CTAM

# Table Of Contents

Contents

[Table Of Contents 1](#_Toc151983249)

[Introduction 2](#_Toc151983250)

[Getting Started 2](#_Toc151983251)

[1. Cloning CTAM Repository 2](#_Toc151983252)

[2. Environment Setup 2](#_Toc151983253)

[3. Discover Test Cases 3](#_Toc151983254)

[4. System Discovery 3](#_Toc151983255)

[5. Basic Test Cases 4](#_Toc151983256)

[Enable & Verify AC Cycle Test Case 4](#_Toc151983257)

[Execute Service Validator 5](#_Toc151983258)

[Full FW Update Flow 5](#_Toc151983259)

[**Compliance Tool for Accelerator Management**](https://github.com/v-ajitpanda/ocp-diag-ctam/tree/developer_msft#compliance-tool-for-accelerator-management)

# Introduction

The [**OCP Test & Validation Initiative**](https://github.com/opencomputeproject/ocp-diag-core) is a collaboration between datacenter hyperscale’s having the goal of standardizing aspects of the hardware validation/diagnosis space, along with providing necessary tooling to enable both diagnostic developers and executors to leverage these interfaces.

Specifically, the [ocp-diag-ctam](https://github.com/opencomputeproject/ocp-diag-ctam) tool provides acceptance testing for Accelerator Management in cloud data centers.

**Note: -**

* To get all compliance test cases to run, mandates a lot of collateral to be readied. This can get overwhelming. This guide should help you with getting used to the tool while incrementally enabling all the collateral.
* Please follow the following steps in the same order. At any point/step, it is assumed that the previous steps PASS indicating that the minimum collateral is met.

# Getting Started

## Cloning CTAM Repository

* Clone repo from GitHub to respective directory - git clone https://github.com/opencomputeproject/ocp-diag-ctam.git
* Change branch to **developer.**
* Create your own branch from developer branch If you want to work on this repository.

## Environment Setup

* Install python with version **python-3.11**.
* Open command prompt and run the below commands.
* Create a Virtual Environment for python using command ***python -m venv <virtual environment name>***
* Activate virtual environment by running command <***path to your virtual environment>/Scripts/Activate***
* Install required packages from pip-requirements.txt file.
* For installing requirements run command ***python -m pip install -r <pip-requirements .txt>***

## Discover Test Cases

* This acts as a test for the local repo setup. Optionally you can explore further and add more test cases along similar lines and rediscover the new set of test cases.
* Check for the **ctam.py** file is present in the current working directory. If not, then go to the directory or else give the path to ctam.py while running the command.
* Run the command to discover all the test cases **python ctam.py -l**

+----------------------------------------------------------------------------------------------------------------------+

| Test Case info table |

+--------- +------- -----+----------+-------------+--------------+--------------+--------------------------------+

| **GroupID** | **GroupName** | **GroupTag** | **TestCaseID** | **TestCaseName** | **TestCaseTag** | **TestCaseWeightScore** |

+----------+-----------------+------------+---------------+------------------+---------------+---------------------+

| | | [] | TestID | Test\_01 | [] | 10 |

| Group | Group | [] | TestID | Test\_01 | [] | 10 | | ID | Name | [] | TestID | Test\_01 | [] | 10 |

| | | [] | TestID | Test\_01 | [] | 10 |

+----------+-----------------+------------+---------------+------------------+---------------+---------------------+

## System Discovery

* Ensure that the system is reachable and BMC & HMC are communicable.
* To run anything we need this properties in **dut info, redfish config uri and .netrc** file in our workspace directory
* **dut\_info.json**

  "properties": {

    "ConnectionIPAddress": {

        "description": "Default prefix for connecting to redfish",

        "type": "string",

        "value": "<>"

    },

    "DefaultPrefix": {

      "description": "Default prefix for connecting to redfish",

      "type": "string",

      "value": "<>/redfish/v1"

    }

  }

* **Redfish\_config\_uri.json**

{

"GPU":{

        "GPUMC": "<>",

        "BaseURI":"<>/redfish/v1",

}

}

* **.netrc file**

<IP ADDRESS 1>

<USERNAME 1>

<PASSWORD 1>

* Make sure all the config files should have inside workspace directory and we need to provide the workspace directory while running the command as command line argument like below.
* Using this configuration, we can run **python ctam.py -w <workspace\_directory> -d**

<add sample o/p>

## Basic Test Cases

* The simple checks will help build the confidence to run FW update test case later.
* **Test Cases**
* ***Python ctam.py -w <workspace\_dir> -t <test\_case\_id or test\_case\_name >***
* **Simple/Basic/Fundamental Test Cases**
  + H10 - CTAMTestRedfishTaskService
  + H7 - CTAMTestRedfishTelemetryService
  + H4 - CTAMTestRedfishUpdateService
  + H8 - CTAMTestRedfishEventService
* **FW/SW Inventory**
  + H5 - CTAMTestRedfishFirmwareInventoryCollection
  + H6 - CTAMTestRedfishFirmwareInventoryExpandedCollection
  + H9 - CTAMTestRedfishSoftwareInventoryCollection
  + H11 - CTAMTestRedfishSoftwareInventoryExpandedCollection
* **Other Test Cases (optional, any failure for the test cases below should not prevent you from moving ahead with the document)**
  + H96 - CTAMTestLogserviceDumpClearlog
  + T2 - CTAMTestTelemetryMRListRead
  + T4 - CTAMTestTelemetryMRRead
  + H97 - CTAMTestLogServiceDumpURIListRead
  + H99 - CTAMTestLogServicesURIListRead
  + H83 - CTAMTestRedfishEventServiceCreateSubscription
  + H81 - CTAMTestRedfishEventServiceDeleteSubscription
  + H80 – CTAMTestRedfishEventServiceSubscription
* <add sample o/p>

# Enable & Verify AC Cycle Test Case

* This will be needed for FW update later.
* To run the below test case we need this properties in dut info, redfish config uri and .netrc file with default configuration.
* Make sure we have the required default configuration files in the workspace directory.
* ***python ctam.py -w <workspace\_dir> -t <test\_case\_id or test\_case\_name >***
* ***AC Cycle***
  + **H100 - CTAMTestAcCyclesInLoop**
  + ***python ctam.py -w workspace -t H100***

# Execute Service Validator

* This is needed for confidence (any failure for the below test cases should **not** prevent you from moving ahead with the document)
* Make sure we have the required default configuration files in the workspace directory.
* ***python ctam.py -w <workspace\_dir> -t <testcase or test\_case\_name >***
* ***Redfish Service Validator***
  + **T0 - CTAMTestServiceValidator**
  + ***python ctam.py -w workspace -t T0***

**Note: -** We need to mention in our test case which URI we are going to validate. For further information please refer to this GitHub link - [Redfish Service Validator](https://github.com/DMTF/Redfish-Service-Validator)

# Full FW Update Flow

* For running firmware update test cases we need additional configuration file which is package\_info.json in the workspace directory.
* In package\_info.json file we need to add the configs below.
* {

"GPU\_FW\_IMAGE": {

    "Path": "directopry from where we need to take the file.",

    "Package": "<fwpkg\_file\_to\_update.fwpkg>",

    "Version": "version",

    "JSON": "<fwpkg\_json\_file.json>",

    "Vendor": "Vendor Name"

  }

}

* First, we need to do precheck to check if the device is updated or not.
* If updateable then we need to stage, the firmware (copy the firmware file into target device).
* Then with a power cycle we are activating the firmware.
* After firmware update completed we need to verify that if the firmware is updated correctly or not. If verified, then test case passed else fail.
* ***python ctam.py -w <workspace\_dir> -t <testcase\_id or test\_case\_name>***
* ***Full FW Update***
  + **F1 - CTAMTestFullDeviceUpdate**
  + ***python ctam.py -w workspace -t F1***

# Full FW Rollback Flow

* For running firmware update test cases we need additional configuration file which is package\_info.json in the workspace directory.
* In package\_info.json file we need to add the configs below.
* {

"GPU\_FW\_IMAGE\_BACKUP":{

    "Path": "directopry from where we need to take the file.",

    "Package": "<fwpkg\_file\_to\_update.fwpkg>",

    "Version": "version",

    "JSON": "<fwpkg\_json\_file.json>",

    "Vendor": "Vendor Name"

  }

}

* First, we need to do precheck to check if the device is updated or not.
* If updateable then we need to stage, the firmware (copy the firmware file into target device).
* Then with a power cycle we are activating the firmware.
* After the firmware update is completed, we need to verify that if the firmware is updated correctly or not. If verified, then test case passed else fail.
* ***python ctam.py -w <workspace\_dir> -t <testcase\_id or test\_case\_name>***
* ***Full FW Update RollBack***
  + **F1 - CTAMTestFullDeviceUpdateRollback**
  + ***python ctam.py -w workspace -t F0***