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 | Document separated in two.   * Install Guide (this doc) * Admin Guide   Install Guide updated to reflect   * 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://devl00.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\_DEVL, 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\_DEVL@CEDDB01 and when a valid savepoint name is specified, data will be retrieved from CED3\_HIST@CEDHIST[[1]](#footnote-1).

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:

# 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\_devl@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\_devl;

Grant succeeded.

ced3\_owner> grant insert, select on hist\_cmpnt\_prop to ced3\_ops, ced3\_devl;

Grant succeeded.

ced3\_owner> grant insert, select on hist\_cmpnt\_prop\_val to ced3\_ops, ced3\_devl;

Grant succeeded.

ced3\_owner> grant select on event\_id to ced3\_ops, ced3\_devl;

Grant succeeded.

ced3\_owner> grant execute on ced\_notify to ced3\_ops, ced3\_devl;

Grant succeeded.

ced3\_owner> grant execute on CED\_ADMIN to ced3\_ops, ced3\_devl;

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\_devl 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>).

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.

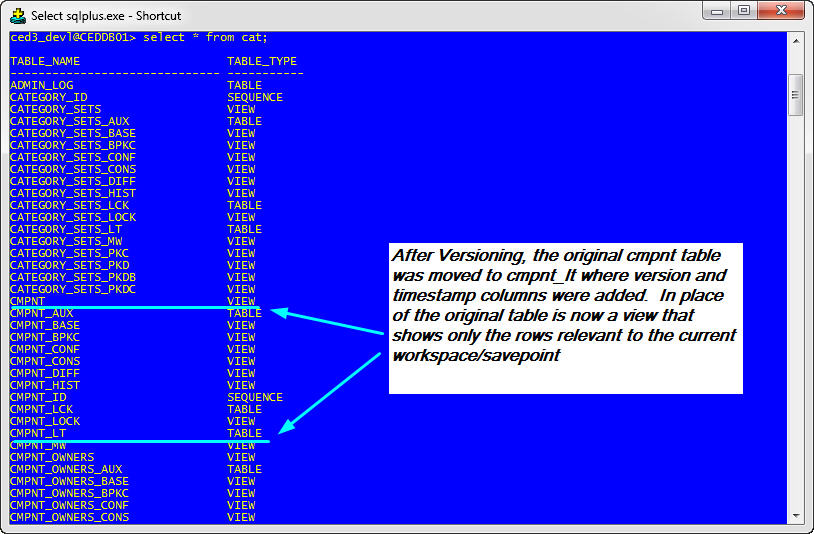


Figure 2

## 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\_devl@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.

💣

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. [↑](#footnote-ref-1)