

CED Install Guide (v3.2)

By Theo Larrieu

| | |
|--|----|
| Introduction | 2 |
| History | 2 |
| Overview | 2 |
| Database Creation..... | 3 |
| Repository | 3 |
| Define CED Roles..... | 3 |
| Create CED Users | 3 |
| Create CED Schema Objects..... | 4 |
| Multiple Schemas..... | 4 |
| Install the CED_ADMIN package | 6 |
| CED_ADMIN Initialization | 7 |
| READ_CED schemas | 7 |
| Explicit Permissions for Owner Schema..... | 8 |
| Explicit Permissions for DEVL Schema | 9 |
| CED Event Server Configuration..... | 10 |
| .so file | 10 |
| Extproc.ora..... | 10 |
| Wrapper function..... | 10 |
| History Triggers | 10 |
| Logoff Triggers | 11 |
| Enable Versioning | 13 |
| Create Workspaces | 15 |

Introduction

History

| | | |
|-----|------------|---|
| 3.0 | 2012-03-14 | Initial document revision corresponding to the 3.0 Schema |
| 3.1 | 2016-03-17 | <p>Document separated in two.</p> <ul style="list-style-type: none">• Install Guide (this doc)• Admin Guide <p>Install Guide updated to reflect</p> <ul style="list-style-type: none">• Segregation of history into separate database instance• The automated history checkpoint process• The creation of the v_zone_s and v_cmpnt_s views |
| 3.2 | 2016-07-19 | Update Event Server section to reflect changes in ced_notify 3.0 which brought support for multiple *EDs. |

Overview

This document addresses topics related to the high-level organization and administration of the CEBAF Element Database (CED) infrastructure. Topics covered include how to create the database and how to move data between workspaces and schemas. This document will evolve as our tools and methods change over time, so be sure to check the CED wiki

(https://accwiki.acc.jlab.org/pub/AHLA/CED/CED_Administrator.pdf) to make sure you are referencing the most up-to-date version.

Database Creation

Repository

The database creation sql scripts mentioned in this document as well as the Erwin Model and the master for this document itself are all stored in the ACE git repository. Use the git clone command as shown below to obtain the source.

```
%> cd ~
%> git clone ssh://dev100.acc.jlab.org/usr/devsite/git/ceddb.git
%> cd ceddb
%> ls
doc  erwin  sql
```

Define CED Roles

A prerequisite to creating of a CED schema is to create the READ_CED and OWN_CED roles. When it is created, each CED schema will grant SELECT privilege on all its tables to the READ_CED role and will grant ALL privilege to the OWN_CED role.

Because each CED schema is granted the READ_CED role by default at creation time, each CED schema has the ability to view, but not modify the data from other CED schemas in the same database instance. The OWN_CED role when granted permits that user (e.g. CED3_OWNER) also to insert, update, and delete data across all CED schemas.

```
# Assuming source was checked out as shown previously
%> cd ~/ceddb/sql
%> setenv ORACLE_SID #SID#
%> sqlplus '/ as sysdba'

SQL> @createCEDRoles.sql
'*****'
'* NOTE: This script must be run as sysdba *'
'*****'
SQL> exit
%>
```

Create CED Users

A CED Database user is created by connecting to the oracle database as the SYSDBA and running the script createUser.sql script as the example in the textbox below illustrates. The script must be run four times, once for each of the CED schemas (ced3_devl, ced3_ops, ced3_hist, ced3_owner).

```

%> cd ~/ceddb/sql
%> setenv ORACLE_SID #SID#
%> sqlplus '/ as sysdba'

SQL> @createCEDUser.sql
'*****'
'* NOTE: This script must be run as sysdba *'
'*****'
Enter the schema owner to create >ced3_owner
Enter the schema password to set >*****

User created.
Grant succeeded.
(...)
Grant succeeded.
Grant succeeded.
SQL> exit
%>

```

Create CED Schema Objects

Multiple Schemas

While it would be possible to run the CED using a single versioned schema, we have opted to spread the functionality across four schemas and two database instances as illustrated in both Table 1 and Figure 1 below.

Table 1

| Schema | Versioned? | Role | Description |
|--|------------|----------|---|
| CED3_OWNER@CEDDB01 CED3_OWNER@CEDHIST | N | OWN_CED | Used by API to connect. Has privilege to modify data in all CED schemas. Owns the CED_ADMIN PL/SQL package. |
| CED3_OPS@CEDDB01 | N | READ_CED | Current Operational data. For performance reasons, is not versioned |
| CED3_DEVL@CEDDB01* | Y | READ_CED | Contains Multiple Developmental Workspaces. "LIVE" intended to be a mirror of CED3_OPS. |
| CED3_HIST@CEDHIST | Y | READ_CED | Contains Single LIVE workspace to preserve a history of CED3_OPS via creation of regular savepoints. |

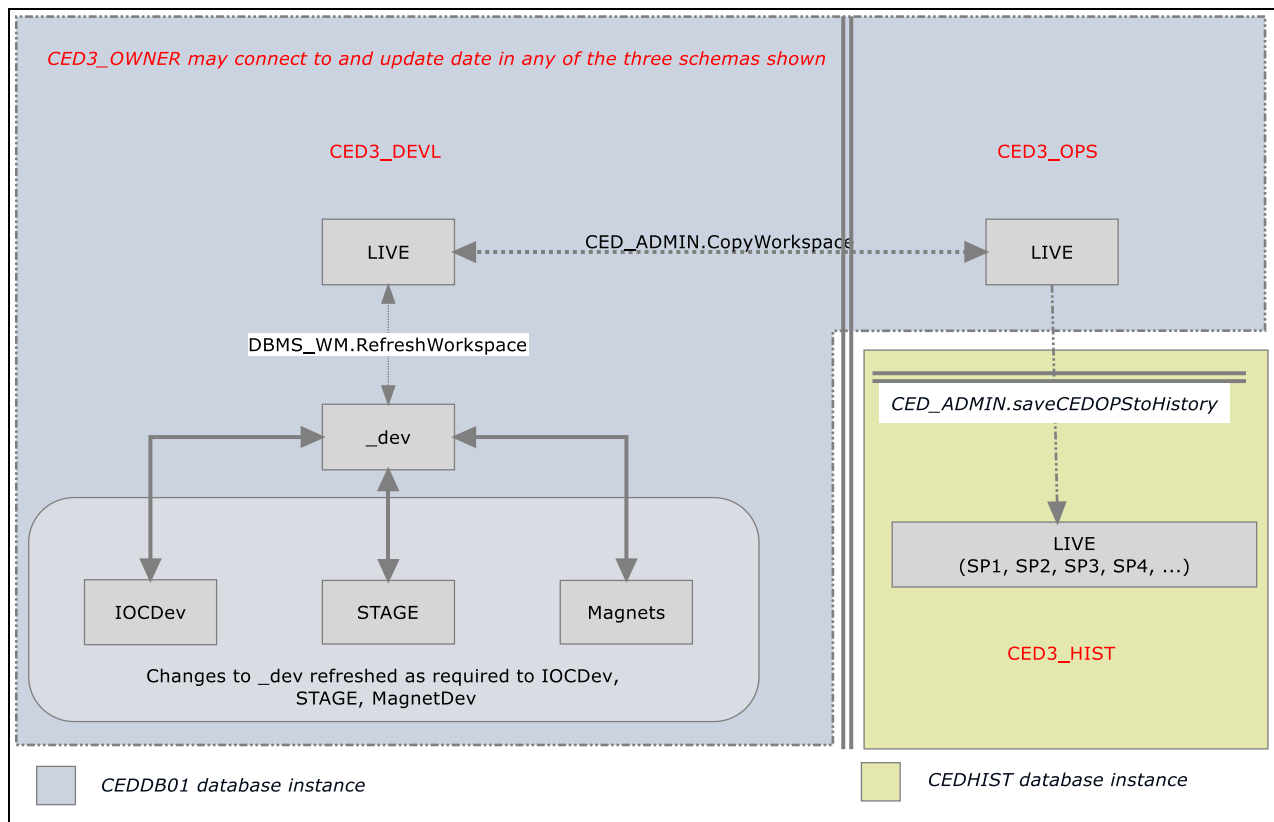


Figure 1 - Illustrates the schemas and database instances in which they are used.

The bundle of four schemas (CED3_OWNER, CED3_OPS, CED3_DEV, CED3_HIST) will comprise a single production CED installation. When no workspace or savepoint name is specified to make a CED connection, the CED API Library will by default access the contents of the CED3_OPS@CEDDB01 schema. When a valid workspace name is specified, data will be retrieved from CED_DEV@CEDDB01 and when a valid savepoint name is specified, data will be retrieved from CED3_HIST@CEDHIST¹.

To define the tables, views, sequences and other objects in a CED schema, use SQLPLUS to connect to Oracle as the user who owns the schema and execute the create createSchema3.sql script as illustrated in the example below:

¹ These connections are enacted in the CED API config file /cs/certified/apps/cedlib/cfg/cedlib5.cfg. The ability to connect to a separate database instance for history requires cedlib 4.0 or newer -- prior to that release, the history schema was stored in the ceddb01 instance.

```
# Replace @ceddb with the actual SID of the CED database.
%> sqlplus ced3_owner@ceddb

SQL> @/cs/dvlhome/apps/c/CED/dvl/src/createSchema3.sql

SQL> exit


%> sqlplus ced3_ops@ceddb

SQL> @/cs/dvlhome/apps/c/CED/dvl/src/createSchema3.sql

SQL> exit

..
```

The createSchema3.sql script creates empty tables, sequences, and views. In a multiple-schema environment, it must be executed within each schema by the owner of that schema.

 The createSchema3.sql is generated programmatically using the ERWin Entity-Relationship Modeling software and should not be hand-edited (hand-edits will be lost the next time the script is generated). Changes should instead be made to the ERWin model and then a new createSchema3.sql script generated from there. The ERWin Model file is stored in the CED V3.erwin file located in the erwin directory of the git repository.

Install the CED_ADMIN package

A package of stored procedures named CED_ADMIN facilitates the creation and maintenance of multiple CED schemas. It should be installed in and owned by the CED3_OWNER schema, however all schemas with the READ_CED role should be granted execute permission on it and allowed to access it via a public synonym.

```

%> cd ~/ceddb/sql
%> sqlplus ced3_owner@ceddb01

SQL>@CED_ADMIN.pks
SQL>/

Package specification created.

SQL>@CED_ADMIN.pkb
SQL>/

Package body created.

SQL>grant execute on CED_ADMIN to READ_CED;

Grant succeeded.

SQL>create public synonym CED_ADMIN for CED_ADMIN;

Synonym created.

SQL>grant execute on CED_ADMIN to WRITE_CED;

Grant succeeded.

/* We also need to grant permissions explicitly to certain users because
   Role-based permissions are not honored in some contexts such as when a
   Job runs scheduled via DBMS_JOB. */

SQL>grant execute on CED_ADMIN to CED3_DEVL;

Grant succeeded.

SQL>grant execute on CED_ADMIN to CED3_HIST;

Grant succeeded.

SQL>grant execute on CED_ADMIN to CED3_OPS;

Grant succeeded.

/* We just need one master copy of the Staff materialized view */

ced3_owner@CEDDB01> grant select on staff to READ_CED;

Grant succeeded.

SQL> exit

```

CED_ADMIN Initialization

READ_CED schemas

After creating non-owner schemas (e.g those with READ_CED, not OWN_CED role), the following minor initialization should be performed.

```

%> sqlplus ced3_dev1@ceddb01

SQL> create synonym CED_ADMIN for ced3_owner.CED_ADMIN;

Synonym created.

SQL> exec CED_ADMIN.grantPermissions;

PL/SQL procedure successfully completed.

SQL> exec CED_ADMIN.createStageTables;

PL/SQL procedure successfully completed.

SQL> drop table web_auth;

Table dropped.

SQL> create synonym web_auth for ced3_owner.web_auth;

Synonym created.

SQL> drop table admin_lock;

Table dropped.

SQL> create synonym admin_lock for ced3_owner.admin_lock;

Synonym created.

SQL> drop materialized view staff;

Materialized view dropped.

SQL> create synonym staff for ced3_owner.staff;

Synonym created.

SQL> exit

```

Explicit Permissions for Owner Schema

Permissions granted via role are not available in certain circumstances such as when a stored procedure is executed via DBMS_JOB. Therefore in addition to running the createCEDRoles.sql script as described earlier, it is necessary for the DBA to grant some permissions explicitly to the master Schema (i.e. CED3_OWNER)

```

# On the database server
%> sqlplus '/ as sysdba'

SQL> exec DBMS_WM.GrantSystemPriv('ACCESS_ANY_WORKSPACE', 'CED3_OWNER', 'NO');

PL/SQL procedure successfully completed.

SQL> exec dbms_wm.grantsystempriv('MERGE_ANY_WORKSPACE', 'CED3_OWNER', 'NO');

PL/SQL procedure successfully completed.

SQL>exit

```


Explicit Permissions for DEVL Schema

Permissions granted via role are not available in certain circumstances such as when a stored procedure is executed via DBMS_JOB. Therefore in addition to running the createCEDRoles.sql script as described earlier, it is necessary for the DBA to grant some permissions explicitly to the development Schema (i.e. CED3_DEVL)

```
# On the database server
%> sqlplus '/ as sysdba'

SQL> exec DBMS_WM.GrantSystemPriv('ACCESS_ANY_WORKSPACE', 'CED3_DEVL', 'NO');

PL/SQL procedure successfully completed.

SQL> exec dbms_wm.grantsystempriv('MERGE_ANY_WORKSPACE', 'CED3_DEVL', 'NO');

PL/SQL procedure successfully completed.

SQL>exit
```

CED Event Server Configuration

The CED event server is the mechanism that allows database actions to trigger external events. This section documents configuring Oracle to notify the event server of events.

.so file

The ced_notify.so file that can be found in /cs/certified/apps/cedes/PRO/bin/rhel-6-x86_64 must be copied to \$ORACLE_HOME/lib on the database server.

Extproc.ora

Beginning with Oracle 11.2.0, the following directives are placed in the file \$ORACLE_HOME/hs/admin/extproc.ora instead of configuring listener.ora as was done in earlier versions of Oracle.

```
SET EXTPROC_DLLS=ANY
```

Wrapper function

```
-- As Sysdba
SQL> create library ced_notify_lib is
        '/opt/oracle/sesrv/11.2.0/db_3/lib/ced_notify.so';
/

-- As ced3_owner
create or replace function ced_notify(deployment IN char, ced_event IN char) return
binary_integer
as external
    name "ced_notify"
    library ced_notify_lib
    language C
    parameters (deployment string, ced_event string);
/
```

History Triggers

In order for history triggers in the ced3_devl and ced3_ops schemas to work properly, these schemas must be granted permissions directly (not just via role) to the hist_* tables in the ced3_owner schema.

```

ced3_owner > grant insert, select on hist_cmpnt to ced3_ops, ced3_dev1;

Grant succeeded.

ced3_owner> grant insert, select on hist_cmpnt_prop to ced3_ops, ced3_dev1;

Grant succeeded.

ced3_owner> grant insert, select on hist_cmpnt_prop_val to ced3_ops, ced3_dev1;

Grant succeeded.

ced3_owner> grant select on event_id to ced3_ops, ced3_dev1;

Grant succeeded.

ced3_owner> grant execute on ced_notify to ced3_ops, ced3_dev1;

Grant succeeded.

ced3_owner> grant execute on CED_ADMIN to ced3_ops, ced3_dev1;

Grant succeeded.

```

After the afore-mentioned permissions have been granted, execute the script hist_triggers.sql as ced3_ops. The script can also be excuted for ced3_dev1 if versioning has not yet been enabled. If versioning has been enabled, then the triggers must be added using the Oracle Workspace Manager beginDDL/comittDDL process.

Logoff Triggers

The trigger below is responsible for calling the event server at the end of a session. It must be installed in each schema.

```

/*
-- Check for invalid triggers
select object_name, object_type, status
from dba_objects
where object_name = 'CED_LOGOFF';
*/

-- as each ced user
create or replace trigger ced_logoff
before logoff on schema
begin
    insert into admin_log (log_id, logdate, logmsg) values (log_id.nextval, sysdate,
    'Firing Logoff trigger');
    CED_ADMIN.endHistEvent(true);
    -- insert into admin_log (log_id, logdate, logmsg) values
    (ced3_owner.log_id.nextval, sysdate, 'logoff 2 '||to_char(retVal));

end;
/

```


Enable Versioning

Two of the CED schemas (CED3_DEVL@CEDDB01 and CED3_HIST@CEDHIST) will be version-enabled using the Oracle Workspace Manager toolkit. This means that the tables in these schemas will be capable of storing multiple versions of each data row in their tables. Rows may be versioned using either workspaces or savepoints or both. (For further discussion of workspace and savepoints, see the Oracle Workspace Manager manual available at [http://devweb/oradocs/appdev.111/b28396/toc.htm](http://devweb.oradocs.appdev.111/b28396/toc.htm)).

In our usage, the CED3_DEVL is intended to be versioned using workspaces, while the CED3_HIST tables will be versioned using savepoints. In both cases, the enableCEDVersioning script is run to version-enable the tables in each schema.

```
%> sqlplus ced3_devl@ceddb01

SQL>@/cs/dvlhome/apps/c/CED/dvl/src/enableCEDVersioning.sql
SQL> exit

%> sqlplus ced3_hist@ceddhist

SQL>@/cs/dvlhome/apps/c/CED/dvl/src/enableCEDVersioning.sql
SQL> exit
```

After enableCEDVersioning completes, the original tables will have been renamed with an _LT suffix and replaced with a view of the original name as can be seen in Figure 2.

Select sqlplus.exe - Shortcut

ced3_dev1@CEDDB01> select * from cat;

| TABLE_NAME | TABLE_TYPE |
|--------------------|------------|
| ADMIN_LOG | TABLE |
| CATEGORY_ID | SEQUENCE |
| CATEGORY_SETS | VIEW |
| CATEGORY_SETS_AUX | TABLE |
| CATEGORY_SETS_BASE | VIEW |
| CATEGORY_SETS_BPKC | VIEW |
| CATEGORY_SETS_CONF | VIEW |
| CATEGORY_SETS_CONS | VIEW |
| CATEGORY_SETS_DIFF | VIEW |
| CATEGORY_SETS_HIST | VIEW |
| CATEGORY_SETS_LCK | TABLE |
| CATEGORY_SETS_LOCK | VIEW |
| CATEGORY_SETS_LT | TABLE |
| CATEGORY_SETS_MW | VIEW |
| CATEGORY_SETS_PKC | VIEW |
| CATEGORY_SETS_PKD | VIEW |
| CATEGORY_SETS_PKDB | VIEW |
| CATEGORY_SETS_PKDC | VIEW |
| CMPNT | VIEW |
| CMPNT_AUX | TABLE |
| CMPNT_BASE | VIEW |
| CMPNT_BPKC | VIEW |
| CMPNT_CONF | VIEW |
| CMPNT_CONS | VIEW |
| CMPNT_DIFF | VIEW |
| CMPNT_HIST | VIEW |
| CMPNT_ID | SEQUENCE |
| CMPNT_LCK | TABLE |
| CMPNT_LOCK | VIEW |
| CMPNT_LT | TABLE |
| CMPNT_MW | VIEW |
| CMPNT_OWNERS | VIEW |
| CMPNT_OWNERS_AUX | TABLE |
| CMPNT_OWNERS_BASE | VIEW |
| CMPNT_OWNERS_BPKC | VIEW |
| CMPNT_OWNERS_CONF | VIEW |
| CMPNT_OWNERS_CONS | VIEW |

After Versioning, the original cmpnt table was moved to cmpnt_lt where version and timestamp columns were added. In place of the original table is now a view that shows only the rows relevant to the current workspace/savepoint

Figure 2

Create Workspaces

After a schema has been version-enabled, it is possible to create workspaces in it using procedures made available through Oracle Workspace Manger. The following example shows examples of the commands used to workspaces.

```
%> sqlplus ced3_dev1@ceddb01

SQL> EXEC DBMS_WM.gotoworkspace('LIVE');

/* _dev will be "hidden" parent to development workspaces like IOCDev */
SQL> EXEC DBMS_WM.createWorkspace('_dev');

SQL> EXEC DBMS_WM.gotoWorkspace('_dev');
SQL> EXEC DBMS_WM.createWorkspace('IOCDev', 'Workspace for maintaining IOC data');
SQL> EXEC DBMS_WM.createWorkspace('MagnetDev', 'Workspace for adding new Magnets and
related properties');
SQL> EXEC DBMS_WM.createWorkspace('STAGE', 'Workspace for AHLA group to add and edit
bealmine elements and merge Elegant Decks');

SQL> exit
```

- ☛ Note that when workspaces are created, they are owned by the user who created them. The API relies on this owner information to determine which schema to use. While it may be correct to create the workspaces as user CED3_OWNER from an Oracle perspective, it will cause the API problems because it will not know to switch to schema CED3_DEVL before performing queries.