_PARAM** and today’s date as **END\_DATETIME\_PARAM** parameters. If today is **12/07/2018**, the query is:

**INSERT INTO** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB (

**adhoc\_job\_type**,

**start\_datetime\_param**,

**end\_datetime\_param**,

**is\_pending**,

**ACD\_SOURCE**,

**WFM\_SOURCE**)  
**VALUES** (

**'load\_call\_back'**,

**'2018-12-06 00:00:00'**,

**'2018-12-07 00:00:00'**,

1,

**'CISCO'**,

**'NA'**);

**COMMIT**;

## 3.2 Verify adhoc ***cahco\_file\_feed*** job run was successful

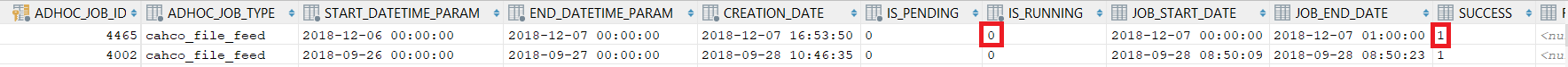
Verify if the adhoc job (that runs at the 35th min of the hour) is completing successfully as expected.

**SELECT** *\**

**FROM** CISCO\_ENTERPRISE\_CC.CC\_A\_ADHOC\_JOB

**WHERE ADHOC\_JOB\_TYPE** = **'cahco\_file\_feed'**

**ORDER BY ADHOC\_JOB\_ID DESC**;



Once the query has 1 in SUCCESS column, please let the app admin know to check for existence of new files.

# 4.0 Updates to CC\_HCO\_F\_V2\_CALL

This update is necessary to get the CC\_HCO\_F\_V2\_CALL table in sync with CC\_F\_V2\_CALL\_SV

## 4.1 Update CC\_A\_LIST\_LKUP table

Update CC\_A\_LIST\_LKUP table with the date on which load\_contact\_center job was run – usually sysdate

**update cc\_a\_list\_lkup**

**set out\_var = sysdate**

**where name = 'CAHCO\_V2\_CALL\_UPDATE\_DATE'**

**and list\_type = 'CAHCO\_V2\_CALL\_UPDATE\_DATE'**

**and value = 'CA HCO';**

**commit;**

## 4.2 Execute UPDATE\_HCO\_V2\_CALL procedure

Execute the following procedure to update the CC\_HCO\_F\_V2\_CALL table

**exec MAXDAT\_ADMIN.UPDATE\_HCO\_V2\_CALL;**

### 4.3 Validate the data in Semantic view

Once the procedure executes successfully, execute the queries below to validate the table data matches to the semantic view.

**select trunc(a.datetime), count(\*)**

**from cc\_f\_v2\_call\_sv a, cc\_d\_project b**

**where a.d\_project\_id = b.project\_id**

**and b.project\_name = 'CA HCO'**

**and trunc(a.datetime) >= '01-APR-19'**

**group by trunc(a.datetime)**

**order by trunc(a.datetime) desc;**

**select trunc(a.datetime), count(\*)**

**from cc\_hco\_f\_v2\_Call a**

**where trunc(a.datetime) >= '01-APR-19'**

**group by trunc(a.datetime)**

**order by trunc(a.datetime) desc;**

Both the above queries should produce the same result set.