Cisco IM&P External Databases Whitepaper (Best Practices)

7 June 2016

Purpose:

This high-level discussion is targeted to provide some detailed information related to the use of external (customer supplied) databases. This information is provided to assist a database administrator.
This information will cover:

- Persistent Chat database tables and schema definitions
- Advanced File Transfer database tables and schema definitions
- Compliance Message Archiver database tables and schema definitions
- Some useful database SQL commands
- Suggested database maintenance
- File server file maintenance and directory architecture

Disclaimers:

These specifications are subject to changes from release to release. No guarantees are made or implied that changes will be published on any timely basis.

The original assumptions, when these IM&P features were developed using "customer supplied" external databases, was that the provisioning and maintenance of these databases was the responsibility of the customer. Generally, the Cisco IM&P features will store data into these databases for historical and/or regulatory / compliance purposes. Searching or access to any specific information in the databases is left up to customers. There are a few of the persistent chat database tables that are used for writing, reading, and update/delete. But most are only written.

Also, this document includes the external databases as of CUCM / IM&P 11.5.1. Not all IM&P releases support all of these and there may be some differences in column definitions between releases.

Maintenance activities would include:

- Backup / archival (if required)
- Monitoring database storage space usage (tablespace)
- Removing or deleting obsolete database records
- Applying any needed updates / patches to database software
- Applying any O/S updates / patches on database server(s)
- Monitoring performance of database server(s) (CPU, Virtial Memory, I/O Wait, etc)
- Removal of aged or obsolete files from file server(s)

Database Tables and Schemas

Persistent Chat Database Tables

Table: tc_rooms

Column	Type (PG)	Type (Oracle)	Type(MSSQL)	Attribute	Description
room_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (PK)	Room jabber identifier
creator_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	Creator's jabber identifier
subject	Varchar(255)	Varchar2(255)	Varchar(255)		Room subject description
type	Varchar(32)	Varchar2(32)	Varchar(32)	Not Null (Index: **index_tc_rooms_type**)	Type of room: "ad-hoc", "persistent"
config	Text	Clob	Text		The entire packet from the last time the room was configured. This information enables the room to be reconfigured when the room is recreated (for example, at start-up)
spacket	Text	Clob	Text		The entire packet from the last time the subject was set for the room. This information enables the room subject to be displayed when the room is recreated.
start_msg_id	BigInt	Number(19)	BigInt		A sequence number that is used to populate the MSG_ID column in the tc_msgarchive table. DO NOT MODIFY.
next_msg_id	BigInt	Number(19)	BigInt	Not Null	A sequence number that is used to populate the MSG_ID column in the tc_msgarchive table. DO NOT MODIFY.

Rows in this table define each Chat Room that is created.
When a chat room administrator deletes the chat room the related row will be removed from this table.

Table: tc_users					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
room_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (**PK[0]**)	Room jabber identifier
real_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (**PK[1]**) (Index: **tc_users_real**)	User jabber identifier. This is actual ID of the user, rather th an alternate name.
role	Varchar(32)	Varchar2(32)	Varchar(32)	Not Null	The role of the user in the room. This value is constrained to one of the following: "none", "hidden",

Table: tc_users					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
					"visitor", "participant", or "moderator".
affiliation	Varchar(32)	Varchar2(32)	Varchar(32)	Not Null	The affiliation of the user in the room. This valud is constrained to one of the following: "none", "outcast", "member", "admin", or "owner".
nick_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)		The ID of the room, plus the alternate name for the user. The format is "room@tc-server/nick".
reason	Varchar(255)	Varchar2(255)	Varchar(255)		The reason entered when the user's affiliation was last changed.
initiator_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)		The ID of the room in which the configuration change occurred.

Rows in this table define all the users that are members / participants in each of the chat rooms.
The combination of room_jid + real_jid should product a unique key for each row.
When a chat room is deleted, all of the associated rows of users of the chat room are also deleted.
If a user is deleted from CUCM/IM&P Users list, any associated user rows **will not** be deleted from this table.

Table: tc_messages					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
msg_id	BigInt	Number(19)	BigInt	Not Null (PK)	The ID of the message. This is a unique identifier for each message per chat room; it is not globally unique.
room_jid	Varchar(3071)	Varchar2(2071)	Varchar(3071)	Not Null (Index: index_messages_jid)	Room jabber identifier that the message was sent to.
stamp	TimeStamp	TimeStamp	DateTime	(Index: index_messages_stamp)	The date and time the message was sent.
msg	Text	Clob	Text		The entire message.

Rows in this table contain historical text messages for each chat room up to the IM&P configured maximum number of historical persistent chat messages.

As each new message is added to the chat room, the historical records are removed to maintain the proper number of historical records. In effect, rows in this table are automatically self-limiting.

Table: tc_timelog					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
real_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (Index: index_messages_jid)	User jabber identifier entering or leaving room.
nick_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	The ID of the room, plus the alternate name for the user.
direction	Varchar(1)	Varchar(1)	Varchar(1)	Not Null	Indicates whether the user entered (E) or left (L) the room. Constrained to the values “E”, “L”
stamp	TimeStamp	TimeStamp	DateTime	Not Null	The date and time at which the user entered or left the room.

Rows in this table are written as a running log chat room user activity.

Table: tc_msgarchive					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
msg_id	BigInt	Number(19)	BigInt	Not Null (**index_tc_msgarchive_msgid**)	A unique identifier for the message.
to_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (**index_tc_msgarchive_tojid**)	Room destination jabber identifier.
from_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null (**index_tc_msgarchive_fromjid**)	Source user jabber identifier
nick_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	The ID of the room, plus the alternate name of the sender; for example: "room@conference.example.com/nick"
sent_date	TimeStamp	TimeStamp	DateTime	Not Null	The date and time on which the message was sent.
msg_type	Char(1)	Varchar(1)	Char(1)		The first character of the type attribute of the message. The possible values are "c" (chat), "g" (group chat), "h" (headline), and "e" (error).
body_len	Integer	Number(9)	Integer		The length in characters of the message body.
message_len	Integer	Number(9)	Integer		The length in character so the message, including the subject and body.
body_string	Varchar(4000)	Varchar2(4000)	Varchar(4000)		The message body.
message_string	Varchar(4000)	Varchar2(4000)	Varchar(4000)		The entire raw packet.
body_text	Text	Clob	Text		If the message body exceeds 4000 characters, it is stored in this field rather than the "body_string" field.
message_text	Text	Clob	Text		If the entire raw packet exceeds 4000 characters, it is stored in this column rather than in the "message_string" column.
subject	Varchar(255)	Varchar2(255)	Varchar(255)		The current subject of the room.

This table contains a row for each chat room message. This is an archive of messages for the chat rooms.

For Oracle DB and PostgreSQL DB, there is one additional index created:
index_tc_msgarchive_search on (to_jid, msg_type, sent_date, msg_id)

Persistent Chat Stored Procedures

```
tc_add_message_clear_old (
   @new_msg_id bigint output,
   @node_in varchar(3071),
   @stamp_in datetime,
   @text_in text,
   @max_msg_in int )

get_affiliation (
   @affil varchar(32) output,
   @room_jid varchar(3071),
   @user_jid varchar(3071) )

get_occupancy (
   @occupancy int output,
   @room_jid varchar(3071))
```

Advanced / Managed File Transfer Database Tables

Table: aft_log					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
aft_index	BigInt	Number(19)	BigInt	Not Null (**PK**)	
Jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	
method	Varchar(63)	Varchar2(63)	Varchar(63)	Not Null	
filename	Varchar(511)	Varchar2(511)	Varchar(511)	Not Null	File name transferred
filesize	BigInt	Number(19)	BigInt	Not Null	
bytes_transfered	BitInt	Number(19)	BitInt		
timestampvalue	TimeStamp	TimeStamp	DateTime	Not Null	

Rows in this provide information about an IM&P client user downloading or uploading a file.

Compliance Message Archiver

Table: jm					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
to_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	The Jabber ID of the user who is receiving the message being archived.
from_jid	Varchar(3071)	Varchar2(3071)	Varchar(3071)	Not Null	The Jabber ID of the user who is sending the message being archived.
sent_date	TimeStamp	TimeStamp	DateTime	Not Null	The date and time the message was sent.
subject	Varchar(128)	Varchar2(128)	Varchar(128)		Subject line of the message.
thread_id	Varchar(128)	Varchar2(128)	Varchar(128)		The thread ID of the message.
msg_type	Char(1)	Varchar(1)	Char(1)	Default='N'	The first character of the message type attribute. Possible values are: • "c" – chat • "n" – normal • "g" – groupchat • "h" – headline • "e" – error
direction	Char(1)	Varchar(1)	Char(1)		Direction of the message; Outgoing='O', Incoming='I'. If the message is sent between users on the same server, it is logged twice; once as outgoing and once as incoming.
body_len	Integer	Number(9)	Integer		The number of characters in the message body.
message_len	Integer	Number(9)	Integer		The number of characters in the message, including the subject and the body.
body_string	Varchar(4000)	Varchar2(4000)	Varchar(4000)		The message body.
message_string	Varchar(4000)	Varchar2(4000)	Varchar(4000)		The entire raw packet.
body_text	Text	Clob	Text		If the message body exceeds 4000 characters, it is stored in this field rather than the "body_string" field.
message_text	Text	Clob	Text		If the raw packet exceeds 4000 characters, it is stored in this field rather than the "message_string" field.
history_flag	Char(1)	Varchar(1)	Char(1)		Used when the room history messages are sent to new participants (upon entering an existing room). This allows you to distinguish between messages received while actively participating in a room and those received as part of a history push. The

Table: jm					
Column	**Type (PG)**	**Type (Oracle)**	**Type(MSSQL)**	**Attribute**	**Description**
					latter message type is flagged with "H". Otherwise this field is "N".

Rows in this table contains messages received and optionally the messages sent from each user. There is a configuration setting on whether to include the sent (outbound) messages in this log.

There are some defined views for this table:
View: **message_threads**
SELECT DISTINCT thread_id, to_jid, sent_date FROM jm where direction='O':
View: **conversations**
SELECT thread_id, from_jid, to_jid, sent_date, msg_type, subject, body_len, body_text, body_string FROM jm WHERE direction='O' ORDER BY sent_date;

External Database Maintenance

Table	Maint Req'd
tc_rooms	No
tc_users	No
tc_messages	No
tc_timelog	Yes
tc_msgarchive	Yes
aft_log	Yes
jm	Yes

The tables with "Maint Req'd" values of "Yes" indicates that these tables will continuously grow in number of records (rows) and require a customer to periodically perform some maintenance by removing records (rows) that are determined to be old or obsolete. Many times this might be determined based on when the records were inserted into the database table.

Persistent Chat Database Tables

Useful general SQL commands:
WARNING:
Many of these SQL commands may have a VERY significant impact on any active online features due to causing significant database server activity. It is HIGHLY recommended that use of any of these be done at off-peak times or during a maintenance time window. Some of these could also take a significant amount of time to run. Database performance of some of these commands may be improved by creating indexes on some of the selected date / timestamp fields

but this may cause future IMP updates to fail if these indexes exist during upgrades.

Tc_rooms

```
SELECT COUNT(*) FROM tc_rooms;   /* Displays number of Rooms in table */
SELECT subject, type, room_jid, creator_jid FROM tc_rooms;    /* List rooms in table */
```

Tc_users

```
SELECT COUNT(*) FROM tc_users;  /* Displays total number of rows in table */
SELECT room_jid, real_jid, role, affiliation FROM tc_users ORDER BY room_jid, real_jid;  /* List table */
```

Tc_messages

```
SELECT COUNT(*) FROM tc_messages;   /* Displays total number of rows in table */
    /* # of rows before 2016 */
MSSQL:      SELECT COUNT(*) FROM tc_messages WHERE YEAR(stamp) < 2016;
Oracle:     SELECT COUNT(*) FROM tc_messages WHERE stamp < to_date('01-JAN-2016');
PostgreSQL: SELECT COUNT(*) FROM tc_messages WHERE stamp < to_date('01-01-2016','DD-MM-
YYYY');
```

Tc_timelog

```
SELECT COUNT(*) FROM tc_timelog; /* # rows in table */
    /* # rows written before 2016 */
MSSQL:      SELECT COUNT(*) FROM tc_timelog WHERE YEAR(stamp) < 2016;
Oracle:     SELECT COUNT(*) FROM tc_timelog WHERE stamp < to_date('01-JAN-2016');
PostgreSQL: SELECT COUNT(*) FROM tc_timelog WHERE stamp < to_date('01-01-2016','DD-MM-YYYY');
    /* # rows older 6 mo */
MSSQL:      SELECT COUNT(*) FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182);
PostgreSQL: SELECT COUNT(*) FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182);
Oracle:     SELECT COUNT(*) FROM tc_timelog WHERE stamp < (sysdate – 182);
    /* list the rows */
MSSQL:      SELECT * FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182) ORDER BY stamp;
PostgreSQL: SELECT * FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182) ORDER BY stamp;
Oracle:     SELECT * FROM tc_timelog WHERE stamp < (sysdate – 182) ORDER BY stamp;
    /* Delete the rows */
MSSQL:      DELETE FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182);
PostgreSQL: DELETE FROM tc_timelog WHERE stamp < (CURRENT_DATE – 182);
Oracle:     DELETE FROM tc_timelog WHERE stamp < (sysdate – 182);
```

Tc_msgarchive

```
SELECT COUNT(*) FROM tc_msgarchive;   /* Total # rows in table */
    /* # rows older than 6 months */
MSSQL:      SELECT COUNT(*) FROM tc_msgarchive WHERE sent_date < (CURRENT_DATE – 182);
PostgreSQL: SELECT COUNT(*) FROM tc_msgarchive WHERE sent_date < (CURRENT_DATE – 182);
Oracle:     SELECT COUNT(*) FROM tc_msgarchive WHERE sent_date < (sysdate – 182);
    /* Delete rows older than 6 months */
MSSQL:      DELETE FROM tc_msgarchive WHERE sent_date < (CURRENT_DATE – 182);
PostgreSQL: DELETE FROM tc_msgarchive WHERE sent_date < (CURRENT_DATE – 182);
Oracle:     DELETE FROM tc_msgarchive WHERE sent_date < (sysdate – 182);
```

Aft_log

```
SELECT COUNT(*) FROM aft_log;   /* Total # rows in table */
    /* # rows older than 6 months */
```

```
MSSQL:      SELECT COUNT(*) FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
PostgreSQL: SELECT COUNT(*) FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
Oracle:     SELECT COUNT(*) FROM aft_log WHERE timestampvalue < (sysdate – 182);
    /* List rows older than 6 months */
MSSQL:      SELECT * FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
PostgreSQL: SELECT * FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
Oracle:     SELECT * FROM aft_log WHERE timestampvalue < (sysdate – 182);
    /* List files older than 6 months */
MSSQL:      SELECT timestampvalue, filename FROM aft_log WHERE timestampvalue <
(CURRENT_DATE – 182);
PostgreSQL: SELECT timestampvalue, filename FROM aft_log WHERE timestampvalue <
(CURRENT_DATE – 182);
Oracle:     SELECT timestampvalue, filename FROM aft_log WHERE timestampvalue < (sysdate – 182);
    /* Delete rows older than 6 months */
MSSQL:      DELETE FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
PostgreSQL:         DELETE FROM aft_log WHERE timestampvalue < (CURRENT_DATE – 182);
Oracle:     DELETE FROM aft_log WHERE timestampvalue < (sysdate – 182);
```

Jm

```
SELECT COUNT(*) FROM jm;   /* Total # rows in table */
    /* Number of rows older than 6 months */
MSSQL:      SELECT COUNT(*) FROM jm WHERE sent_date < (CURRENT_DATE – 182);
PostgreSQL: SELECT COUNT(*) FROM jm WHERE sent_date < (CURRENT_DATE – 182);
Oracle:     SELECT COUNT(*) FROM jm WHERE sent_date < (sysdate – 182);
    /* Delete rows older than 6 months */
MSSQL:      DELETE FROM jm WHERE sent_date < (CURRENT_DATE – 182);
POstgreSQL: DELETE FROM jm WHERE sent_date < (CURRENT_DATE – 182);
Oracle:     DELETE FROM jm WHERE sent_date < (sysdate – 182);
```

Advanced File Transfer Maintenance

The file transfer server maintenance must be done by the customer / owner of the file server. This maintenance includes:

- Monitoring available disk storage space for file storage
- Monitoring server availability
- Server Backup / Restore / Reliability operations
- Removal of Aged or Obsolete files (maintenance of file disk space) (**on-going periodic maintenance**)

On AFT (Advanced File Transfer) servers the following directory architecture is maintained:

/opt/mftFileStore/node_1/files/im/yyyymmdd/hh/…
/opt/mftFileStore/node_1/staging/im/yyyymmdd/hh/…

“node_#” : represents the IM&P server node
“yyyymmdd” : Date file was uploaded
“hh” : Hour file was uploaded (00-23)

The following can be put into a bash shell script to be used for doing file deletions:

```
#!/bin/sh
basepath="/opt/mftFileStore"
imsubpath="/files/im"
stagesubpath="/staging/im"

help() {
   echo "Usage: $0 yyyy [test]"
   echo "  yyyy : 4-digit year where all files earlier than this year are removed."
   echo "  Use 'test' parameter to run without actually deleting anything."
   echo "Examples:"
   echo "  $0 2016 test  # Shows what would be removed for earlier than 2016 year."
   echo "  $0 2016       # Actually remove the files and directories."
   exit 1
}

if [ "$1" == "" ]; then
   echo "Missing parameter (year)"
   help
fi
if [ "$1" == "-h" ]; then
   help
fi
if [ "$1" == "-H" ]; then
   help
fi
if [ "$1" == "--help" ]; then
   help
fi
let yr=$1
test=$2
if [ -d "${basepath}" ]; then
   pushd "${basepath}" >/dev/null
else
   echo "ERROR: ${basepath} does not exist."
   exit 1
fi
for node in `ls`; do
   if [ -d "${basepath}/${node}${imsubpath}" ]; then
      pushd "${basepath}/${node}${imsubpath}" >/dev/null
      for f in `ls`; do
         if [[ "${f}" < "${yr}" ]]; then
            if [ "${test}" == "test" ]; then
               echo "  Remove ${basepath}/${node}${imsubpath}/${f}"
            else
               echo "  rm -rf ${basepath}/${node}${imsubpath}/${f}"
               rm -rf "${basepath}/${node}${imsubpath}/${f}"
            fi
         fi
      done
      popd >/dev/null
   fi
   if [ -d "${basepath}/${node}${stagesubpath}" ]; then
```

```
        pushd "${basepath}/${node}${stagesubpath}" >/dev/null
        for f in `ls`; do
            if [[ "${f}" < "${yr}" ]]; then
                if [ "${test}" == "test" ]; then
                    echo "  Remove ${basepath}/${node}${stagesubpath}/${f}"
                else
                    echo "  rm -rf ${basepath}/${node}${stagesubpath}/${f}"
                    rm -rf "${basepath}/${node}${stagesubpath}/${f}"
                fi
            fi
        done
        popd >/dev/null
    fi
done

popd >/dev/null
exit 0
```

Useful External References:

[Database Setup for IM and Presence Service on Cisco Unified Communications Manager Release 11.5(1)]

Includes information on general DB setup as well as specifics for PostgreSQL and Oracle with IM&P Services setup.
Also contains some specifics on the database tables and schemas.

[Cisco Unified Communications Manager IM & Presence Service]

Documentation for various releases of Cisco Unified Communications Manager IM & Presence Service.