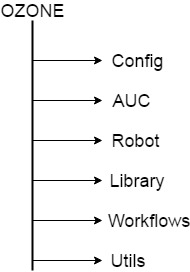
**Automation Framework**

**Folder Structure**

****

**OZONE:**This is main folder for Ozone test automation

**Config:**This folder contains input parameters which will be required for ozone test execution. Preferably yaml files.

**AUC (Associate use cases):**All test functionality as AUCs going to keep in this folder.

**Robot:**  This folder contains the Created Robot Suite files to test the Ozone framework.

**Library**: This folder contains the Library files to interact with the endpoints i.e. Ozone, vCenter

**Utils:**  This folder contain any required utils for the framework like ssh client, yaml parser.

**Workflows:**  This will contain the workflow for all the test methods.

**Pulling the Automation Code**

First clone the repo and pull the latest code in develop branch

Create new directory under C drive as I am following C:\ozone\_taf

cd C:\ ozone\_taf

Clone the repo

git clone ssh://git@pie6.rtp.lab.emc.com/ehcinstall/ozonetestframework.git

Check out the branch

cd ozonetestframework

git checkout develop

Pull the code

git pull origin develop

# Updating System Variables in windows

PYTHONPATH

C:\Python27;C:\Python27\Lib\site-packages;

C:\Users\volups\Downloads\ozonetestframeworkdevelop@32307aaeaa5;

C:\Users\volups\Downloads\ozonetestframework-develop@32307aaeaa5\ozone\library;

Here C:\Users\volups\Downloads\ozonetestframeworkdevelop@32307aaeaa5 is the location of

Ozone test framework code

PATH should be extended as

C:\Python27;C:\Python27\Scripts;

Packages to be installed:

paramiko

pyyaml

xlutils

requests

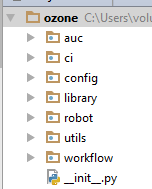
robotframework

pyVmomi

pyVim

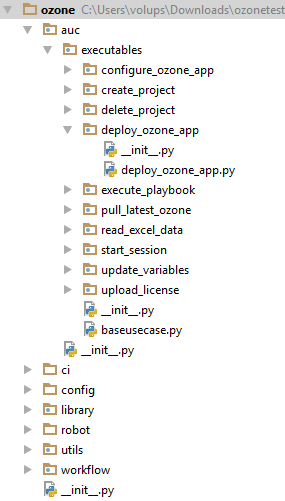
cifsutil

Once we done with pulling the latest automation code and the folder structure looks like this below.



In this structure, we have to add our each functionality as AUCs in this folder under executables,

Here I am showing an example deploy\_ozone\_app



Now we start with the Automation. I specified below the steps needed for automation to create your user story

1. AUC creation/ Check for already existing AUC
2. Config file Creation/Update
3. Update workflow file with AUC Created
4. Robot file Creation

**AUC Creation**

First check any AUC is there with the functionality otherwise create AUC.

To create any new user story as AUC these are the steps to be followed

1. First create a folder under auc\executables on with your auc name

Eg. Created deploy\_ozone\_app folder under auc\executables

This folder contains 2 files

1. deploy\_ozone\_app.py
2. \_\_init\_\_.py

I am attaching deploy\_ozone\_app.py as below. Copy the entire file and change as for

Your user story

*"""  
Python file to Deploy Ozone Machine  
"""****from*** *ozone.auc.executables.baseusecase* ***import*** *BaseUseCase****from*** *robot.api* ***import*** *logger****class*** *DeployOzoneVM(BaseUseCase):  
 """  
 Deploys the Ozone Machine  
 """* ***def*** *deploy\_ozone\_machine(self):  
 """"  
 To deploy ozone machine  
 """  
 self.deployment\_status = self.deployment\_session.deployOzonevApp(self.vcenter\_details,  
 self.ova\_details, self.timeout)* ***def*** *runTest(self):  
 # Actual method call to run the Test  
 self.deploy\_ozone\_machine()* ***def*** *\_validate\_input\_args(self, \*\*kwargs):  
 # Validating and Assigning the input kwargs  
 self.in\_section = kwargs.get(****'section'****)* ***def*** *\_validate\_context(self):  
 '''  
 # Validating and mapping the variables taken from the context  
 '''* ***if*** *self.ctx\_in:  
  
 section = getattr(self.ctx\_in, self.in\_section)* ***or*** *self.ctx\_in* ***assert*** *section.deployment\_session,* ***'Deployment session is None'*** *self.deployment\_session = section.deployment\_session  
 self.vcenter\_details = {}  
 self.vm\_details = {}  
 self.ova\_details = {}  
 self.deploy\_ozone = section.deploy\_ozone\_vApp* ***assert*** *hasattr(self.deploy\_ozone,* ***'vcenter\_details'****),* ***'vcenter details not provided'*** *self.vcenter\_details[****'vCenterIPAddress'****] = self.deploy\_ozone.vcenter\_details.vCenterIPAddress  
 self.vcenter\_details[****'vCenterUsername'****] = self.deploy\_ozone.vcenter\_details.vCenterUsername  
 self.vcenter\_details[****'vCenterPassword'****] = self.deploy\_ozone.vcenter\_details.vCenterPassword  
 self.vcenter\_details[****'datastore'****] = self.deploy\_ozone.vcenter\_details.datastore  
 self.vcenter\_details[****'cluster'****] = self.deploy\_ozone.vcenter\_details.cluster  
 self.vcenter\_details[****'hostName'****] = self.deploy\_ozone.vcenter\_details.hostName  
 self.vcenter\_details[****'vCenterPort'****] = self.deploy\_ozone.vcenter\_details.vCenterPort  
 self.vcenter\_details[****'dataCenter'****] = self.deploy\_ozone.vcenter\_details.dataCenter  
 self.vcenter\_details[****'resourcePool'****] = self.deploy\_ozone.vcenter\_details.resourcePool* ***assert*** *hasattr(self.deploy\_ozone,* ***'ova\_details'****),* ***'ova details not provided'*** *self.ova\_details[****'vmName'****] = self.deploy\_ozone.ova\_details.vmName  
 self.ova\_details[****'network'****] = self.deploy\_ozone.ova\_details.network  
 self.ova\_details[****'ovaPath'****] = self.deploy\_ozone.ova\_details.ovaPath  
 self.ova\_details[****'masterIP'****] = self.deploy\_ozone.ova\_details.masterIP  
 self.ova\_details[****'masterFQDN'****] = self.deploy\_ozone.ova\_details.masterFQDN  
 self.ova\_details[****'workerIP'****] = self.deploy\_ozone.ova\_details.workerIP  
 self.ova\_details[****'workerFQDN'****] = self.deploy\_ozone.ova\_details.workerFQDN  
 self.ova\_details[****'gateway'****] = self.deploy\_ozone.ova\_details.gateway  
 self.ova\_details[****'netmask'****] = self.deploy\_ozone.ova\_details.netmask  
 self.ova\_details[****'dns'****] = self.deploy\_ozone.ova\_details.dns* ***assert*** *hasattr(self.deploy\_ozone,* ***'timeout'****),* ***'timeout not provided'*** *self.timeout = self.deploy\_ozone.timeout* ***def*** *\_finalize\_context(self):* ***if*** *self.deployment\_status:  
 logger.info(****'Deploy Ozone VM Successful'****, False, True)* ***else****:  
 logger.error(****'Deploy Ozone VM Failed'****)* ***raise*** *AssertionError,* ***'Deploy Ozone VM Failed'***

Here are specifying the different sections in the AUC.

1. imports section : Imports needed from developer code.

***from*** *ozone.auc.executables.baseusecase* ***import*** *BaseUseCase****from*** *robot.api* ***import*** *logger*

1. Class Name : Name the class with appropriate name as per the functionality

Eg: **class** DeployOzoneVM(BaseUseCase)

Optional section*:* validate input args section

It is used to provide additional inputs to the AUCs from workflow

***Eg: def*** *\_validate\_input\_args(self, \*\*kwargs):  
 # Validating and Assigning the input kwargs  
 self.in\_section = kwargs.get(****'section'****)*

1. Validate context section: where we map the inputs from the config file to the inputs needed for the script.

Eg:

***def*** *\_validate\_context(self):  
 '''  
 # Validating and mapping the variables taken from the context  
 '''* ***if*** *self.ctx\_in:  
  
 section = getattr(self.ctx\_in, self.in\_section)* ***or*** *self.ctx\_in* ***assert*** *section.deployment\_session,* ***'Deployment session is None'*** *self.deployment\_session = section.deployment\_session  
 self.vcenter\_details = {}  
 self.vm\_details = {}  
 self.ova\_details = {}  
 self.deploy\_ozone = section.deploy\_ozone\_vApp* ***assert*** *hasattr(self.deploy\_ozone,* ***'vcenter\_details'****),* ***'vcenter details not provided'*** *self.vcenter\_details[****'vCenterIPAddress'****] = self.deploy\_ozone.vcenter\_details.vCenterIPAddress  
 self.vcenter\_details[****'vCenterUsername'****] = self.deploy\_ozone.vcenter\_details.vCenterUsername  
 self.vcenter\_details[****'vCenterPassword'****] = self.deploy\_ozone.vcenter\_details.vCenterPassword  
 self.vcenter\_details[****'datastore'****] = self.deploy\_ozone.vcenter\_details.datastore  
 self.vcenter\_details[****'cluster'****] = self.deploy\_ozone.vcenter\_details.cluster  
 self.vcenter\_details[****'hostName'****] = self.deploy\_ozone.vcenter\_details.hostName  
 self.vcenter\_details[****'vCenterPort'****] = self.deploy\_ozone.vcenter\_details.vCenterPort  
 self.vcenter\_details[****'dataCenter'****] = self.deploy\_ozone.vcenter\_details.dataCenter  
 self.vcenter\_details[****'resourcePool'****] = self.deploy\_ozone.vcenter\_details.resourcePool* ***assert*** *hasattr(self.deploy\_ozone,* ***'ova\_details'****),* ***'ova details not provided'*** *self.ova\_details[****'vmName'****] = self.deploy\_ozone.ova\_details.vmName  
 self.ova\_details[****'network'****] = self.deploy\_ozone.ova\_details.network  
 self.ova\_details[****'ovaPath'****] = self.deploy\_ozone.ova\_details.ovaPath  
 self.ova\_details[****'masterIP'****] = self.deploy\_ozone.ova\_details.masterIP  
 self.ova\_details[****'masterFQDN'****] = self.deploy\_ozone.ova\_details.masterFQDN  
 self.ova\_details[****'workerIP'****] = self.deploy\_ozone.ova\_details.workerIP  
 self.ova\_details[****'workerFQDN'****] = self.deploy\_ozone.ova\_details.workerFQDN  
 self.ova\_details[****'gateway'****] = self.deploy\_ozone.ova\_details.gateway  
 self.ova\_details[****'netmask'****] = self.deploy\_ozone.ova\_details.netmask  
 self.ova\_details[****'dns'****] = self.deploy\_ozone.ova\_details.dns* ***assert*** *hasattr(self.deploy\_ozone,* ***'timeout'****),* ***'timeout not provided'*** *self.timeout = self.deploy\_ozone.timeout*

1. Call the actual function in the developers code.

Change the method name in runTest to appropriate name and change the above method to the same name and call the operation inside it what you need to perform.

E.g.:

***def*** *deploy\_ozone\_machine(self):  
 """"  
 To deploy ozone machine  
 """  
 self.deployment\_status = self.deployment\_session.deployOzonevApp(self.vcenter\_details, self.ova\_details, self.timeout)****def*** *runTest(self):  
 # Actual method call to run the Test  
 self.deploy\_ozone\_machine()*

1. finalize context section:

In this based on the status of self.deployment\_status in deploy\_ozone\_app method , it will call the verification step and pass or fail the test case. Mostly same for all the AUCs.

E.g.:

***def*** *\_finalize\_context(self):* ***if*** *self.deployment\_status:  
 logger.info(****'Deploy Ozone VM Successful'****, False, True)* ***else****:  
 logger.error(****'Deploy Ozone VM Failed'****)* ***raise*** *AssertionError,* ***'Deploy Ozone VM Failed'***

That’s done with AUC part

In \_\_init\_\_.py just import this AUC as

E.g. ***from*** *deploy\_ozone\_app* ***import*** *DeployOzoneVM*

And we need to import this file in the following path

ozone/auc/executables/**\_\_init\_\_.py**

Add a extra line to the file as

**from** deploy\_ozone\_app **import** DeployOzoneVM

Here deploy\_ozone\_app – AUC

DeployOzoneVM – Class Name

**Config file Creation/Update**

In this Framework , we are using 2 files for inputs.

generic.yaml

workflow.yaml

It is in the path ozone/auc/executables/config

generic.yaml –

It is the file created to place the inputs which are common for the entire suite. As of now, it is empty.

workflow.yaml –

It is used for running the functional tests. You need to specify the inputs in the following format.

E.g.

**d**eploy\_ozone\_vApp:  
 **v**center\_details:  
 **v**CenterIPAddress: 192.168.1.15  
 **v**CenterUsername: administrator@vsphere.local  
 **v**CenterPassword: Password123!  
 **v**CenterPort: 443  
 **d**ataCenter: VXRAIL  
 **c**luster: HCI\_CLUSTER  
 **d**atastore: nfsDatastore01  
 **h**ostName: vesxi007.vlab.local  
 **r**esourcePool:  
  
 **o**va\_details:  
 **o**vaPath: Z:\vapp\_dev\develop-branch\EHC-Automation-Tool-dev-build-779.ova  
 **v**mName: Ozone-vAPP-779-site1  
 **n**etwork: EHC Automation  
 **m**asterIP: 192.168.1.90  
 **m**asterFQDN: master1.vlab.local  
 **w**orkerIP: 192.168.1.91  
 **w**orkerFQDN: worker1.vlab.local  
 **g**ateway: 192.168.1.250  
 **n**etmask: 255.255.248.0  
 **d**ns: 192.168.1.10  
  
 **t**imeout: 3000

**Update workflow file with the AUC Created**

We need to update the workflow with the AUC implemented to work with robot

E.g.ozone/workflow/**baseworkflow.py**

Now, we will add a extra function in workflow with your AUC .

First need to import our AUC in it.

*from ozone.auc.executables import DeployOzoneVM*

Next create a method like this

***def*** *deploy\_ozone\_vapp(self, section=None):  
 kw = {****'section'****: section}  
 DeployOzoneVM(  
 self.deploy\_ozone\_vapp.\_\_name\_\_,  
 ctx\_in=self.wf\_context,  
 ctx\_out=self.wf\_context,  
 \*\*kw  
 ).run()*

You need to change the following things

1. method name to appropriate and
2. inside that call your AUC
3. Change the section name to the name specified in config file

Here section would be Site1, Site2 etc

**Robot File Creation**

Create a Robot file with your test cases and run it. It should be under

ozone/robot/ folder

ozone/robot/**ozone.robot**

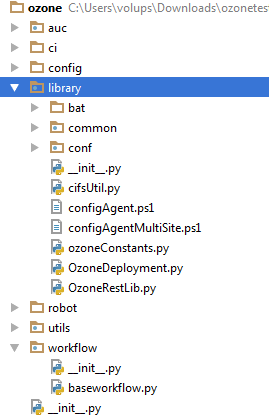
***\*\*\* Settings \*\*\*  
Library*** *ozone.workflow.BaseWorkflow****Documentation*** *Demo Workflow****Suite Setup*** *Initiate Suite****Default Tags*** *CI****\*\*\* Test Cases \*\*\*  
Avamar DD Playbook*** *Start Ozone Session  
 Read excel data EHCINSTALL-1234  
 Upload Licenses EHCINSTALL-1234  
 Create Project EHCINSTALL-1234  
 Update Project Variables EHCINSTALL-1234  
 Execute Main Playbook EHCINSTALL-1234  
 Delete Project EHCINSTALL-1234****\*\*\* Keywords \*\*\*  
Initiate Suite*** *Apply Settings From Files ../config/generic.yaml ../config/workflow\_vpod.yaml  
 Start Vcenter Session  
 Pull Latest Ozone Build  
 Start Ozone Session  
 Configure Ozone vAPP*

Here I am importing the workflow in the starting line only.(BaseWorkflow)

Here EHCINSTALL-1234 is the section number

Finally, in Apply Settings From Files you need to provide generic file and workflow yaml file paths.

Now adding the Libraries Development of Ozone



Here we are using 2 library files

OzoneDeployment.py – uses build automation code for deployment and configuration of Ozone

OzoneRestLib.py – uses Ozone REST Calls for all Ozone related functionality.

<https://pie4.rtp.lab.emc.com/display/EHC/Ozone+API+Documentation#expand-OzoneAPICode>

is the library reference for Ozone REST Calls