CMDB Scripts developed by Softura

1. cmdbdesign.sql
2. insertscript.sql
3. masterscript.py
4. cmdbinsert.py
5. Edrstatus.py
6. release-insert.py
7. Jenkinsupdate.py

CMDB connection details,

Connect to 3.125.182.23 server, there you can use the root user and root password to connect,

mysql -u root -proot;

show databases;

use cmdb\_design;

show tables;

select \* from cmdb\_design.repo;

select \* from cmdb\_design.repocommits;

1.cmdbdesign.sql:

This script will create the cmdb design database. Which will have the following tables.

1. release: This table will have Release information including release branch name, date and time and some static information like URLs for the Jenkins, GITHUB, EDR , Confluence and etc which will pull this information from the insertscript.sql. Release ID is a Primary Key in this table which will have a unique number.
2. repo: This table will have the information pulling from the CQ Excel sheet which will have the info of repo names, latest commit number, Release Type and Jenkins status
3. sco: This table will have the information of SCO#
4. defect: This table will have information of the SCO and SRP.
5. repocommits: This table will have the information of the all the repos including the repos with the same name but multiple commits in the CQ Defect excel sheet, EDR number and it’s status.
6. AuditLogs: This table will have the information of Log level info, log message and log time stamp.
7. Properties: This table will have the static information like property name and property value (e.g., Build Machine url, Toll Gate url, updated by and update time) pull from the insertscript.sql file.

Find the current CMDB Design in the attached picture.

A screenshot of a cell phone

Description automatically generated

2. insertscript.sql: This sql script will have the information of static information of Build Machine url, Toll Gate url, if it is changes you can edit this script. It’s fairly simple.

3. masterscript.py: Master script is the main script and the only script you run for each release. First it will look for the QueryResult.xls file in the folder then it will convert the xls file into csv to read the information of the defects. It will look for the DEF\_Changes list to confirm the defect belongs to git or MD.

If it is GIT, it will check the Product\_Discriptors to verify the defect belongs to MCA Platform or MCA Applications. Once the project is verified then it will look for the repo name from Product\_Descriptor\_Level3, then after it will check for the repo type. We have four kind of repos under github platform organization, those are Service based repo, COTS repos, Docker build image repos and library repos. We only need release branch for service based repo, for remaining repos we need to merge develop with the master branch. So script will identify the type of repo script itself will creates the release branch for Service based repos if not it will merge the develop branch with the master. Now it will take the repo name from Product\_Discriptors\_Level3 field then based on the version number you have given in the Def\_ChangesList it will create the release branch also pre-release tag using GITHUB APIs.

If it is MD item it will tag the MD project, takes the version number from the Def\_Changeslist and look for the Project name from Product\_Discriptors. We have four different projects in MD, those are MCA Platform SW, MCA Systems, MCA IV&V and MCA Applications. These four project has the project ID. So based on the project mentioned in the Product\_Discriptors, it will take the ID from the Projected.py python script then it will tag the given version by using MD APIs. with the same name you used to create the release branch.

(We must follow the work instructions given by the CM team to use this scripts, check the work instructions document [mca-cfg-cqwi\_defects.docx](https://devcloud.swcoe.ge.com/devspace/download/attachments/1532307676/mca-cfg-cqwi_defects.docx?api=v2) here <https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?pageId=1532307676> ).

Usage: python masterscript.py

Then it will ask you to enter the release branch name : IVV-PLT-2020.0317.1200 (Follow this pattern IVV-PLT-YEAR.DATE.TIME).

4. cmdbinsert.py: When you run the masterscript.py, it will also run two scripts one of them is cmdbinsert.py. In the cmdbinsert.py script, it will read the initial values (like Repo name, Latest commit, Release type and Jenkins status, Jenkins status for git items are initially Jenkins has stated) from the QueryResult.xls file and insert into the Repo table and repo commits table. Cmdbinsert.py scripts runs the one more python script called edrstatus.py.

5. edrstatus.py: This script will run with the cmdbinsert.py, we can get the edr status with this and insert the edr status into the repocommits table. The edr number input will be taken from xls file given in the DSRVW # field of CQ.

6. release-insert.py: As I have mentioned in the earlier, when you run the master script it will also runs the release-insert.py script. This script will insert the values into the release table.

7. jenkinsupdate.py: This script supposed to run with the Jenkinsfile, the main intention of this script is once the creation of release branch from git it will trigger a build and it takes some time after the successful build it will push the docker image to the DTR. So here we need to know the Jenkins status and DTR information as well. This script will take those arguments from the Jenkins file and it will insert those values into the repo table. As of now, we have not integrated this script with the Jenkinsfile. We can also run this script manually but it needs the branch name, repo name, DTR and Jenkins status as arguments. Here is the instructions and example to run this script.

Python jenkinsupdate.py BRANCH\_NAME REPO DTR JENKINS\_STATUS

E.g., jenkinsupdate.py IVV-PLT-2020.0317.1200 gets-filetransferservice IVV-PLT-2020.0317.1200.1 Success

Check the following flowchart which explains the flow of the scripts.

A close up of a map

Description automatically generated