avaloq
An NEC Company

Oracle Multitenant Support

Customization Upgrade Guide
Release 5.6
Area: Technical Base

Latest version of this document

The latest version of this document can be found at https://docs.avaloq.com

Feedback

Please send any feedback to documentation@avaloq.com

Avaloq Group Ltd. Allmendstr. 140 ¦ CH-8027 Zürich ¦ Switzerland

Version history

Version / Date	Section	Description of the change
5.6v5 /13 February 2025		Added new section on External Password Management (EPM).
5.6v4 / 14 August 2024		Replaced Oracle 23c with Oracle 23ai.
5.6v3 / 26 April 2023		Added a paragraph at the end of "Step 3: Create a CDB" in section 1.3.
5.6v2 27 January 2023	• "General Oracle Multitenant migration considerations" on page 6 • "Migration process" on page 7	Small clarification edits.
5.6v1 / 30 November 2022	• "General Oracle Multitenant migration considerations" on page 6 • "Step 4: Plug in the ACP DB into the CDB" on page 10	Updates to mention the various options for the plug in of the non-Multitenant ACP database into the CDB.
5.6v0 / 12 October 2022		New document for ACP 5.6.

Contents

1 Introduction

Starting with ACP release 5.6, Avaloq supports the Oracle Multitenant architecture for ACP databases, having one pluggable database only. Switching to Oracle Multitenant with ACP 5.6 is optional.

With Oracle 23ai, the next Oracle release for ACP, the Oracle Multitenant architecture will become mandatory. This is mandated by Oracle.

To de-couple the Multitenant migration from the migration to Oracle 23ai within the ACP upgrade, we strongly recommend that you already change to the Multitenant architecture with ACP 5.6 on Oracle 19c.

This change can be part of the ACP 5.6 release upgrade or it can be a separate migration step after the ACP 5.6 upgrade. That decision is mainly based on these factors:

- How long it takes to convert a database to a Multitenant database
- What changes are needed to your operational tasks and procedures, for example, database backup

This document describes the steps needed to set up the Avaloq Unix environment for a Multitenant ACP database, and gives a step by step instructions on how migrate an existing non-Multitenant ACP database.

Please consult the official Oracle documentation on https://docs.oracle.com to make yourself familiar with the Multitenant feature beforehand.

We use the following terms in this documentation:

Term	Meaning
CDB	Container database
PDB	Pluggable database
PDBName	Name of the PDB (this is not an ORACLE_SID)

Additional steps may be required if you are using the Avaloq External Password Management (EPM) feature.

If you are using that feature, read "External Password Management (EPM)" on page 12 before you migrate to the Oracle Multitenant architecture.

1.1 Major changes to the Avaloq Unix

- There is a new Avaloq-specific shell environment variable AAA_PDB_NAME. You must define this environment variable in the **aaatab.conf.1** file for the designated CDB. It specifies the PDBName in that CDB.

  Here's an example:

```
*  orac3d00032  aaaservice  mount_point      /orabin
*  c3d00032     env         AAA_SERVICE      orac3d00032
```

```
*  c3d00032     env         AAA_PDB_NAME     c3d00pdb
*  c3d00032     env         ORACLE_HOME      /orabin/app/oracle/product/19.13.0.0.211019-195
*  c3d00032     env         ORACLE_DB_ADMIN  /orabin/app/oracle/admin/c3d00032
*  c3d00032     env         ORACLE_DB_DATA   /orac3d00032/oradata/c3d00032
```

This example sets the PDBName to c3d00pdb for the CDB c3d00032. Notice that although the term PDB indicates that this is a database, no entries for it are needed in **aaatab.conf.1**.

By definition, the setup is non-CDB if either:

- The AAA_PDB_NAME environment variable is not defined
- The environment variable is defined but is empty

- The **aaainstall.sh** shell script accepts a new optional parameter:

  `-pdb <PDBName>`

  This parameter:

  - Must be specified when installing the Avaloq Unix environment for a CDB
  - Must **not** be specified when installing the Avaloq Unix environment for a non-CDB

  Here's an example for an interactive Avaloq Unix installation for a CDB:

  ```
  $> ksh ./aaainstall.sh -sid c3d00032 -pdb c3d00pdb -install all -compile -aaatab
  ```

  This will install the Unix node and DB part for the CDB c3d00032 and the contained PDB c3d00pdb.

- There is a new shell script **aaacrdb_cdb.ksh** to create an empty CDB that **must** be used to plug in an existing non-Multitenant ACP DB as a PDB.

1.2 General Oracle Multitenant migration considerations

You cannot convert an existing non-Multitenant ACP database into a CDB. Instead, you must create a new Oracle DB as a CDB, and then plug your existing non-Multitenant ACP database into that CDB.

For the plug in of the non-Multitenant ACP database as a new PDB, two main options are available:

- create_pdb_from_xml
- NON$CDB@<dblink>

Other options are also possible. Avaloq gives no recommendation about which option to use.

Because Avaloq uses the NON$CDB@<dblink> option in its inhouse build environment, we describe this option in the next section. If you want to use a different option, please consult the Oracle documentation.

Both of the options listed above work in a similar way, but with one big difference: only create_pdb_from_xml offers the "move" option. The move option does not create a copy of the files, but instead it moves the files.

If the ${ORACLE_DB_DATA} of both the non-Multitenant database and the CDB are on the same mountpoint, moving the files can sometimes be faster than – and require less disk space than – copying the

files. However, this is not always the case. It depends on various factors including the OS and storage system.

The duration of the plugin procedure depends very much on which option you choose, and on the size of the DB and IO performance. It can take several hours.

With the copy option, you will need twice the diskspace (for the non-Multitenant and the Multitenant ACP DB).

A drawback of the move option is that if something goes wrong, the non-Multitenant DB no longer exists, and may need to be rebuilt completely from a valid full backup. A simple move back of the files may not be possible in the case of a media failure or if the execution of the **noncdb_to_pdb.sql** script has failed.

If you plan to do the Multitenant migration as part of the ACP 5.6 upgrade, this must be done directly after the Avaloq 5.6 Unix upgrade but before the ACP 5.6 kernel and customization upgrade (see *Upgrade Guide – Upgrade Document (doc ID: 1606)*).

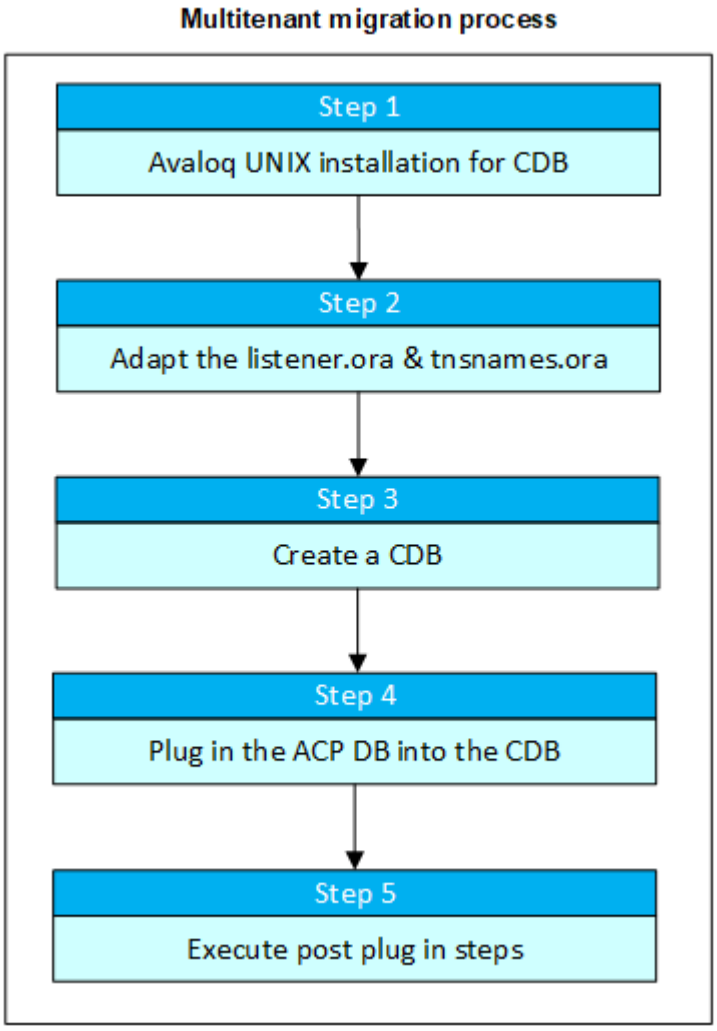
Because the new PDBName might correspond to your existing ACP DB name, and the new CDB must have a different name, please select a suitable new naming schema for all your ACP DBs beforehand. For example, you can add a "c" prefix or postfix to the CDB name and use the existing ACP DB name for the PDBName.

Note that the PDBName in the Oracle DB is case insensitive and always uppercase. The value for AAA_PDB_NAME can be mixed case but must comply with the naming restrictions specified by Oracle.

1.3 Migration process

This diagram shows the necessary steps for the migration of a non-Multitenant ACP DB into a Multitenant ACP DB. See the detailed instructions for each step below. Steps 1-3 and 5 must be executed exactly as described below. Step 4 is an example for using the NON$CDB@<dblink> option only.

If you plan to also upgrade to a new Oracle Configuration, i.e. to a new Oracle release upgrade (RU) we recommend that you upgrade the non-multitenant DB to the new target Oracle configuration beforehand. If you decide to create the CDB with a newer Oracle binary configuration, you must run $ORACLE_HOME/OPatch/datapatch directly after the execution of the **noncdb_to_pdb.sql** script.



Step 1: Avaloq Unix installation for CDB

1. Enter the new Oracle CDB database (ORACLE_SID) in **/etc/oratab**.
2. Add the new Oracle CDB database to the **aaatab.conf.1** file, including the AAA_PDB_NAME variable and merge the content into the **aaatab.conf** file.
3. Install the ACP 5.6 Avaloq Unix with the `-sid` and `-pdb` options.

   Example for the CDB c3d00032 and the contained PDB with PDBName c3d00pdb:

   ```
   $> ksh ./aaainstall.sh -sid c3d00032 -pdb c3d00pdb -install all -compile -aaatab
   ```

   This will install the Unix node and DB part for the CDB c3d00032 and the contained PDB c3d00pdb.
4. Re-source the shell for that ORACLE SID so that AAA_PDB_NAME is set properly.

It is forbidden to set the shell environment variable ORACLE_PDB_SID. Although it is mentioned in various Oracle related web sites, it is not an officially documented Oracle feature. Setting this variable will result in severe errors in various Avaloq Unix scripts.

Step 2: Adapt the listener.ora & tnsnames.ora

With the Multitenant architecture, all DB connections to the ACP application must go to the PDB and not to the CDB. Because the PDB is not a classical Oracle DB with a SID, some Oracle Net amendments are necessary. This section describes the necessary adaptations. It may only be partially applicable, depending on your setup. If you have any questions or anything is unclear, consult the relevant Oracle documentation.

listener.ora

The listener entry for the CDB listener must be configured to use the parameter USE_SID_AS_SERVICE_LISTENER, and the SID_NAME must be the PDBName.

Example for the CDB c3d00032 and the contained PDB with PDBName c3d00pdb:

```
USE_SID_AS_SERVICE_LISTENER_orac3d00032=ON

LISTENER_orac3d00032=
  (DESCRIPTION_LIST=
    (DESCRIPTION= (ADDRESS_LIST= (ADDRESS= (PROTOCOL= TCP)(HOST= c3d00032.sys.net)(PORT= 1522))))
  )
SID_LIST_LISTENER_orac3d00032=
    (SID_LIST= (SID_DESC= (ORACLE_HOME= /orabin/app/oracle/product/19.13.0.0.211019-195) (SID_
NAME= c3d00pdb)))
```

tnsnames.ora

For the TNS Alias the SID must be the PDBName including the DB_DOMAIN, and the Alias name must be:

PDBName.DB_DOMAIN

Example for the CDB c3d00032 and the contained PDB with PDBName c3d00pdb and DB_DOMAIN avaloq:

```
c3d00pdb.avaloq =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(Host = c3d00032.sys.net)(Port = 1522))
    (CONNECT_DATA = (SID = c3d00pdb.avaloq)(GLOBAL_NAME = c3d00032.avaloq))
  )
```

The Avaloq Unix environment contains **listener.ora** and **tnsnames.ora** template files. The files will not be installed in **/var/opt/oracle** unless they don't exist there, to avoid overwriting your custom configuration.

You can consult the generated ***.substituted** files in the folder of the Avaloq Unix file. These ***.substituted** files will only be present if an Avaloq Unix installation has been executed from there.

In ADAI you might want to use a guest either for a Multitenant or a non-Multitenant DB. This means that you will need two TNS Alias entries.

Step 3: Create a CDB

Use the new **aaacrdb_cdb.ksh** script to create an empty CDB:

```
$> aaacrdb_cdb.ksh -s <ORACLE_SID> -sys <PW> -system <PW>
```

You can use the optional parameters `-sys` and `-system` to specify the passwords for the those DB users. If you do not use these parameters, their passwords will be set to "aaamanager".

This shell script uses the template script **$ORACLE_DB_ADMIN/create/crdb_cdb${ORACLE_SID}.sql**, which has low initial sizes for the CDB tablespaces and the redo logs. You can change these settings beforehand.

This shell script also uses the pfile installed with the Avaloq Unix environment. You can use your own pfile, for example, an adapted copy from your existing non-Multitenant ACP DB. If you do this, make sure that the parameter `enable_pluggable_database` is set to true.

For other Multitenant-specific database parameters see document *Current Oracle Configuration - Reference Guide (doc ID: 2411)*. Multitenant-specific parameters have an explicit remark in that document. Please note also that Oracle parameter configuration 143 or newer must be chosen. This also means that you have to be on the Oracle binary configuration 19.17.0.0.221018-199 or newer.

Step 4: Plug in the ACP DB into the CDB

There are various procedures you can use to implement this step.

In this step, we describe the procedure if you are using the NON$CDB@<dblink> option without ASM (Automatic Storage Management). This is the procedure that is used and tested internally at Avaloq.

If you decide to use a different procedure:

- Ignore this step and instead consult the Oracle documentation for information on how to plug in the ACP DB into the CDB.
- Then proceed to step 5.

Make sure that your non-Multitenant ACP DB is in session level zero, that all jobs – such as EOD (End of Day) – have been completed, and that all external interfaces are disconnected. Also, a current backup must exist, at least for your production DB.

Follow these steps on the non-Multitenant ACP DB as sysdba:

1. Create a DB user as follows:

```
SQL> create user <MIGR_USER> identified by <PW>;
SQL> grant CONNECT, RESOURCE, CREATE PLUGGABLE DATABASE to <MIGR_USER>;
```

2. Set the DB to read-only:

```
SQL> shutdown immediate
SQL> startup open read only
```

Then follow these steps on the CDB as sysdba:

1. Create a DB-link to the non-Multitenant ACP DB:

```
SQL> create database link <DB_LINK_NAME> connect to <MIGR_USER> identified by <PW> using '<ACP
DB>';
```

2. Plug in the DB:

```
SQL> create pluggable database <PDBName> FROM NON$CDB@<DB_LINK_NAME> file_name_convert=('<SRC_
DATA_FILES_PATH>','<DEST_DATA_FILE_PATH>');
```

   where:

   - <SRC_DATA_FILES_PATH> is the full path to the datafiles of the non-Multitenant DB including a trailing /
   - <DEST_DATA_FILE_PATH> is the full path to the datafiles of the CDB including a trailing /

   In a non-ASM setup, both paths are equal to the ORACLE_DB_DATA settings in **aaatab.conf**.

3. Check for possible PDB violations:

```
SQL> set linesize 150
col name    format a20
col cause   format a25
col message format a35
col status  format a8


select name
       ,cause
       ,type
       ,message
       ,status
from   PDB_PLUG_IN_VIOLATIONS
where  type    = 'ERROR'
       and  status != 'RESOLVED';
```

   There should not be any unresolved errors.

4. Execute the **noncdb_to_pdb.sql** script from within the new PDB:

```
SQL> alter session set container = <PDBName>;
SQL> @?/rdbms/admin/noncdb_to_pdb.sql
```

5. Check that the ACP PDB has been successfully created:

```
SQL> show pdbs
```

   This should result in this output (the CON_ID might be different):

```
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 <PDBName>                      MOUNTED
```

6. Open the ACP PDB and save its state:

```
SQL> alter pluggable database <PDBName> open;
SQL> alter pluggable database <PDBName> save state;
```

7. The ACP PDB should now be open:

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         3 <PDBName>                      READ WRITE
```

Step 5: Execute post plug in steps

1. Restart the CDB with **aaadbctrl.ksh**:

```
$> aaadbctrl.ksh -s <ORACLE_SID> -c restart -o nogarbol
```

2. Execute the CODE_DB_COPY_ACTIONs in the PDB as user **K**:

```
SQL> begin
       k.config_intf#.after_db_copy_action(
         i_action  => config_intf#.c_after_db_file_copy
         ,i_is_prod => '-'
       );
     end;
     /
```

   If this is the designated production CDB, the parameter `i_is_prod` must be set to "+".

2 External Password Management (EPM)

Read this section if you are using the Avaloq External Password Management feature, which is described in Protecting external passwords.

The migration to the Oracle Multitenant architecture potentially changes some database characteristics that are used by EPM for the EPM keystore master key calculation, so that access to EPM may not be possible after the migration. This is the case if any of the following values of the Avaloq the `avq.epm` base parameter's `master_key_calc_fact_list` item:

- `db_id`
- `host_id`
- `instance_id`

are included in the list obtained from the following query:

```
select epm_par_admin#.master_key_calc_fact_list FROM dual;
```

The values `host_id` and `instance_id` are node-specific, which means that the value is different for the different nodes (systems) that access the ACP database. For these values, access to EPM will be blocked in multi-node environments.

The following table lists these values and their dependencies.

master_key_calc_fact_list	Derived from	Node-specific
`db_id`	The database identifier (Oracle DBID)	No
`host_id`	The hostname of the database instance	Yes
`instance_id`	The database instance name	Yes

In the following text, we use the term **database characteristics** to refer to the three values.

Banks that are affected by this problem must either:

- Recreate their EPM keystore and reload the secrets **after migrating** to an Oracle Multitenant database

  For information on how to recreate the EPM keystore, see Protecting external passwords.

- Migrate their existing EPM keystore **during the migration** to an Oracle Multitenant database

  In this case, follow the steps below.

2.1 Migrating an existing EPM keystore

To migrate your existing keystore during the migration, follow these steps:

1. Make a note of the current values of the database characteristics (`db_id`, `host_id`, `instance_id`) in the `avq.epm` base parameter's `master_key_calc_fact_list` item.
2. Migrate your existing EPM keystore without including any of the database characteristics (`db_id`, `host_id`, `instance_id`) that are used for the EPM master key calculation.

Before you follow the steps to migrate to Oracle Multitenant, change the EPM keystore password to a password that does not use the database characteristics (`db_id`, `host_id`, `instance_id`).

Set the input parameter `i_master_key_calc_fact_list` in a procedure call to `epm_par_admin#.set_master_key_calc_fact_list` to null.

For example:

```
exec dbms_session.reset_package;
begin
  -- Start session --
  session#.open_session;

  -- Set session level "0" to stop all active sessions --
  base#.session_level(0, false);
  --
  if base#.session_level = 0 then
    install#.log#info('Successfully set session-level "0".');
  else
    install#.log#error('Failed to set session-level "0"!');
  end if;

  -- Change EPM keystore password (master key) to password without database characteristics
calculation factors --
  begin
    epm_par_admin#.set_master_key_calc_fact_list(
      i_master_key_calc_fact_list => null
     ,i_startup_pwd               => null
     ,i_opt                       => epm_par_admin#.c_update_master_key
    );
    --
    install#.log#info('Successfully changed EPM keystore master key (factors "'||epm_par_
admin#.master_key_calc_fact_list||'").');
  exception
    when others then
      install#.log#error('EPM keystore master key change failed: '||sqlerrm);
  end;

  -- Verify EPM keystore access with new master key before migration --
  if epm_par_admin#.is_keystore_ready then
    install#.log#info('EPM keystore accessible with new master key.');
  else
    install#.log#error('EPM keystore NOT accessible with new master key!');
  end if;
end;
/
```

3. Migrate your database to the Oracle Multitenant architecture

   Follow the steps in the previous sections (see "Introduction" on page 5).

4. Migrate your existing EPM keystore with the desired database characteristics (`db_id`, `host_id`, `instance_id`) that are used for the master key calculation.

   Change the EPM keystore password to a password that uses again the database characteristics (`db_id`, `host_id` , `instance_id`.). To do this, set the input parameter `i_master_key_calc_fact_list` in a call to the procedure `epm_par_admin#.set_master_key_calc_fact_list`, according to your bank-specific needs.

For the bank-specific calculation factors, use the values from the `avq.epm` base parameter's `master_key_calc_fact_list` item before the migration. You made a note of them in step 1 above.

For example:

```
exec dbms_session.reset_package;
begin
  -- Start session --
  session#.open_session;

  -- Verify EPM keystore access after migration --
  if epm_par_admin#.is_keystore_ready then
    install#.log#info('EPM keystore accessible with current master key.');
  else
    install#.log#error('EPM keystore NOT accessible with current master key!');
  end if;

  -- Change EPM keystore password (master key) to password with customer specific database
characteristics calculation factors --
  begin
    epm_par_admin#.set_master_key_calc_fact_list(
      i_master_key_calc_fact_list => '<to be set to the bank specific calculation factors>'
     ,i_startup_pwd               => null
     ,i_opt                       => epm_par_admin#.c_update_master_key
    );
    --
    install#.log#info('Successfully changed EPM keystore master key (factors "'||epm_par_
admin#.master_key_calc_fact_list||'").');
  exception
    when others then
      install#.log#error('EPM keystore master key change failed: '||sqlerrm);
  end;

  -- Verify EPM keystore access with new master key after migration --
  if epm_par_admin#.is_keystore_ready then
    install#.log#info('EPM keystore accessible with new master key.');
  else
    install#.log#error('EPM keystore NOT accessible with new master key!');
  end if;
end;
/
```