

Matter Test-Harness User Manual

v2.13+summer2025

Test-Harness Links

Matter Version	TH Version	TH image location	Supporting Documentation	SDK commit	Comments
Matter 1.0	v2.6	link	Causeway Link	96c9357	
Matter 1.1	v2.8.1	link	Causeway Link	5a21d17	
Matter 1.2	TH-Fall2023	link	Causeway Link	19771ed	
Matter 1.3	v2.10.2+spring2024	N/A	Causeway Link	50bcad1	To install, use the tag "v2.10.2+spring2024" in the instructions of TH Installation on Raspberry Pi
Matter 1.4	v2.11+fall2024	N/A	Causeway Link	f2e5de7	To install, use the tag "v2.11+fall2024" in the instructions of TH Installation on Raspberry Pi
Matter 1.4.1	v2.12+spring2025	N/A	Causeway Link	91eab26	To install, use the tag "v2.12+spring2025" in the instructions of TH Installation on Raspberry Pi
Matter 1.5	v2.13+summer2025	N/A		e34015c	To install, use the tag "v2.13-beta2+spring2025" in the instructions of TH Installation on Raspberry Pi

Revision History

Revision	Date	Author	Description
1	11-Jan-2022	[GRL]Suraj Seenivasan	* Initial version of the user manual used for V1.0.
2	22-Aug-2022	[GRL]Suraj Seenivasan	* Updated the steps required to run the Python script inside docker: Input given by Tennessee Carmel-Veilleux.
3	06-Feb-2023	[GRL]Suraj Seenivasan	* Added instructions to update the existing image and personal access token: How to execute simulated test cases.
4	09-Feb-2023	[GRL]Suma	* Structuring the user manual. * Added relevant screenshots.
5	24-Feb-2023	[SiLabs]Yinyi Hu	* Instruction to Flash SiLabs RCP. * Instruction to collect Logs and submit to TEDS.
6	01-Mar-2023	[GRL]Suraj Seenivasan	* Instructions to flash nRF52840 Dongle and nRF52840 DK.
7	16-Mar-2023	[Apple]Fábio Wladimir Monteiro Maia	* Added TH layout explanation.
8	12-Jun-2023	[Apple]Carolina Lopes	* Added instructions to install TH without a Raspberry Pi.
9	06-Sep-2023	[Apple]Antonio Melo Jr.	* Moved the table of contents to a further page. * Added TH links for images and support documentations.
10	25-Oct-2023	[Apple]Hilton Lima	* Added SDK commit column to Test-Harness Links table.
11	26-Oct-2023	[Apple]Carolina Lopes	* Updated some links and images.
12	06-Dec-2023	[Apple]Hilton Lima	* Added Custom Yaml test section. * Updated TH update instructions. * Revision History table style formatted.
13	07-Dec-2023	[Apple]Carolina Lopes	* Updated instructions on how to install TH without a Raspberry Pi.
14	14-Dec-2023	[Apple]Hilton Lima	* Updated SDK Python Test section.
15	20-Dec-2023	[Apple]Antonio Melo Jr.	* Updated Test Harness links related to release of Spring 2024.
16	04-Jan-2024	[Apple]Antonio Melo Jr.	* Updated Test Harness Release location link to the new Drive folder.

17	04-Jan-2024	[Apple]Romulo Quidute	* Renamed Custom Yaml test section to Customized Test Scripts and also added Custom Python test information.
18	05-Jan-2024	[Apple]Carolina Lopes	* Added information about the SDK Python Test suites.
19	30-Jan-2024	[Apple]Romulo Quidute	* Updated some referenced links.
20	07-Feb-2024	[Apple]Hilton Lima	* Added 'Common Test Failures' section for SDK tests.
21	07-Feb-2024	[Apple]Carolina Lopes	* Added 'Bringing up the Matter Python REPL' section.
22	15-Feb-2024	[Apple]Hilton Lima	* Updated 'TH Installation on Raspberry Pi' section.
23	26-Feb-2024	[Apple]Carolina Lopes	* Added volume mapping for data model XML files on docker run commands for SDK containter.
24	27-Feb-2024	[Apple]Carolina Lopes	* Updated mapping in docker run command for SDK container.
25	27-Mar-2024	[Apple]Hilton Lima	* Update nRF52840 firmware download link.
26	01-Apr-2024	[Apple]Hilton Lima	* Update download link to Flash SiLabs RCP.
27	01-Apr-2024	[Apple]Romulo Quidute	* Update OTBR scripts location.
28	17-Apr-2024	[Apple]Hilton Lima	* Updated Test Harness links related to final release of Spring 2024.
29	18-Apr-2024	[Apple]Hilton Lima	* Update section 'TH installation without a Raspberry Pi'.
30	30-Apr-2024	[Apple]Hilton Lima	* Update location for update scripts.
31	10-Jun-2024	[Apple]Hilton Lima	* Updated Test Harness links.
32	17-Jun-2024	[Google]Tennessee Carmel-Veilleux	* Added an example of an external operational dataset.
33	01-Aug-2024	[Apple]Carolina Lopes	* Added information about Certification Mode.
34	02-Aug-2024	[Apple]Antonio Melo Jr.	* Update the Ubuntu mentions to the current supported version.

35	06-Aug-2024	[Apple]Antonio Melo Jr.	<ul style="list-style-type: none"> * Add the TH release version to the front page of the document. * Add a warning to TH update when using a different Ubuntu version. * Update the OTBR start script parameter. * Fix the Thread RCP Firmware link. * Add Nrfconnect Sample APPs Firmwares section with link.
36	09-Aug-2024	[Apple]Antonio Melo Jr.	<ul style="list-style-type: none"> * Adding the new cleanup procedure to the troubleshooting of TH Installation section.
37	13-Aug-2024	[Apple]Hilton Lima	<ul style="list-style-type: none"> * Remove old references of TH image (obsolete). * Add a warning to ensure that the username needs to be 'ubuntu'.
38	30-Aug-2024	[Apple]Hilton Lima	<ul style="list-style-type: none"> * Replace images by text examples in 'Project Configuration' section. * Removed section 'Collect Logs and Submit to TEDS'. * Added informations about 'qr-code' and 'manual-code' parameters.
39	04-Sep-2024	[Apple]Hilton Lima	<ul style="list-style-type: none"> * Moved PIXIT section. * Added 'Test Parameters for SDK Python Tests' section. * Changed Table of Contents display level.
40	24-Sep-2024	[Apple] Hilton Lima	<ul style="list-style-type: none"> * Updated nRF util and desktop application links.
41	14-Oct-2024	[Apple]Romulo Quidute	<ul style="list-style-type: none"> * Added Warning for WIFI_SSID with special chars or empty spaces. * Updated Test-Harness Links for v2.11+fall2024 release.
42	03-Jan-2025	[Apple]Romulo Quidute	<ul style="list-style-type: none"> * Added Reuse commissioning information section.
43	25-Feb-2025	[Apple]Romulo Quidute	<ul style="list-style-type: none"> * Updated supported Ubuntu version to 24.04.2
44	25-Feb-2025	[Apple]Antonio Melo Jr.	<ul style="list-style-type: none"> * Replacing the patch number with a 'x' in the supported Ubuntu version.
45	13-Mar-2025	[Apple]Romulo Quidute	<ul style="list-style-type: none"> * Updated Test-Harness Links for v2.12+spring2025 release.
46	31-Mar-2025	[Apple]Romulo Quidute	<ul style="list-style-type: none"> * Added mapping volume section inside SDK Python Tests.

Table of contents

1. Introduction	7
2. References	8
3. Test-Harness (TH) Design	9
3.1. TH Layout	9
3.2. Data Model	10
3.3. Data Flow	11
4. Getting Started with Matter Test-Harness (TH)	12
4.1. TH Installation on Raspberry Pi	12
4.1.1. Prerequisites	12
4.1.2. TH Installation on Raspberry Pi	13
4.2. TH installation without a Raspberry Pi	14
4.2.1. Create an Ubuntu virtual machine	14
4.2.2. Setup TH in Ubuntu	15
4.2.3. Substitute the SDK's docker image and update sample apps	15
4.3. Update Existing TH	16
4.4. Updating Existing Yaml Test Script	16
4.5. Customized Test Scripts (Yaml/Python Tests)	17
4.6. Troubleshooting	18
4.6.1. Read-Only File System Error	18
4.6.2. Generate Personal Access Token	19
4.6.3. Bringing Up of Docker Containers Manually	19
4.6.4. Cleaning The Environment Manually	20
5. Bringing Up of Matter Node (DUT) for Certification Testing	21
5.1. Bringing Up of Reference Matter Node (DUT) on Raspberry Pi	21
5.1.1. To Provision Raspberry Pi Using Wi-Fi Configuration	22
5.1.2. To Provision Raspberry Pi Over Ethernet Configuration	22
5.2. Bringing Up of Reference Matter Node (DUT) on Thread Platform	22
5.2.1. Prerequisites	22
5.2.2. Setting Up Thread Board (nRF52840-DK)	23
5.2.2.1. Instructions to Set Up nRF52840-DK Using nRF Connect Desktop Application Tool	23
5.2.2.2. Instructions to Set Up nRF52840-DK Using Docker Environment	26
5.2.2.3. Building and Flashing Sample Apps for nRF-Connect	27
6. Bringing up the Matter Python REPL	29
7. OT Border Router (OTBR) Setup	30
7.1. Instructions to Flash the Firmware NRF52840 RCPDongle	30
7.2. Nrfconnect Sample APPs Firmwares to Flash on the NRF52840DK Kit	31
7.3. Instructions to Flash SiLabs RCP	31
7.4. Forming Thread Network and Generating Dataset for Thread Pairing	32

7.5. Troubleshooting: Boarder Router Container failure to initialize	35
8. Test Configuration	37
8.1. Project Configuration	37
8.1.1. Projects Menu	37
8.1.2. Wi-Fi Mode	38
8.1.2.1. On Network Mode	39
8.1.3. Thread Device Mode	39
8.1.4. PAA Certificates	40
8.2. Test Parameters	41
8.2.1. Upload PICS File	42
8.2.2. Test Menu	42
8.2.3. Utility Menu	47
8.2.4. Settings Menu	47
9. Test Case Execution	48
9.1. Automated and Semi Automated Tests	49
9.1.1. Automated Test Cases	49
9.1.2. Semi Automated Test Cases	49
9.2. Python Tests	51
9.3. Manual Tests	53
9.4. Simulated Tests	54
9.5. SDK Python Tests	55
9.5.1. Run Tests Inside SDK Docker Container	55
9.5.1.1. Prerequisite	55
9.5.1.2. Mapped Volumes	55
9.5.1.3. Placeholders for Steps	56
9.5.1.4. Common Steps	56
9.5.1.5. For On-Network Pairing	56
9.5.1.6. For BLE+Wi-Fi Pairing	57
9.5.1.7. For BLE+Thread Pairing	57
9.5.1.8. Post-Test Steps	57
9.5.1.9. Possible Issues	57
Common Test Failures	58
9.5.2. Run Tests on the TH User Interface	58
9.5.2.1. Test Parameters for SDK Python Tests	58
PIXIT Support	58
Test Parameters Examples	59
9.5.2.2. Test suites	60
9.5.2.3. Reuse commissioning information	61

1. Introduction

The Matter Test-Harness is a comprehensive test tool used for certification testing of Matter devices in accordance with the Matter protocol as defined in the [Matter specification](#).

This user guide serves as the primary user documentation to work with the Test-Harness (TH) tool, providing high-level architecture of the tool, how to use the tool to execute certification tests and submit the test results to CSA for certification.

The TH tool runs on the Raspberry Pi platform, providing an intuitive Web user interface to create a test project, configure the project/Device Under Test (DUT) settings, load the required test cases using the PICS xml file and execute test cases for various devices (commissioner, controller and controlee) as defined in the Matter specification.

The TH tool provides an option to execute the following test scripts— Automated, Semi Automated, Python, Manual and Simulated. Upon completion of the test execution, detailed logs and test results will be available for user analysis. The user will also be able to submit logs to ATL's for review to obtain device certification.

The TH tool can be used by any DUT vendor to run the Matter certification tests, or by any hobby developer to get acquainted with the Matter certification testing tools or technologies.

2. References

1. Matter Specification: [Matter Specification \(Causeway\)](#) / [Matter Specification \(Github\)](#)
2. Matter SDK Repo github: <https://github.com/project-chip/connectedhomeip>
3. Matter Test Plans: [Matter Test Plans \(Causeway\)](#) / [Matter Test Plans \(GitHub\)](#)
4. PICS Tool: [PICS Tool - Connectivity Standards Alliance \(csa-iot.org\)](#)
5. XML Files: <https://groups.csa-iot.org/wg/members-all/document/folder/4120>
6. TEDS Matter tool: <https://groups.csa-iot.org/wg/matter-wg/document/28545>

Important: Some links contained in this user manual require a CSA membership and authentication as a CSA authorized user in order to be accessed

3. Test-Harness (TH) Design

This section outlines the TH architecture, data model and data flow on how different components of TH communicate with each other.

3.1. TH Layout



Figure 1. The Test-Harness Layout

Each of the main subsystems of the Test Harness (Proxy, Frontend, Backend and Database) runs on its own docker container deployed to a Ubuntu Raspberry Pi platform. The Proxy container hosts an instance of the [traefik](https://traefik.io/traefik/) application proxy (<https://traefik.io/traefik/>) which is responsible to route user requests coming from an external (to the Raspberry Pi) web browser to either the Frontend or the Backend as appropriate. The Frontend container serves the dynamic web pages that comprise the Web GUI to be rendered on the user browser including the client-side logic. According to that client-side logic and user input, REST API requests are sent again by the external browser to the Application Proxy and get redirected to the Backend container, where a FastAPI (<https://fastapi.tiangolo.com/>) Python application implements the server-side logic. Any application information that needs to be persisted gets serialized and written by the server-side logic to the Postgres database running in the Database container.

In addition to the four main containers described above, which get created and destroyed when the Raspberry Pi platform respectively boots up and shuts down, two other containers are created and destroyed dynamically on demand according to the test execution lifecycle: the SDK container and the OTBR container. The SDK container has copies of the Matter SDK tools (binary executables)

which can be used to play the role of clients and servers of the Matter protocol in test interactions, either as Test Harness actuators or DUT simulators. That container gets automatically created and destroyed by the server-side logic at the start and at the end, respectively, of a Test Suite which needs actuators or simulators. The OTBR container, on the other hand, hosts an instance of the Open Thread Border Router and needs to be explicitly started by the TH user when they want to test a real Matter device that runs over a Thread fabric, as described in [Section 7, OT Border Router \(OTBR\) Setup](#).

3.2. Data Model

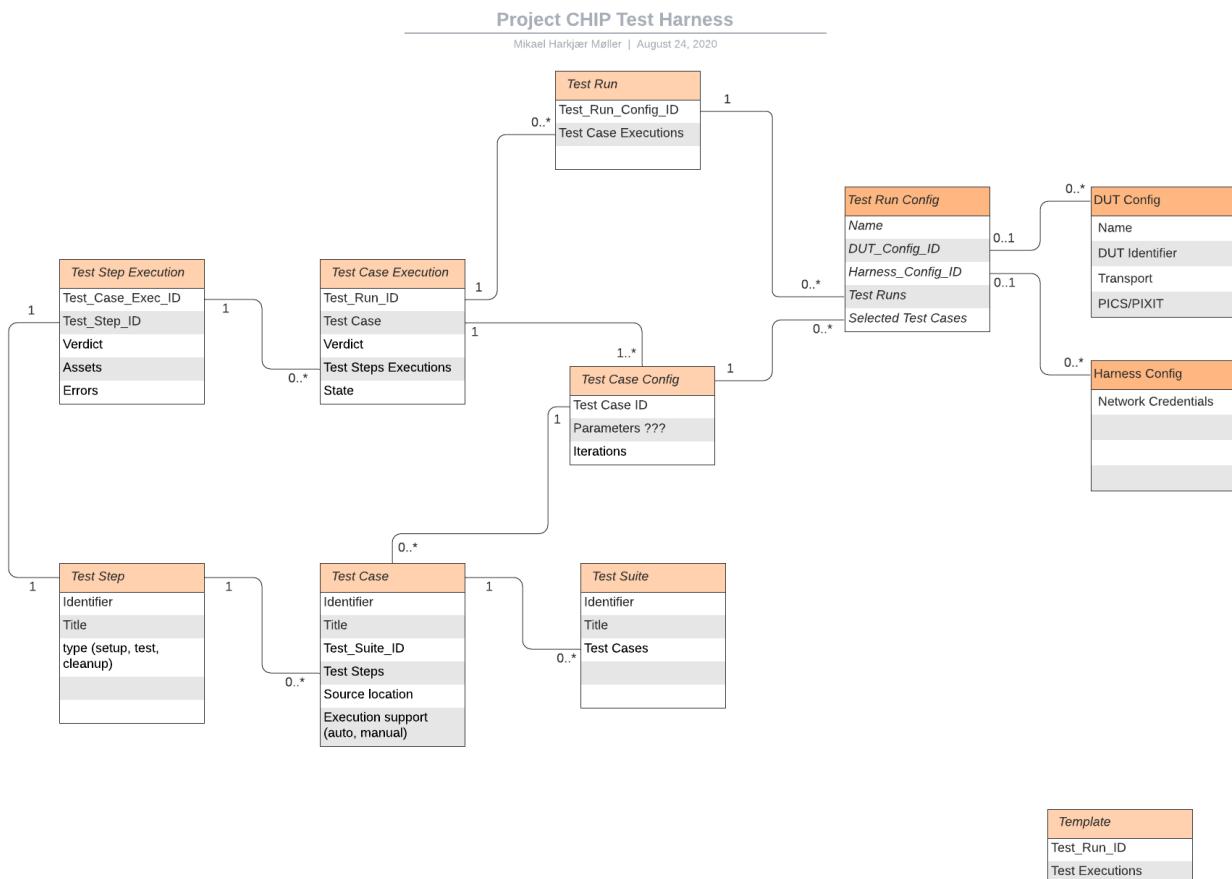


Figure 2. The Data Model

The data model diagram in Figure 2 shows the various data objects that the Test Execution consumes and maintains and the relationship between these data objects.

- Test Run
- Test Run Config
- DUT Config
- Harness Config
- Test Case Execution
- Test Step Execution

- Test Case
- Test Step
- Test Suite
- Test Case Config

3.3. Data Flow



Figure 3. The Data Flow

4. Getting Started with Matter Test-Harness (TH)

The Matter Node (DUT) that is used for certification testing can either be a commissioner, controller or controlee.

If the DUT is a controlee (e.g., light bulb), the TH spins a reference commissioner/controller using chip-tool binary shipped with the SDK. The TH commissioner provisions the DUT and is used to execute the certification tests on the controlee.

If the DUT is a commissioner/controller, the Test TH spins an example accessory that is shipped with the SDK and uses that for the DUT to provision, control and run certification tests.

Refer to [Section 5, Bringing Up of Matter Node \(DUT\) for Certification Testing](#) to bring up the DUT and then proceed with device testing by referring to [Section 8, Test Configuration](#).

For hobby developers who want to get acquainted with certification tools/process/TC's, can spin DUT's using the example apps provided in the SDK. Refer to the instructions to set up one [here](#).

The TH runs on the official **Ubuntu Server 24.04.x LTS (64-bit)** version. If the TH device happens to be using a different Ubuntu release or other OS, we strongly recommend fresh installing version Ubuntu Server 24.04.x LTS (64-bit) for reliable results.

The official installation method uses a Raspberry Pi ([TH Installation on Raspberry Pi](#)), but there's an alternative method used in the tool's development that uses a virtual machine instead ([TH installation without a Raspberry Pi](#)). Keep in mind that thread networking is not officially supported in VM installations at the moment.

4.1. TH Installation on Raspberry Pi

There are two ways to obtain the latest TH on Raspberry Pi. Follow the instructions in [Section 4.1.2, TH Installation on Raspberry Pi](#) to install TH from scratch OR if you already have the TH, follow the instructions in [Section 4.4, Update Existing TH](#) to update the TH.



This instruction applies to the latest version of the Test Harness this document refers to. For earlier versions of the TH please follow the user guide of that specific TH version as, for example, Ubuntu versions might differ per installation.

4.1.1. Prerequisites

The following equipment will be required to have a complete TH setup:

- **Raspberry Pi Version (4 or 5) with SD card of minimum 64 GB Memory**

The TH will be installed on Raspberry PI. The TH contains couple of docker container(s) with all the required dependencies for certification tests execution.

- **Windows or Linux System (Laptop/Desktop/Mac)**

The Mac/PC will be used to flash the Ubuntu image on the SD card to be used on Raspberry Pi. Download the [Raspberry Pi Imager](#) or [Balena Etcher](#) tool. The same can be used to set up the required build environment for the Matter SDK or building Matter reference apps for various platforms.

- **RCP dongle**

If the DUT supports thread transport, an RCP dongle provisioned with a recommended RCP firmware for the default OTBR router that comes with the TH will be required to function properly. Currently, the OTBR can work with a Nordic RCP dongle or a SiLabs RCP dongle. Refer to [Section 6, OT Border Router \(OTBR\) Setup](#) on how to install the RCP firmware.

4.1.2. TH Installation on Raspberry Pi



Starting with version v2.10 we have moved from distributing TH as an SD-Card image to publishing the TH Docker containers at Github Container Registry and pulling them at install time. By doing that the release process has been made much faster and less error-prone, while at the same time installation time has gone shorter.

1. Place the blank SD card into the user's system USB slot.
2. Open the [Raspberry Pi Imager](#) or [Balena Etcher](#) tool on the Mac/PC and select the 'Ubuntu Server 24.04.x LTS (64-bit)'.
 - Edit the SO custom settings to:
 - username: ubuntu
 - password: raspberrypi
 - hostname: ubuntu
 - Make sure you have enabled the SSH service.
3. After the SD card has been flashed, remove the SD card and place it in the Raspberry Pi's memory card slot.
4. Power on the Raspberry Pi and ensure that the local area network, display monitor and keyboard are connected.
5. Enter the username and password.
6. Install the TH system:
 - Clone the TH repository:
 - `$git clone -b <Target_Branch/Tag> https://github.com/project-chip/certification-tool.git`
 - Goto to TH folder:
 - `$cd certification-tool`



The username must be 'ubuntu'. Changing the name may cause problems running TH.

- password: raspberrypi
- hostname: ubuntu

◦ Make sure you have enabled the SSH service.

3. After the SD card has been flashed, remove the SD card and place it in the Raspberry Pi's memory card slot.

4. Power on the Raspberry Pi and ensure that the local area network, display monitor and keyboard are connected.

5. Enter the username and password.

6. Install the TH system:

◦ Clone the TH repository:

▪ `$git clone -b <Target_Branch/Tag> https://github.com/project-chip/certification-tool.git`

◦ Goto to TH folder:

▪ `$cd certification-tool`

- Install/configure the TH dependencies:
 - `./scripts/pi-setup/auto-install.sh`
 - At the end of the script, select option 1 to restart the RaspberryPi.
7. Wait about 10 minutes.
8. Using the `ifconfig` command, obtain the IP address of the Raspberry Pi. The same IP address will be used to launch the TH user interface on the user's system using the browser.
 9. Proceed with test configuration and execution (refer to [Section 8, Test Configuration](#) and [Section 9, Test Case Execution](#) respectively).

4.2. TH installation without a Raspberry Pi

The official installation method uses a Raspberry Pi ([TH Installation on Raspberry Pi](#)). **This alternative installation method is targeted for development purpose and it only supports onnetwork pairing mode.**



To install TH without using a Raspberry Pi you'll need a machine with Ubuntu Server 24.04.x LTS (64-bit). You can [create a virtual machine](#) for this purpose, but **be aware that if the host's architecture is not arm64** you'll need to substitute `backend`, `frontend` and [the SDK's docker image](#) in order for it to work properly.



Images for linux/amd64 will not always be available in the github registry. So, if necessary, the images need to be built locally using the following script:

```
./certification-tool/scripts/build.sh
```

4.2.1. Create an Ubuntu virtual machine

Here's an example of how to create a virtual machine for TH using multipass (<https://multipass.run/>).

Please make sure the docker images are compatible with the host architecture.

- Install multipass

```
brew install multipass
```

- Create new VM with Ubuntu Server 24.04.x LTS (64-bit) (2 cpu cores, 8G mem and a 50G disk)

```
multipass launch 24.04.x -n matter-vm -c 2 -m 8G -d 50G
```

- SSH into VM

```
multipass shell matter-vm
```



About Multipass:

Seems like bridged network is not available, so you will not be able to test with DUT outside the docker container, but you can develop using the sample apps on

the platform.

4.2.2. Setup TH in Ubuntu

- Clone git repo

```
git clone -b <Target_Branch/Tag> https://github.com/project-chip/certification-tool.git
```

- Go into the repo directory

```
cd certification-tool
```

- Run TH auto install script

```
./scripts/ubuntu/auto-install.sh
```

- Reboot VM

If using multipass, to find the IP address use the command

```
multipass list
```

4.2.3. Substitute the SDK's docker image and update sample apps

If the platform of the machine that will run the TH is 'linux/arm64' it will not be necessary to build a new SDK docker image.

To run TH on a machine using the 'linux/amd64' platform, you will need to first build a new SDK docker image.

- Get the SDK commit SHA

Value for variable `SDK_DOCKER_TAG` in TH repository path `certification-tool/backend/app/core/config.py`

- Download the Dockerfile for chip-cert-bins from the commit you need

Substitute `<COMMIT_SHA>` with the value from `SDK_DOCKER_TAG`:

```
github.com/project-chip/connectedhomeip/blob/<COMMIT_SHA>/integrations/docker/images/chip-cert-bins/Dockerfile
```

- Copy Docker file to TH's machine
- Make sure that no other SDK image for that commit SHA is loaded in the machine

Run `docker images`

If there's an image with a tag for the commit you're using, delete that image

```
docker image rm <IMAGE_ID>
```

- Build new SDK image (this could take about 3 hours)

Substitute <COMMIT_SHA> with the value from **SDK_DOCKER_TAG**:

```
docker buildx build --load --build-arg COMMITHASH=<COMMIT_SHA> --tag connectedhomeip/chip-cert-bins:<COMMIT_SHA> .
```

- Update TH sample apps

To update your sample apps using the new image run this script in the certification-tool repository
`./backend/test_collections/matter/scripts/update-sample-apps.sh`

4.3. Update Existing TH



If the Operating System is not the **Ubuntu Server 24.04.x LTS (64-bit)**, please flash and use a SD card with that Ubuntu release to use this version of Test Harness. Beware that the auto update process below will fail in the case of a different release version.

To update an existing TH environment, follow the instructions below on the terminal.

```
cd ~/certification-tool
./scripts/ubuntu/auto-update.sh <Target_Branch/Tag>
./scripts/start.sh
```

Wait for 10 mins and open the TH application using the browser

4.4. Updating Existing Yaml Test Script

It is possible to update yaml test script content by directly editing the file content. It is useful when validating small changes or fixing misspelled commands.

Yaml files are located at:

```
~/certification-
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/yaml_tests/yaml/sdk/
```

To update an existing Yaml test script: (e.g. `Test_TC_ACE_1_1.yaml`)

- Open the script file:

```
~/certification-
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/yaml_tests/yaml/sdk/Test_TC_ACE_1_1.yaml
```

- Update/change the desired information.
- Save and close the file.
- Restart TH's backend container:

```
$docker restart certification-tool_backend_1
```

- Changes will be available on the next execution of the yaml test.

To create a new Yaml test script:

- Use an existing test script as a starting point.
- Rename the file to a new one: e.g. `Test_TC_ACE_1_1.yaml` to `Test_TC_ACE_9_9.yaml`
- Update the name entry inside the yaml file:

```
FROM name: 42.1.1. [TC-ACE-1.1] Privileges
```

```
TO name: 42.1.1. [TC-ACE-9.9] Privileges
```

- Proceed as explained on updating an existent yaml file.

4.5. Customized Test Scripts (Yaml/Python Tests)

To use customized tests, the files must be placed in the specific folder (described below). This way, Test-Harness will load and display the available tests on the interface. These tests will not be affected if the system is restarted or if the SDK Yaml tests are updated.

Custom Yaml files folder are located at:

```
~/certification-
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/yaml_tests/yaml/custom/
```

Custom Python files folder are located at:

```
~/certification-
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/python_testing/scripts/custom/
```



Figure 4. Test-Harness displaying the custom tests.

Hint: You can copy the original SDK Yaml/Python test to Custom Yaml/Python folder and do any changes on it.

4.6. Troubleshooting

4.6.1. Read-Only File System Error

- During the execution of TH installation commands if a read-only file system error or an error showing "Is docker daemon running?" occurs, follow the steps below to fix the issue:

```
$sudo fsck ( Press 'y' for fixing all the errors )
```

- Upon successful completion, try the following commands:

```
$sudo reboot
ssh back into the TH IP address using:
$ssh ubuntu@<IPADDRESS-OF-THE-RASPI>
```

- In case "sudo fsck" fails, use the following commands:

```
sudo fsck -y -f /dev/mmcblk0p2
fsck -y /dev/mmcblk0p2
```

- In case the "remote: Repository not found" fatal error occurs, try the following steps to fix the issue. Clone the certification-tool with personal access token (Refer to [Section 4.2.2, Generate Personal Access Token](#) to generate the personal access token) and follow the steps below.

```
cd ~
```

Take the backup of Test Harness binary using below command:

```
$mv certification-tool certification-tool-backup  
$git clone https://<token>@github.com/project-chip/certification-tool.git
```

Follow the instructions given in the section below on how to [update an existing Test-Harness](#)

4.6.2. Generate Personal Access Token

The Personal Access Token may be required during the process of updating an existing TH. Below are the instructions to obtain the personal access token.

1. Connect to the Github account (the one recognized and authorized by Matter).
2. On the upper-right corner of the page, click on the profile photo, then click on **Settings**.
3. On the left sidebar, click on **Developer settings**.
4. On the left sidebar, click on **Personal access tokens** [Personal access tokens (classic)].
5. Click on **Generate new token** .
6. Provide a descriptive name for the token.
7. Enter an expiration date, in days or using the calendar.
8. Select the scopes or permissions to grant this token.
9. Click on **Generate new token** .
10. The generated token will be printed out on the screen. Make sure to save it as a local copy as it will disappear.



Sample token: ghp_hUQExoppLKma*****Urg4P

4.6.3. Bringing Up of Docker Containers Manually

During the initial reboot of the Raspberry Pi, if the docker is not initiated automatically, try the following command on the Raspberry Pi terminal to bring up the dockers.

Use the command `ssh ubuntu@IP_address` from the PC to log in to Raspberry Pi. Refer to previous sections on how to obtain the IP address of Raspberry Pi.

Once the SSH connection is successful, start the docker container using the command
`$./certification-tool/scripts/start.sh`

The above command might take a while to get executed, wait for 5-10 minutes and then proceed with the Test Execution Steps as outlined in the below sections.

4.6.4. Cleaning The Environment Manually

If the Test-Harness environment is facing issues to install, update or start and no other action is working, you may try the cleanup command followed by a install operation.



Please, be advised that this cleanup operation will delete all previous data from the TH database, along with all the docker networks, containers, images used by the application and more.

Follow the bellow procedure to clean and install Test-Harness:

Use the command `ssh ubuntu@IP_address` from the PC to log in to Raspberry Pi. Refer to previous sections on how to obtain the IP address of Raspberry Pi.

Once the SSH connection is successful, clean the environment using the command:

```
$ ./certification-tool/scripts/clean-up.sh
```

Finally, execute a new installation with the following command:

```
$ ./certification-tool/scripts/pi-setup/auto-install.sh
```

5. Bringing Up of Matter Node (DUT) for Certification Testing

A Matter node can either be a commissioner, controller, conteree, software component or an application. The Matter SDK comes with a few example apps that can be used by Vendors as a reference to build their products. Refer to the examples folder in the [SDK github repo](#) for the same.

DUT vendors need to get the device flashed with the production firmware revision that they want to get their device certified and execute all the applicable TC's for their products using the TH. DUT vendors can skip the below sections as the TH brings up the reference applications automatically during the certification tests execution.

A hobby developer can build Matter reference apps either using a Raspberry Pi or Nordic DK board (if the user wants to use thread transport). Follow the instructions below for the [Raspberry Pi](#) and [Nordic](#) platforms.

5.1. Bringing Up of Reference Matter Node (DUT) on Raspberry Pi

In the case where a device maker/hobby developer needs to bring up a sample/reference DUT, i.e. light bulb, door lock, etc. using the example apps provided in SDK and verify provisioning of the DUT over the Bluetooth LE, Wi-Fi and Ethernet interfaces, follow the below steps to set up the DUT.

Users can either use the example apps (i.e. light bulb, door lock, etc.) that are shipped with the TH OR build the apps from the latest SDK source.

To use the apps that are shipped with the TH, follow the instructions below:

- Do a fresh install of TH ([Installation on Raspberry Pi](#)).
- Go to the apps folder in /home/ubuntu/apps (as shown below) and launch the app that the user is interested in.

```
ubuntu@ubuntu:~/apps$ pwd  
/home/ubuntu/apps  
ubuntu@ubuntu:~/apps$ ls  
chip-all-clusters-app      chip-app1      chip-cert      chip-lock-app      chip-ota-requester-app  chip-tool      chip-tv-casting-app  
chip-all-clusters-minimal-app  chip-bridge-app  chip-lighting-app  chip-ota-provider-app  chip-shell    chip-tv-app   thermostat-app  
ubuntu@ubuntu:~/apps$
```

To build the example apps from the latest SDK source, follow the instructions below:

- User to acquire Raspberry Pi Version (4 or 5) with SD card of minimum 64 GB memory.
- Do a fresh install of the Ubuntu Server 24.04.x LTS (64-bit) image and install all the required dependencies as outlined in <https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/BUILDING.md>.
- Clone the connected home SDK repo using the following commands:

```
$ git clone git@github.com:project-chip/connectedhomeip.git --recursive  
$ cd connectedhomeip  
$ source scripts/bootstrap.sh  
$ source scripts/activate.sh
```

- Select the sample app that the user wants to build as available in the examples folder of the SDK repo e.g., lighting-app, all-cluster-app. The user needs to build these apps for the Linux platform using the following command:

Build the app using the below command:

```
./scripts/examples/gn_build_example.sh examples/all-clusters-app/linux/examples/all-clusters-app/linux/out/all-clusters-app chip_inet_config_enable_ipv4=false
```

5.1.1. To Provision Raspberry Pi Using Wi-Fi Configuration

The sample app (lighting-app or lock-app or all-cluster-app) can be provisioned over the Wi-Fi network when the app is launched with the "--wifi" argument.

```
./chip-all-clusters-app --wifi
```

5.1.2. To Provision Raspberry Pi Over Ethernet Configuration

The sample app (lighting-app or lock-app or all-cluster-app) can be provisioned over the Ethernet (using onnetwork configuration) that it is connected when the app is launched with no arguments.

```
./chip-all-clusters-app
```

5.2. Bringing Up of Reference Matter Node (DUT) on Thread Platform

Follow the instructions below to set up the Matter Node on Thread Platform. For additional reference, go to the following link:

<https://github.com/project-chip/connectedhomeip/tree/master/examples/all-clusters-app/nrfconnect#matter-nrf-connect-all-clusters-example-application>

5.2.1. Prerequisites

The following devices are required for a stable and full Thread Setup:

- **DUT:** nRF52840-DK board and one nRF52840-Dongle



The DUT nRF52840-DK board mentioned in this manual is used for illustration purposes only. If the user has a different DUT, they will need to configure the DUT following the DUT requirements.

5.2.2. Setting Up Thread Board (nRF52840-DK)

To set up the Thread Board, follow the instructions below.



The nRF52840-DK setup can be performed in two methods either by flashing the pre-built binary hex of sample apps which is released along with the TH by using the nRF Connect Desktop application tool (refer Section 5.2.2.1) or by building the docker environment to build the sample apps (refer Section 5.2.2.2).

5.2.2.1. Instructions to Set Up nRF52840-DK Using nRF Connect Desktop Application Tool

a. Requirements:

1. nRF Connect for Desktop tool installer: [Link](#)

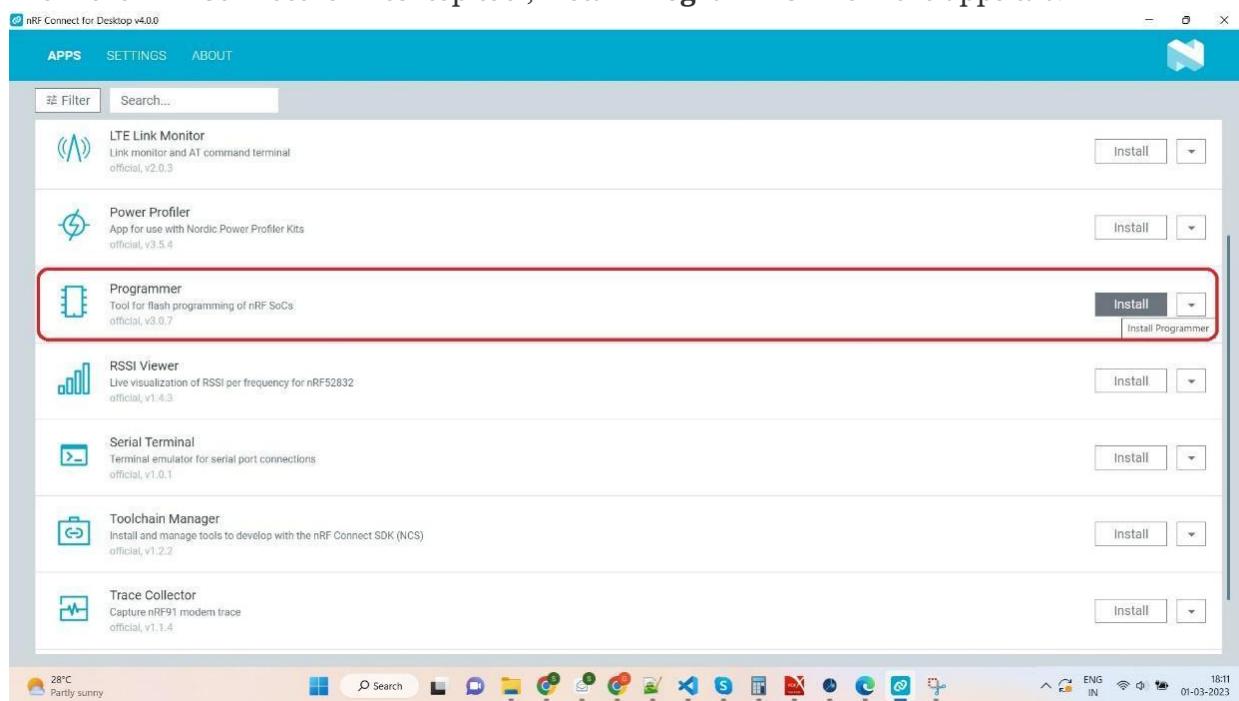


The J-Link driver needs to be separately installed on macOS and Linux. Download and install it from [SEGGER](#) under the section J-Link Software and Documentation Pack.

2. Download thread binary files which are released along with the TH.

b. From the User Interface:

1. Connect nRF52840-DK to the USB port of the user's operating system.
2. From the nRF Connect for Desktop tool, install **Programmer** from the apps tab.



3. Open the Programmer tool to flash the downloaded binary hex file on nRF52840-DK.



4. In the Programmer tool, select the device name from the **SELECT DEVICE** drop-down list.



5. Select **Add file** and browse the downloaded file to upload the desired sample app hex file.



6. Select Erase & write to flash the hex file on the device.



7. Check the log for successful flash.



8. Connect the nRF52840-Dongle to the USB port of the Raspberry Pi having the latest TH.
9. For the Thread DUT, enable discoverable over Bluetooth LE (e.g., on nRF52840 DK: select Button 4) and start the Thread Setup Test execution by referring to [Section 8, Test Configuration](#).

5.2.2.2. Instructions to Set Up nRF52840-DK Using Docker Environment

1. To build the sample apps for nRF-Connect, check out the Matter repository and bootstrap using following commands:

```
git clone https://github.com/project-chip/connectedhomeip.git
cd ~/connectedhomeip/
source scripts/bootstrap.sh
cd ~/connectedhomeip/
source scripts/activate.sh
```

2. If the nRF-Connect SDK is not installed, create a directory running the following command:

```
$ mkdir ~/nrfconnect
```

3. Download the latest version of the nRF-Connect SDK Docker image by running the following command:

```
$ sudo docker pull nordicsemi/nrfconnect-chip
```

4. Start Docker using the downloaded image by running the following command:

```
sudo docker run --rm -it -e RUNAS=$(id -u) -v ~/nrfconnect:/var/ncs -v
~/connectedhomeip:/var/chip -v /dev/bus/usb:/dev/bus/usb --device-cgroup-rule "c 189:*
rmw" nordicsemi/nrfconnect-chip
```

5. The following commands can be executed to change the settings if required:

`~/nrfconnect` can be replaced with an absolute path to the nRF-Connect SDK source directory.
`~/connectedhomeip` can be replaced with an absolute path to the CHIP source directory.

```
-v /dev/bus/usb:/dev/bus/usb --device-cgroup-rule "c 189: rmw"*
```



Parameters can be omitted if flashing the example app onto the hardware is not required. This parameter gives the container access to USB devices connected to your computer such as the nRF52840 DK.

`--rm` can be omitted if you do not want the container to be auto-removed when you exit the container shell session.

`-e RUNAS=$(id -u)` is needed to start the container session as the current user instead of root.

6. Update the nRF-Connect SDK to the most recent supported revision, by running the following command:

```
$ cd /var/chip  
$ python3 scripts/setup/nrfconnect/update_ncs.py --update
```

5.2.2.3. Building and Flashing Sample Apps for nRF-Connect

Perform the following procedure, regardless of the method used for setting up the environment:

1. Navigate to the example directory:

```
$ cd examples/all-clusters-app/nrfconnect
```

2. Before building, remove all build artifacts by running the following command:

```
$ rm -r build
```

3. Run the following command to build the example, with **build-target** replaced with the build target name of the Nordic Semiconductor's kit, for example, nrf52840dk_nrf52840:

```
$ west build -b <build-target> --pristine always -DCONFIG_CHIP_LIB_SHELL=y
```

Target Name	Compatible Kit
nRF52840 DK	nrf52840dk_nrf52840
nRF5340 DK	nrf5340dk_nrf5340_cmuapp
nRF52840 Dongle	nrf52840dongle_nrf52840
nRF7002 DK	nrf7002dk_nrf5340_cmuapp

4. To flash the application to the device, use the west tool and run the following command from the example directory:

```
$ west flash --erase
```

5. Connect the nRF52840-Dongle to the USB port of the Raspberry Pi having the latest TH.
6. For the Thread DUT, enable discoverable over Bluetooth LE (e.g., On nRF52840 DK: Press Button 4) and start the Thread Setup Test execution by referring to [Section 8, Test Configuration](#).

6. Bringing up the Matter Python REPL

The [Matter Python REPL](#), also known as `chip-repl`, is a native IPython shell environment loaded with a Python-wrapped version of the C++ Matter stack to permit interacting as a controller to other Matter-compliant devices.

You can use the `chip-cert-bins` SDK image to run `chip-repl` on your Test Harness by following these instructions:

- Start container:

Remember to set `PATH_TO_PAA_ROOTS` and substitute `<SDK SHA RECOMMENDED>`

```
docker run -v $PATH_TO_PAA_ROOTS:/paa_roots -v  
/var/run/dbus/system_bus_socket:/var/run/dbus/system_bus_socket -v /home/ubuntu/certification-  
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/python_testing:/root/python_testin  
g -v $(pwd):/launch_dir --privileged --network host -it connectedhomeip/chip-cert-bins:<SDK  
SHA RECOMMENDED>
```

- Activate python environment:

```
source python_env/bin/activate
```

- Run chip-repl:

```
python3 python_env/bin/chip-repl
```

7. OT Border Router (OTBR) Setup

If the DUT supports Thread Transport, DUT vendors need to use the OTBR that is shipped with the TH for certification testing. Here are the instructions to set up OTBR that comes with the TH. Users need to get the RCP programmed with the recommended version and connect it to the Raspberry Pi running the TH. The OTBR will be started when the TH runs the thread transport related TC's.

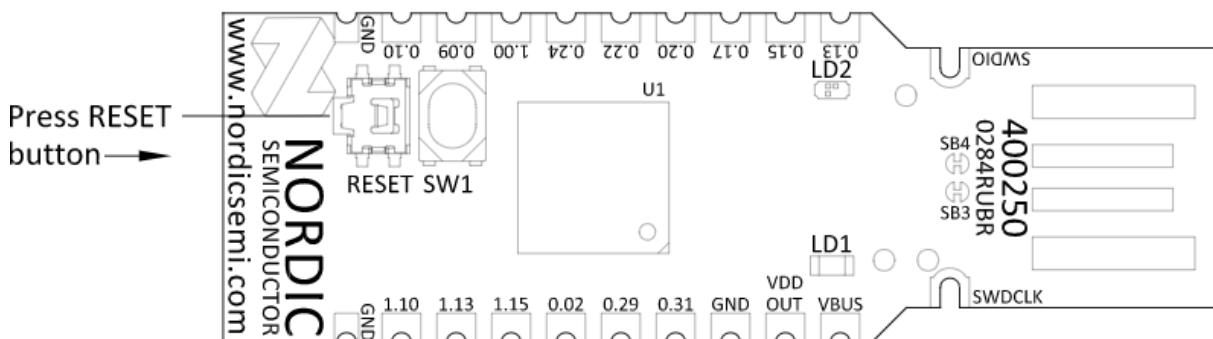
Currently the OTBR in the TH works with either the Nordic RCP dongle or SiLabs RCP dongle. Refer to [Section 7.1](#) to flash the NRF52840 firmware or [Section 7.2](#) to flash the SiLabs firmware and get the RCP's ready. Once the RCP's are programmed, the user needs to insert the RCP dongle on to the Raspberry Pi running the TH and reboot the Raspberry Pi.

7.1. Instructions to Flash the Firmware NRF52840 RCPDongle

1. Download RCP firmware package from the following link on the user's system — [Thread RCP Firmware Package](#)
2. nRF Util is a unified command line utility for Nordic products. For more details, refer to the following link— <https://www.nordicsemi.com/Products/Development-tools/nrf-util>
3. Install the nRF Util dependencies on the user's system using the following commands:

```
python3 -m pip install -U nrfutil  
nrfutil install nrf5sdk-tools
```

4. Connect the nRF52840 Dongle to the USB port of the user's system.
5. Press the Reset button on the dongle to enter the DFU mode (the red LED on the dongle starts blinking).



6. To install the RCP firmware package on to the dongle, run the following command from the path where the firmware package was downloaded:

```
nrfutil dfu usb-serial -pkg <FILE NAME> -p /dev/ttyACM0
```

Example:

```
nrfutil dfu usb-serial -pkg nrf52840dongle_rcp_c084c62.zip -p /dev/ttyACM0
```

7. Once the flash is successful, the red LED turns off slowly.
8. Remove the Dongle from the user's system and connect it to the Raspberry Pi running TH.
9. In case any permission issue occurs during flashing, launch the terminal and retry in sudo mode.

7.2. Nrfconnect Sample APPs Firmwares to Flash on the NRF52840DK Kit

The [Nrfconnect Sample apps binary Package](#) is available for download and should be flashed in the development kit NRF52840DK to use it as DUT in the Test-Harness tests.

7.3. Instructions to Flash SiLabs RCP

Download the latest version of ot-rpc-binaries from the assets list of the latest release: [Silicon Labs Matter GitHub](#)

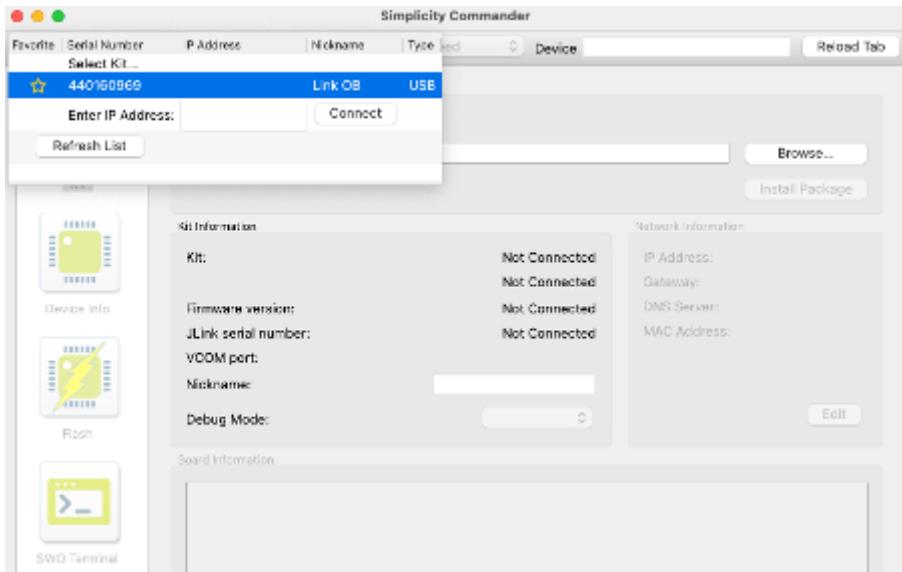
For detailed RCP firmware usage, refer to: <https://www.silabs.com/documents/public/application-notes/an1256-using-sl-rpc-with-openthread-border-router.pdf>

Requirements:

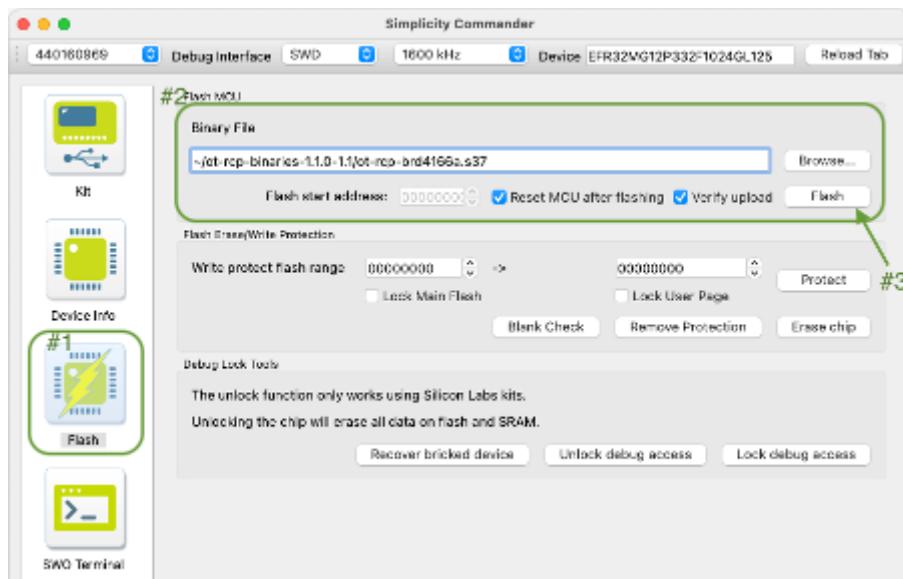
- SiLabs RCP: Thunderboard Sense 2 Sensor-to-Cloud Advanced IoT Kit or EFR32MG Wireless Starter Kit
- SiLabs RCP Firmware: See Session 6.2
- Simplicity Commander: Installer for [Windows](#), [MAC](#) or [Linux](#)

From UI:

- Connect the RCP dongle to the USB port of the user's operating system or via Ethernet.
- From the Simplicity Commander app, select and connect to RCP:
 - For USB connection, select the corresponding Serial Number from the drop-down list.
 - For Ethernet connection, enter the IP address of the RCP and click on **Connect**.



- To flash an image, go to "Flash", select the RCP binary file, and click on **Flash**.



From CLI:

- In case RCP is connected via Ethernet and the Simplicity Commander UI is not an option, the RCP image can be flashed using CLI.
- From path to Simplicity Commander:
`commander flash <rcp-image-path> --ip <rcp-ip-address>`

7.4. Forming Thread Network and Generating Dataset for Thread Pairing

TH spins the OTBR docker image automatically when executing the thread related test cases. Follow the steps below if the user wants to start OTBR with custom parameters. The user needs to generate a dataset for the custom OTBR. To generate hexadecimal code required for manual Thread pairing procedure, use the instructions below.

ssh the Raspberry-Pi in the User System using the command "`ssh ubuntu@IP_address`"

Example output for the above command to generate the dataset value:

```

ubuntu@ubuntu:~$ ./certification-tool/backend/test_collections/matter/scripts/OTBR/otbr_start.sh
nrfconnect/otbr 9185bda 083c8472bc52 10 months ago 1.21GB
otbr image nrfconnect/otbr:9185bda already installed
54d868724cbb0c05c155983d5df5e9a3c1b61cbdafdf38eef2d8d1928f305a

waiting 10 seconds to give the docker container enough time to start up...
Param: 'dataset init new'
Done
Param: 'dataset channel 25'
Done
Param: 'dataset panid 0x5b35'
Done
Param: 'dataset extpanid 5b35dead5b35beef'
Done
Param: 'dataset networkname 5b35'
Done
Param: 'dataset commit active'
Done
Param: 'prefix add fd11:35::/64 pasor'
Done
Param: 'ifconfig up'
Done
Param: 'thread start'
Done
Param: 'netdata register'
Done
Param: 'dataset active -x
0e0800000000000010000000300001935060004001ffe002085b35dead5b35beef0708fd902fb12bca8af9
051000112233445566778899aabbccddeeff03043562333501025b350410cdfe3b9ac95afd445e659161b
03b3c4a0c0402a0f7f8
Done
Simple Dataset:
000300001902085b35dead5b35beef051000112233445566778899aabbccddeeff01025b35

```

If any issue occurs while using **otbr_start.sh**, follow the steps below to generate the dataset value manually:

On Terminal 1:

1. Follow the steps below to build the OTBR docker:
 - a. Create the docker network by executing the following commands:

```

sudo docker network create --ipv6 --subnet fd11:db8:1::/64 -o
com.docker.network.bridge.name=otbr0 otbr
sudo sysctl net.ipv6.conf.otbr0.accept_ra_rt_info_max_plen=128
sudo sysctl net.ipv6.conf.otbr0.accept_ra=2

```

- b. Run the dependency:

```
sudo modprobe ip6table_filter
```

- c. Run the docker:

```

sudo docker run -it --rm --privileged --network otbr -p 8080:80 --sysctl
"net.ipv6.conf.all.disable_ipv6=0 net.ipv6.conf.all.forwarding=1" --name otbr -e
NAT64=0 --volume /dev/ttyACM0:/dev/ttyACM0 nrfconnect/otbr:9185bda --radio-url
spinel+hdlc+uart:///dev/ttyACM0

```

2. Generate the Thread form for dataset by entering ‘<Raspberry-Pi IP>:8080’ on the user’s system browser. The OTBR form will be generated as shown below.
3. Click on the **Form** option and follow the sequence to generate the OTBR form.

The screenshot shows a web browser window with the URL `192.168.3.176:8080`. The page title is "OT Border Router". On the left, there's a sidebar with the following menu items: Home, Join, Form (which is highlighted with a blue background), Status, Settings, Commission, and Topology. The main content area features the "OPEN THREAD" logo with "released by Nest" underneath. Below the logo, the text "What is OpenThread?" is displayed. A detailed description follows, mentioning it's an open-source implementation of the Thread networking protocol, designed to be OS and platform agnostic, and a Thread Certified Component.

The screenshot shows the "Form" screen for creating a Thread Network. The left sidebar shows the "Form" option selected. The main form fields are as follows: Network Name is "OpenThreadDemo", Network Extended PAN ID is "1111111122222222", PAN ID is "0x1234", Passphrase/Commissioner Credential is "j01Nm", Network Key is "00112233445566778899aabbccddeeff", On-Mesh Prefix is "fd11:22::", and the Channel is set to "15". There is also a checked checkbox for "Default Route". At the bottom of the form is a large blue "FORM" button.

On Terminal 2:

1. Generation of Hex Code:

Obtain the dataset hex value by running the following command:

```
sudo docker exec -ti otbr ot-ctl dataset active -x
```

Example hex code :

```
0e08000000000000100000030000f35060004001ffffe0020811111112222220708fdabd97fc1941f290510  
00112233445566778899aabbccddeeff030e4f70656e54687265616444656d6f010212340410445f2b5ca6f2a9  
3a55ce570a70efeeecb0c0402a0f7f8
```

2. The above generated sample pairing code can be used during the manual Thread pairing procedure with the following command:

```
./chip-tool pairing ble-thread <node-id> hex:<dataset hex value> <setup-pin>  
<discriminator>  
./chip-tool pairing ble-thread 97  
hex:0e08000000000000100000030000f35060004001ffffe0020811111112222220708fd882e3d3a7373dc  
051000112233445566778899aabbccddeeff030f4f70656e54687265616444656d70790102123404101570fcfd  
6de18b3d78d6d39881a8a5710c0402a0f7f8 20202021 3840
```

7.5. Troubleshooting: Boarder Router Container failure to initialize

1. Error message: (Example)

```
Error occurred during setup of test suite.FirstChipToolSuite. 409 Client Error for  
http+docker://localhost/v1.42/containers/10ad48500522af3d5a23c181a6018053248250b958a353  
ed88d5a5f538dcbf33/exec: Conflict ("Container  
10ad48500522af3d5a23c181a6018053248250b958a353ed88d5a5f538dcbf33 is not running")
```

Solution:

- a. Check for the presence of rogue executions of the otbr-chip container. Using command:

```
$docker ps
```

Stop any running otbr-chip containers from the result.

```
$docker container stop <container_id>
```

- b. Check host (**raspberry**) network configuration interface's ip address does not conflict with **otbr-chip** default interface ip address.

Conflicting network configuration could be pointed out by checking container's initialization log.

```
$docker logs <container_id>
```

Example Log Output:

```
...
+ service tayga start
* Starting userspace NAT64 tayga
RTNETLINK answers: File exists
RTNETLINK answers: File exists
RTNETLINK answers: File exists
RTNETLINK answers: File exists
...fail!
+ die 'Failed to start tayga'
+ echo '* ERROR: Failed to start tayga'
* ERROR: Failed to start tayga
+ exit 1
tail: cannot open '/var/log/syslog' for reading: No such file or directory
tail: no files remaining
```

Default Tayga interface address:

```
ipv4-addr 192.168.255.1 # This address could be checked on /etc/tayga.conf on otbr-chip
container
```

Use command below on host (**raspberrypi**) to check interface's ip addresses

```
$ifconfig
...
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.2.2 netmask 255.255.255.0 broadcast 192.168.2.255 inet6
fdcb:377:2b62:f8fd:dea6:32ff:fe94:c54c prefixlen 64 scopeid 0x0<global> inet6
fe80::dea6:32ff:fe94:c54c prefixlen 64 scopeid 0x20<link> ether dc:a6:32:94:c5:4c
txqueuelen 1000 (Ethernet) RX packets 250969 bytes 184790487 (184.7 MB) RX errors 0
dropped 0 overruns 0 frame 0 TX packets 125202 bytes 85904550 (85.9 MB) TX errors 0
dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10<host> loop txqueuelen
1000 (Local Loopback) RX packets 520 bytes 48570 (48.5 KB) RX errors 0 dropped 0
overruns 0 frame 0 TX packets 520 bytes 48570 (48.5 KB) TX errors 0 dropped 0 overruns 0
carrier 0 collisions 0
```

If any interface matches tayga ip address, change the conflicting IP on host.

8. Test Configuration

8.1. Project Configuration

When the DUT is a client, refer to [Simulated Tests](#). The TH brings up the example accessory using chip-app1 binary. The user will be prompted to commission the device. Once the commissioning process is completed, proceed with the test execution.

In the case where the DUT is a server, the TH spins up the controller, the DUT bring-up procedure should be completed and has to be paired with the controller.

Depending on the DUT's network transport, any one of the appropriate pairing modes can be opted:

- ‘**ble-wifi**’ to complete the pairing for the DUT using BLE Wi-Fi
- ‘**onnetwork**’ to complete the pairing for the DUT that is already on the operational network (e.g., the device is already present on the same Ethernet network of the TH) connection
- ‘**ble-thread**’ to complete the pairing for the Thread Device

Follow the sections below for the project configuration and test execution.

8.1.1. Projects Menu

1. Open a Web browser from the user's system and enter the IP address of the Raspberry Pi as given in [Section 4.1.2, TH Installation on Raspberry Pi](#).
2. In case the TH user interface does not launch, refer to [Section 4.2.3, Bringing Up of Docker Containers Manually](#).



3. A new window will be opened as "Matter Test Harness".
4. Click on the **Create New Project** button. Enter the project name as "Test Project" and edit the Project Config settings to provide additional details.

```
{
  "test_parameters": null,
  "network": {
    "wifi": {
      "ssid": "testharness",
      "password": "wifi-password"
    },
    "thread": {
      "rcp_serial_path": "/dev/ttyACM0",
      "rcp_baudrate": 115200,
      "on_mesh_prefix": "fd11:22::/64",
      "network_interface": "eth0",
      "dataset": {
        "channel": "15",
        "panid": "0x1234",
        "extpanid": "1111111122222222",
        "networkkey": "00112233445566778899aabbcdddeeff",
        "networkname": "DEMO"
      },
      "otbr_docker_image": null
    }
  },
  "dut_config": {
    "discriminator": "3840",
    "setup_code": "20202021",
    "pairing_mode": "onnetwork",
    "chip_timeout": null,
    "chip_use_paa_certs": false,
    "trace_log": true
  }
}
```

8.1.2. Wi-Fi Mode

- Configure the DUT by providing details like discriminator, setup_code and set the **pairing_mode** as "**ble-wifi**".

```
"dut_config": {
  "discriminator": "3840",
  "setup_code": "20202021",
  "pairing_mode": "ble-wifi",
  "chip_timeout": null,
  "chip_use_paa_certs": false,
  "trace_log": true
}
```

- To pair in the BLE Wi-Fi mode, configure the Network settings by providing the ssid and password.

```

"network": {
    "wifi": {
        "ssid": "testharness",
        "password": "wifi-password"
    },
    ...
}

```

8.1.2.1. On Network Mode

- If the DUT is already present on the operational network (e.g., connected to the same network as the controller via Ethernet) then the user can select this mode.
- Configure the DUT by providing details like discriminator, setup_code and set the **pairing_mode** as "**onnetwork**".

```

"dut_config": {
    "discriminator": "3840",
    "setup_code": "20202021",
    "pairing_mode": "onnetwork",
    "chip_timeout": null,
    "chip_use_paa_certs": false,
    "trace_log": true
}

```

8.1.3. Thread Device Mode

- Input the DUT configuration details like discriminator: "3840", setup_code:"20202021", and **pairing_mode** as "**ble-thread**".

```

"dut_config": {
    "discriminator": "3840",
    "setup_code": "20202021",
    "pairing_mode": "ble-thread",
    "chip_timeout": null,
    "chip_use_paa_certs": false,
    "trace_log": true
}

```

- The TH loads the default thread configuration values that match the OTBR built on the TH. The following configuration can be customized as per the user's need.

```

"thread": {
    "rcp_serial_path": "/dev/ttyACM0",
    "rcp_baudrate": 115200,
    "on_mesh_prefix": "fd11:22::/64",
    "network_interface": "eth0",
}

```

```
"dataset": {  
    "channel": "15",  
    "panid": "0x1234",  
    "extpanid": "1111111122222222",  
    "networkkey": "00112233445566778899aabbccddeeff",  
    "networkname": "DEMO"  
},  
    "otbr_docker_image": null  
}
```



The OTBR docker is contained in the TH and runs automatically upon the start of the TH tool.

- c. If using an already configured Thread network with a Thread Border router present on the same network as the TH, it is possible to provide an explicit operational data configuration so that it is used instead of locally configuring a new Thread PAN/

```
"thread": {  
    "operational_dataset_hex":  
    "0e0800000000001000035060004001ffffe00708fd5270f26ee4c02c041064dc641d7195508d7cd17c  
    e22db711420c0402a0f7f8000300000f0102123402081111112222222030444454d4f05100011223  
    3445566778899aabbccddeeff"  
}
```



OTBR needs to be configured and running. TH will not start any OTBR docker containers.

8.1.4. PAA Certificates

For the case that the DUT requires a PAA certificate to perform a pairing operation, input "true" for the flag "chip_tool_use_paa_certs" to configure the Test-Harness to use them.

```
"dut_config": {  
    "discriminator": "3840",  
    "setup_code": "20202021",  
    "pairing_mode": "onnetwork",  
    "chip_timeout": null,  
    "chip_use_paa_certs": true,  
    "trace_log": true  
}
```



Make sure to include the desired PAA certificates in the default path "/var/paa-root-certs/", in the Raspberry-Pi.

8.2. Test Parameters

- a. Input the test parameters like endpoint on the DUT where the cluster to be tested is implemented.

```
"test_parameters": {  
    "endpoint": 5  
}
```

- b. "qr-code" and "manual-code" parameters:

Only one of the following parameter is allowed, also when one of them is configured, the TH will not send "passcode" and "discriminator" (from "dut_config") arguments to DUT.

- i. "qr-code" parameter example:

```
"test_parameters": {  
    "qr-code": "MT:-24J042C00KA0648G00"  
}
```

- ii. "manual-code" parameter example:

```
"test_parameters": {  
    "manual-code": "34970112332"  
}
```

- iii. **Invalid** configuration: "manual-code" and "qr-code" together:

```
"test_parameters": {  
    "qr-code": "MT:-24J042C00KA0648G00"  
    "manual-code": "34970112332"  
}
```



This is an invalid configuration. TH will not accept both parameters set at the same time.

- c. Overwrite the default timeout. Value in [s]:

```
"test_parameters": {  
    "timeout": 300  
}
```

On completion of the "network" and the "dut_config" configuration, select the **Update** and then **Create** button to create the Test Project.

8.2.1. Upload PICS File

The newly created project will be listed under the Project details column.

Click on the Edit option to configure the project to load the required PICS file for the cluster to be tested and select the **Update** button. Refer to [Section 9, Test Case Execution](#).

Version: TH Fall2023
Sha: 1c80214a

Projects + Add Project

Project details

Test Project Created Just now

Showing 1 to 1 of 1 entries

Edit

Version: TH Fall2023
Sha: 1c80214a

Search...

Edit Project

Project Name

Test Project

Project Config

```
{
  "config": {
    "network": {
      "wifi": {
        "ssid": "testharness",
        "password": "wifi-password"
      },
      "thread": {
        "rcp_serial_path": "/dev/ttyACM0",
        "rcp_baudrate": 115200
      }
    }
  }
}
```

PICS

Upload PICS

Door lock Test Plan

Cancel Update

8.2.2. Test Menu

1. Now the Test Project is ready for execution. Click on the **Go To Test-Run** icon and create a new Test Run batch.

The screenshot shows the 'Projects' page of the Connectivity Standards Alliance Test interface. On the left is a dark sidebar with icons for Home, Test, Utility, and Settings, and a 'Project' button at the bottom. The main area has a light blue header with 'Projects' and '+ Add Project'. Below is a table with one entry: 'Test Project' created 'Just now'. A search bar and a 'Go To Test-Run' button are at the bottom right. The footer says '© Connectivity Standards Alliance'.

The screenshot shows the 'Test Project' details page. The sidebar is identical to the previous screenshot. The main area has a large '+' icon and the text 'Create New Test Run'. Below it is a circular arrow icon with a downward arrow and the text 'Import Test Run'. The footer says '© Connectivity Standards Alliance'.

2. A Test Run can be created in Regular Mode or Certification Mode. The test cases are automatically selected based on the PICS files provided in the Project Configuration. For a Test Run in Regular Mode, it is possible to change this selection, but in Certification Mode that selection is unchangable—a test case must be executed if and only if the PICS files indicate that it is applicable.



3. Provide a Test name for this run such as Door Lock First Run. Input any additional description about the run. Enter the Test Engineers Name under Operator. Select only the test cases that are

to be executed and deselect other test cases. There is a search option available to search for a particular test case. The number of times the test is to be executed can be given by clicking on the number spin control.

Ensure that DUT is in the discoverable mode before clicking on the Start button.

Example command to be used to launch the sample apps (e.g., all-cluster-app):

Ble-wifi: ./chip-all-clusters-app --wifi

Onnetwork: ./chip-all-clusters-app

Thread: Enable discoverable over Bluetooth LE (ex: On nRF52840 DK: Press Button 4 to start BLE advertisements)

The screenshot shows the 'Test Project' interface with a 'New Test Run' dialog open. The 'Test Details' section includes a 'Test Name' field set to 'Door Lock First Run' and a 'Description' field set to 'Run 2.2 and 2.3'. An 'Operator' field is set to 'Alpha'. The 'Test Cases' section shows two tabs: 'python_tests' and 'SDK YAML Tests', with 'SDK YAML Tests' selected. Under 'Test Suites', there are three entries: 'FirstManualSuite' (unchecked), 'FirstChipToolSuite' (unchecked, with a count of 07), and 'FirstAppSuite' (unchecked). Under 'Test Cases', there is a list of 11 test cases: TC-DRLK-1.1, TC-DRLK-2.1 (Semi-automated), TC-DRLK-2.4, TC-DRLK-2.5, TC-DRLK-2.6, TC-DRLK-2.7, TC-DRLK-2.8, TC-DRLK-2.9, and TC-DRLK-2.11. Each test case has a spin control next to it. A search bar at the top right contains the text 'drlk'. To the right, a 'Summary (01)' panel shows the structure of the selected test cases. At the bottom, there are 'Start' and 'Clear Selection' buttons.

4. Click on the **Start** button for the test execution. Note that the test execution gets started and the log window appears. Click on the **Abort** button to stop the test execution.

Test Project

Test Execution

Test Cases

- SDK YAML Tests [01]
- FirstChipToolSuite [07]

Logs

```
[1697573212.617460][7:7] CHIP:IN: SecureSession[0x558eac328e00]: Allocated Type:2 LSID:32748
[1697573212.617491][7:7] CHIP:SC: Allocated SecureSession (0x558eac328e00) - waiting for Sigma1 msg
[1697573212.617526][7:7] CHIP:CTL: System State Initialized...
[1697573212.617602][7:7] CHIP:CTL: Setting attestation nonce to random value
[1697573212.617643][7:7] CHIP:CTL: Setting CSR nonce to random value
[1697573212.617671][7:7] CHIP:IN: UDP::init bind&listen port=5550
[1697573212.617764][7:7] CHIP:IN: UDP::init bound to port=5550
[1697573212.617777][7:7] CHIP:IN: TransportMgr initialized
[1697573212.618154][7:15] CHIP:DL: CHIP task running
[1697573212.621156][7:15] CHIP:DL: HandlePlatformSpecificBLEEvent 32786
[1697573212.636692][7:7] CHIP:TOOL: LWS_CALLBACK_GET_THREAD_ID
[1697573212.636989][7:7] CHIP:TOOL: LWS_CALLBACK_PROTOCOL_INIT
Create PICS file for DUT
Sending command: /bin/sh -c "echo 'DRLK.S=1 DRLK.C=1 DRLK.S.A0000=1 DRLK.S.A0001=1 DRLK.S.A0002=1 DRLK.S.A0003=1 DRLK.S.A0004=0 DRLK.S.A0005=0 DRLK.S.A0006=0 DRLK.S.A0007=1 DRLK.S.A0008=2 DRLK.S.A0009=3 DRLK.S.A0010=4 DRLK.S.A0011=5 DRLK.S.A0012=6 DRLK.S.A0013=7 DRLK.S.A0014=8 DRLK.S.A0015=9 DRLK.S.A0016=10 DRLK.S.A0017=11 DRLK.S.A0018=12 DRLK.S.A0019=13 DRLK.S.A0020=14 DRLK.S.A0021=15 DRLK.S.A0022=16 DRLK.S.A0023=17 DRLK.S.A0024=18 DRLK.S.A0025=19 DRLK.S.A0026=20 DRLK.S.A0027=21 DRLK.S.A0028=22 DRLK.S.A0029=23 DRLK.S.A0030=24 DRLK.S.A0031=25 DRLK.S.A0032=26 DRLK.S.A0033=27 DRLK.S.A0034=28 DRLK.S.A0035=29 DRLK.S.A0036=30 DRLK.S.A0037=31 DRLK.S.A0038=32 DRLK.S.A0039=33 DRLK.S.A0040=34 DRLK.S.A0041=35 DRLK.S.A0042=36 DRLK.S.A0043=37 DRLK.S.A0044=38 DRLK.S.A0045=39 DRLK.S.A0046=40 DRLK.S.A0047=41 DRLK.S.A0048=42 DRLK.S.A0049=43 DRLK.S.A0050=44 DRLK.S.A0051=45 DRLK.S.A0052=46 DRLK.S.A0053=47 DRLK.S.A0054=48 DRLK.S.A0055=49 DRLK.S.A0056=50 DRLK.S.A0057=51 DRLK.S.A0058=52 DRLK.S.A0059=53 DRLK.S.A0060=54 DRLK.S.A0061=55 DRLK.S.A0062=56 DRLK.S.A0063=57 DRLK.S.A0064=58 DRLK.S.A0065=59 DRLK.S.A0066=60 DRLK.S.A0067=61 DRLK.S.A0068=62 DRLK.S.A0069=63 DRLK.S.A0070=64 DRLK.S.A0071=65 DRLK.S.A0072=66 DRLK.S.A0073=67 DRLK.S.A0074=68 DRLK.S.A0075=69 DRLK.S.A0076=70 DRLK.S.A0077=71 DRLK.S.A0078=72 DRLK.S.A0079=73 DRLK.S.A0080=74 DRLK.S.A0081=75 DRLK.S.A0082=76 DRLK.S.A0083=77 DRLK.S.A0084=78 DRLK.S.A0085=79 DRLK.S.A0086=80 DRLK.S.A0087=81 DRLK.S.A0088=82 DRLK.S.A0089=83 DRLK.S.A0090=84 DRLK.S.A0091=85 DRLK.S.A0092=86 DRLK.S.A0093=87 DRLK.S.A0094=88 DRLK.S.A0095=89 DRLK.S.A0096=90 DRLK.S.A0097=91 DRLK.S.A0098=92 DRLK.S.A0099=93 DRLK.S.A0100=94 DRLK.S.A0101=95 DRLK.S.A0102=96 DRLK.S.A0103=97 DRLK.S.A0104=98 DRLK.S.A0105=99 DRLK.S.A0106=100 DRLK.S.A0107=101 DRLK.S.A0108=102 DRLK.S.A0109=103 DRLK.S.A0110=104 DRLK.S.A0111=105 DRLK.S.A0112=106 DRLK.S.A0113=107 DRLK.S.A0114=108 DRLK.S.A0115=109 DRLK.S.A0116=110 DRLK.S.A0117=111 DRLK.S.A0118=112 DRLK.S.A0119=113 DRLK.S.A0120=114 DRLK.S.A0121=115 DRLK.S.A0122=116 DRLK.S.A0123=117 DRLK.S.A0124=118 DRLK.S.A0125=119 DRLK.S.A0126=120 DRLK.S.A0127=121 DRLK.S.A0128=122 DRLK.S.A0129=123 DRLK.S.A0130=124 DRLK.S.A0131=125 DRLK.S.A0132=126 DRLK.S.A0133=127 DRLK.S.A0134=128 DRLK.S.A0135=129 DRLK.S.A0136=130 DRLK.S.A0137=131 DRLK.S.A0138=132 DRLK.S.A0139=133 DRLK.S.A0140=134 DRLK.S.A0141=135 DRLK.S.A0142=136 DRLK.S.A0143=137 DRLK.S.A0144=138 DRLK.S.A0145=139 DRLK.S.A0146=140 DRLK.S.A0147=141 DRLK.S.A0148=142 DRLK.S.A0149=143 DRLK.S.A0150=144 DRLK.S.A0151=145 DRLK.S.A0152=146 DRLK.S.A0153=147 DRLK.S.A0154=148 DRLK.S.A0155=149 DRLK.S.A0156=150 DRLK.S.A0157=151 DRLK.S.A0158=152 DRLK.S.A0159=153 DRLK.S.A0160=154 DRLK.S.A0161=155 DRLK.S.A0162=156 DRLK.S.A0163=157 DRLK.S.A0164=158 DRLK.S.A0165=159 DRLK.S.A0166=160 DRLK.S.A0167=161 DRLK.S.A0168=162 DRLK.S.A0169=163 DRLK.S.A0170=164 DRLK.S.A0171=165 DRLK.S.A0172=166 DRLK.S.A0173=167 DRLK.S.A0174=168 DRLK.S.A0175=169 DRLK.S.A0176=170 DRLK.S.A0177=171 DRLK.S.A0178=172 DRLK.S.A0179=173 DRLK.S.A0180=174 DRLK.S.A0181=175 DRLK.S.A0182=176 DRLK.S.A0183=177 DRLK.S.A0184=178 DRLK.S.A0185=179 DRLK.S.A0186=180 DRLK.S.A0187=181 DRLK.S.A0188=182 DRLK.S.A0189=183 DRLK.S.A0190=184 DRLK.S.A0191=185 DRLK.S.A0192=186 DRLK.S.A0193=187 DRLK.S.A0194=188 DRLK.S.A0195=189 DRLK.S.A0196=190 DRLK.S.A0197=191 DRLK.S.A0198=192 DRLK.S.A0199=193 DRLK.S.A0200=194 DRLK.S.A0201=195 DRLK.S.A0202=196 DRLK.S.A0203=197 DRLK.S.A0204=198 DRLK.S.A0205=199 DRLK.S.A0206=200 DRLK.S.A0207=201 DRLK.S.A0208=202 DRLK.S.A0209=203 DRLK.S.A0210=204 DRLK.S.A0211=205 DRLK.S.A0212=206 DRLK.S.A0213=207 DRLK.S.A0214=208 DRLK.S.A0215=209 DRLK.S.A0216=210 DRLK.S.A0217=211 DRLK.S.A0218=212 DRLK.S.A0219=213 DRLK.S.A0220=214 DRLK.S.A0221=215 DRLK.S.A0222=216 DRLK.S.A0223=217 DRLK.S.A0224=218 DRLK.S.A0225=219 DRLK.S.A0226=220 DRLK.S.A0227=221 DRLK.S.A0228=222 DRLK.S.A0229=223 DRLK.S.A0230=224 DRLK.S.A0231=225 DRLK.S.A0232=226 DRLK.S.A0233=227 DRLK.S.A0234=228 DRLK.S.A0235=229 DRLK.S.A0236=230 DRLK.S.A0237=231 DRLK.S.A0238=232 DRLK.S.A0239=233 DRLK.S.A0240=234 DRLK.S.A0241=235 DRLK.S.A0242=236 DRLK.S.A0243=237 DRLK.S.A0244=238 DRLK.S.A0245=239 DRLK.S.A0246=240 DRLK.S.A0247=241 DRLK.S.A0248=242 DRLK.S.A0249=243 DRLK.S.A0250=244 DRLK.S.A0251=245 DRLK.S.A0252=246 DRLK.S.A0253=247 DRLK.S.A0254=248 DRLK.S.A0255=249 DRLK.S.A0256=250 DRLK.S.A0257=251 DRLK.S.A0258=252 DRLK.S.A0259=253 DRLK.S.A0260=254 DRLK.S.A0261=255 DRLK.S.A0262=256 DRLK.S.A0263=257 DRLK.S.A0264=258 DRLK.S.A0265=259 DRLK.S.A0266=260 DRLK.S.A0267=261 DRLK.S.A0268=262 DRLK.S.A0269=263 DRLK.S.A0270=264 DRLK.S.A0271=265 DRLK.S.A0272=266 DRLK.S.A0273=267 DRLK.S.A0274=268 DRLK.S.A0275=269 DRLK.S.A0276=270 DRLK.S.A0277=271 DRLK.S.A0278=272 DRLK.S.A0279=273 DRLK.S.A0280=274 DRLK.S.A0281=275 DRLK.S.A0282=276 DRLK.S.A0283=277 DRLK.S.A0284=278 DRLK.S.A0285=279 DRLK.S.A0286=280 DRLK.S.A0287=281 DRLK.S.A0288=282 DRLK.S.A0289=283 DRLK.S.A0290=284 DRLK.S.A0291=285 DRLK.S.A0292=286 DRLK.S.A0293=287 DRLK.S.A0294=288 DRLK.S.A0295=289 DRLK.S.A0296=290 DRLK.S.A0297=291 DRLK.S.A0298=292 DRLK.S.A0299=293 DRLK.S.A0200=294 DRLK.S.A0201=295 DRLK.S.A0202=296 DRLK.S.A0203=297 DRLK.S.A0204=298 DRLK.S.A0205=299 DRLK.S.A0206=300 DRLK.S.A0207=301 DRLK.S.A0208=302 DRLK.S.A0209=303 DRLK.S.A02010=304 DRLK.S.A02011=305 DRLK.S.A02012=306 DRLK.S.A02013=307 DRLK.S.A02014=308 DRLK.S.A02015=309 DRLK.S.A02016=310 DRLK.S.A02017=311 DRLK.S.A02018=312 DRLK.S.A02019=313 DRLK.S.A02020=314 DRLK.S.A02021=315 DRLK.S.A02022=316 DRLK.S.A02023=317 DRLK.S.A02024=318 DRLK.S.A02025=319 DRLK.S.A02026=320 DRLK.S.A02027=321 DRLK.S.A02028=322 DRLK.S.A02029=323 DRLK.S.A02030=324 DRLK.S.A02031=325 DRLK.S.A02032=326 DRLK.S.A02033=327 DRLK.S.A02034=328 DRLK.S.A02035=329 DRLK.S.A02036=330 DRLK.S.A02037=331 DRLK.S.A02038=332 DRLK.S.A02039=333 DRLK.S.A02040=334 DRLK.S.A02041=335 DRLK.S.A02042=336 DRLK.S.A02043=337 DRLK.S.A02044=338 DRLK.S.A02045=339 DRLK.S.A02046=340 DRLK.S.A02047=341 DRLK.S.A02048=342 DRLK.S.A02049=343 DRLK.S.A02050=344 DRLK.S.A02051=345 DRLK.S.A02052=346 DRLK.S.A02053=347 DRLK.S.A02054=348 DRLK.S.A02055=349 DRLK.S.A02056=350 DRLK.S.A02057=351 DRLK.S.A02058=352 DRLK.S.A02059=353 DRLK.S.A02060=354 DRLK.S.A02061=355 DRLK.S.A02062=356 DRLK.S.A02063=357 DRLK.S.A02064=358 DRLK.S.A02065=359 DRLK.S.A02066=360 DRLK.S.A02067=361 DRLK.S.A02068=362 DRLK.S.A02069=363 DRLK.S.A02070=364 DRLK.S.A02071=365 DRLK.S.A02072=366 DRLK.S.A02073=367 DRLK.S.A02074=368 DRLK.S.A02075=369 DRLK.S.A02076=370 DRLK.S.A02077=371 DRLK.S.A02078=372 DRLK.S.A02079=373 DRLK.S.A02080=374 DRLK.S.A02081=375 DRLK.S.A02082=376 DRLK.S.A02083=377 DRLK.S.A02084=378 DRLK.S.A02085=379 DRLK.S.A02086=380 DRLK.S.A02087=381 DRLK.S.A02088=382 DRLK.S.A02089=383 DRLK.S.A02090=384 DRLK.S.A02091=385 DRLK.S.A02092=386 DRLK.S.A02093=387 DRLK.S.A02094=388 DRLK.S.A02095=389 DRLK.S.A02096=390 DRLK.S.A02097=391 DRLK.S.A02098=392 DRLK.S.A02099=393 DRLK.S.A020100=394 DRLK.S.A020101=395 DRLK.S.A020102=396 DRLK.S.A020103=397 DRLK.S.A020104=398 DRLK.S.A020105=399 DRLK.S.A020106=400 DRLK.S.A020107=401 DRLK.S.A020108=402 DRLK.S.A020109=403 DRLK.S.A020110=404 DRLK.S.A020111=405 DRLK.S.A020112=406 DRLK.S.A020113=407 DRLK.S.A020114=408 DRLK.S.A020115=409 DRLK.S.A020116=410 DRLK.S.A020117=411 DRLK.S.A020118=412 DRLK.S.A020119=413 DRLK.S.A020120=414 DRLK.S.A020121=415 DRLK.S.A020122=416 DRLK.S.A020123=417 DRLK.S.A020124=418 DRLK.S.A020125=419 DRLK.S.A020126=420 DRLK.S.A020127=421 DRLK.S.A020128=422 DRLK.S.A020129=423 DRLK.S.A020130=424 DRLK.S.A020131=425 DRLK.S.A020132=426 DRLK.S.A020133=427 DRLK.S.A020134=428 DRLK.S.A020135=429 DRLK.S.A020136=430 DRLK.S.A020137=431 DRLK.S.A020138=432 DRLK.S.A020139=433 DRLK.S.A020140=434 DRLK.S.A020141=435 DRLK.S.A020142=436 DRLK.S.A020143=437 DRLK.S.A020144=438 DRLK.S.A020145=439 DRLK.S.A020146=440 DRLK.S.A020147=441 DRLK.S.A020148=442 DRLK.S.A020149=443 DRLK.S.A020150=444 DRLK.S.A020151=445 DRLK.S.A020152=446 DRLK.S.A020153=447 DRLK.S.A020154=448 DRLK.S.A020155=449 DRLK.S.A020156=450 DRLK.S.A020157=451 DRLK.S.A020158=452 DRLK.S.A020159=453 DRLK.S.A020160=454 DRLK.S.A020161=455 DRLK.S.A020162=456 DRLK.S.A020163=457 DRLK.S.A020164=458 DRLK.S.A020165=459 DRLK.S.A020166=460 DRLK.S.A020167=461 DRLK.S.A020168=462 DRLK.S.A020169=463 DRLK.S.A020170=464 DRLK.S.A020171=465 DRLK.S.A020172=466 DRLK.S.A020173=467 DRLK.S.A020174=468 DRLK.S.A020175=469 DRLK.S.A020176=470 DRLK.S.A020177=471 DRLK.S.A020178=472 DRLK.S.A020179=473 DRLK.S.A020180=474 DRLK.S.A020181=475 DRLK.S.A020182=476 DRLK.S.A020183=477 DRLK.S.A020184=478 DRLK.S.A020185=479 DRLK.S.A020186=480 DRLK.S.A020187=481 DRLK.S.A020188=482 DRLK.S.A020189=483 DRLK.S.A020190=484 DRLK.S.A020191=485 DRLK.S.A020192=486 DRLK.S.A020193=487 DRLK.S.A020194=488 DRLK.S.A020195=489 DRLK.S.A020196=490 DRLK.S.A020197=491 DRLK.S.A020198=492 DRLK.S.A020199=493 DRLK.S.A020100=494 DRLK.S.A020101=495 DRLK.S.A020102=496 DRLK.S.A020103=497 DRLK.S.A020104=498 DRLK.S.A020105=499 DRLK.S.A020106=500 DRLK.S.A020107=501 DRLK.S.A020108=502 DRLK.S.A020109=503 DRLK.S.A020110=504 DRLK.S.A020111=505 DRLK.S.A020112=506 DRLK.S.A020113=507 DRLK.S.A020114=508 DRLK.S.A020115=509 DRLK.S.A020116=510 DRLK.S.A020117=511 DRLK.S.A020118=512 DRLK.S.A020119=513 DRLK.S.A020120=514 DRLK.S.A020121=515 DRLK.S.A020122=516 DRLK.S.A020123=517 DRLK.S.A020124=518 DRLK.S.A020125=519 DRLK.S.A020126=520 DRLK.S.A020127=521 DRLK.S.A020128=522 DRLK.S.A020129=523 DRLK.S.A020130=524 DRLK.S.A020131=525 DRLK.S.A020132=526 DRLK.S.A020133=527 DRLK.S.A020134=528 DRLK.S.A020135=529 DRLK.S.A020136=530 DRLK.S.A020137=531 DRLK.S.A020138=532 DRLK.S.A020139=533 DRLK.S.A020140=534 DRLK.S.A020141=535 DRLK.S.A020142=536 DRLK.S.A020143=537 DRLK.S.A020144=538 DRLK.S.A020145=539 DRLK.S.A020146=540 DRLK.S.A020147=541 DRLK.S.A020148=542 DRLK.S.A020149=543 DRLK.S.A020150=544 DRLK.S.A020151=545 DRLK.S.A020152=546 DRLK.S.A020153=547 DRLK.S.A020154=548 DRLK.S.A020155=549 DRLK.S.A020156=550 DRLK.S.A020157=551 DRLK.S.A020158=552 DRLK.S.A020159=553 DRLK.S.A020160=554 DRLK.S.A020161=555 DRLK.S.A020162=556 DRLK.S.A020163=557 DRLK.S.A020164=558 DRLK.S.A020165=559 DRLK.S.A020166=560 DRLK.S.A020167=561 DRLK.S.A020168=562 DRLK.S.A020169=563 DRLK.S.A020170=564 DRLK.S.A020171=565 DRLK.S.A020172=566 DRLK.S.A020173=567 DRLK.S.A020174=568 DRLK.S.A020175=569 DRLK.S.A020176=570 DRLK.S.A020177=571 DRLK.S.A020178=572 DRLK.S.A020179=573 DRLK.S.A020180=574 DRLK.S.A020181=575 DRLK.S.A020182=576 DRLK.S.A020183=577 DRLK.S.A020184=578 DRLK.S.A020185=579 DRLK.S.A020186=580 DRLK.S.A020187=581 DRLK.S.A020188=582 DRLK.S.A020189=583 DRLK.S.A020190=584 DRLK.S.A020191=585 DRLK.S.A020192=586 DRLK.S.A020193=587 DRLK.S.A020194=588 DRLK.S.A020195=589 DRLK.S.A020196=590 DRLK.S.A020197=591 DRLK.S.A020198=592 DRLK.S.A020199=593 DRLK.S.A020100=594 DRLK.S.A020101=595 DRLK.S.A020102=596 DRLK.S.A020103=597 DRLK.S.A020104=598 DRLK.S.A020105=599 DRLK.S.A020106=600 DRLK.S.A020107=601 DRLK.S.A020108=602 DRLK.S.A020109=603 DRLK.S.A020110=604 DRLK.S.A020111=605 DRLK.S.A020112=606 DRLK.S.A020113=607 DRLK.S.A020114=608 DRLK.S.A020115=609 DRLK.S.A020116=610 DRLK.S.A020117=611 DRLK.S.A020118=612 DRLK.S.A020119=613 DRLK.S.A020120=614 DRLK.S.A020121=615 DRLK.S.A020122=616 DRLK.S.A020123=617 DRLK.S.A020124=618 DRLK.S.A020125=619 DRLK.S.A020126=620 DRLK.S.A020127=621 DRLK.S.A020128=622 DRLK.S.A020129=623 DRLK.S.A020130=624 DRLK.S.A020131=625 DRLK.S.A020132=626 DRLK.S.A020133=627 DRLK.S.A020134=628 DRLK.S.A020135=629 DRLK.S.A020136=630 DRLK.S.A020137=631 DRLK.S.A020138=632 DRLK.S.A020139=633 DRLK.S.A020140=634 DRLK.S.A020141=635 DRLK.S.A020142=636 DRLK.S.A020143=637 DRLK.S.A020144=638 DRLK.S.A020145=639 DRLK.S.A020146=640 DRLK.S.A020147=641 DRLK.S.A020148=642 DRLK.S.A020149=643 DRLK.S.A020150=644 DRLK.S.A020151=645 DRLK.S.A020152=646 DRLK.S.A020153=647 DRLK.S.A020154=648 DRLK.S.A020155=649 DRLK.S.A020156=650 DRLK.S.A020157=651 DRLK.S.A020158=652 DRLK.S.A020159=653 DRLK.S.A020160=654 DRLK.S.A020161=655 DRLK.S.A020162=656 DRLK.S.A020163=657 DRLK.S.A020164=658 DRLK.S.A020165=659 DRLK.S.A020166=660 DRLK.S.A020167=661 DRLK.S.A020168=662 DRLK.S.A020169=663 DRLK.S.A020170=664 DRLK.S.A020171=665 DRLK.S.A020172=666 DRLK.S.A020173=667 DRLK.S.A020174=668 DRLK.S.A020175=669 DRLK.S.A020176=670 DRLK.S.A020177=671 DRLK.S.A020178=672 DRLK.S.A020179=673 DRLK.S.A020180=674 DRLK.S.A020181=675 DRLK.S.A020182=676 DRLK.S.A020183=677 DRLK.S.A020184=678 DRLK.S.A020185=679 DRLK.S.A020186=680 DRLK.S.A020187=681 DRLK.S.A020188=682 DRLK.S.A020189=683 DRLK.S.A020190=684 DRLK.S.A020191=685 DRLK.S.A020192=686 DRLK.S.A020193=687 DRLK.S.A020194=688 DRLK.S.A020195=689 DRLK.S.A020196=690 DRLK.S.A020197=691 DRLK.S.A020198=692 DRLK.S.A020199=693 DRLK.S.A020100=694 DRLK.S.A020101=695 DRLK.S.A020102=696 DRLK.S.A020103=697 DRLK.S.A020104=698 DRLK.S.A020105=699 DRLK.S.A020106=700 DRLK.S.A020107=701 DRLK.S.A020108=702 DRLK.S.A020109=703 DRLK.S.A020110=704 DRLK.S.A020111=705 DRLK.S.A020112=706 DRLK.S.A020113=707 DRLK.S.A020114=708 DRLK.S.A020115=709 DRLK.S.A020116=710 DRLK.S.A020117=711 DRLK.S.A020118=712 DRLK.S.A020119=713 DRLK.S.A020120=714 DRLK.S.A020121=715 DRLK.S.A020122=716 DRLK.S.A020123=717 DRLK.S.A020124=718 DRLK.S.A020125=719 DRLK.S.A020126=720 DRLK.S.A020127=721 DRLK.S.A020128=722 DRLK.S.A020129=723 DRLK.S.A020130=724 DRLK.S.A020131=725 DRLK.S.A020132=726 DRLK.S.A020133=727 DRLK.S.A020134=728 DRLK.S.A020135=729 DRLK.S.A020136=730 DRLK.S.A020137=731 DRLK.S.A020138=732 DRLK.S.A020139=733 DRLK.S.A020140=734 DRLK.S.A020141=735 DRLK.S.A020142=736 DRLK.S.A020143=737 DRLK.S.A020144=738 DRLK.S.A020145=739 DRLK.S.A020146=740 DRLK.S.A020147=741 DRLK.S.A020148=742 DRLK.S.A020149=743 DRLK.S.A020150=744 DRLK.S.A020151=745 DRLK.S.A020152=746 DRLK.S.A020153=747 DRLK.S.A020154=748 DRLK.S.A020155=749 D
```

New Project ▾

Version: v2.11-beta1+fall2024 Sha: Unknown

Test ▾ + Add Test Import Test Run Certification Mode Search...

Test Name	Created	Success	Failure	Completion
UI_Test_Run_2024_08_01_20_53_48	39 Minutes Ago	1	NA	<div style="width: 100%;"> </div>

8.2.3. Utility Menu

- Click on **Utility Menu** to review the previous test report.

Version: TH Fall2023 Sha: 1c80214a

Utility

File Upload

Browse UI_Test_Run_2023_10_19_14_03_00-4.json

Matter Test Harness - Test Run Report

Print

Test Run Information

Test Name	Status	Date	Time	Operator
UI_Test_Run_2023_10_19_14_03_00	Passed	Oct 19, 2023	21:3:0 - 21:3:10 (0m10s)	Alpha

Result Statistics

Result Summary

S.No	Public ID	State	Time Elapsed
1	FirstChipToolSuite	Passed	0m10s
1.1	TC-DRLK-2.8	Passed	0m5s
1.1.1	Start chip-tool test	Passed	0m0s
1.1.2	Wait for the commissioned device to be established	Passed	0m0s

Time Elapsed

- Click on the **Browse** button to upload the previous report and select the desired log filter options. The console logger contains a filter drop-down list to select the different categories of logs to display. Use the **Print** button to print the test report.

8.2.4. Settings Menu

Click on the "Select theme" option drop-down to select the different theme for the user interface.

9. Test Case Execution

Refer to [Section 2, References](#) for PICS tool documentation to generate the PICS XML files.

PICS (*Protocol Implementation Conformance Statement*) is a list of features supported by a device as defined by a technology *protocol*, standard or specification. Each feature is known as a *PICS Item*, and device *implementation* is either mandatory or optional. PICS is used by the device manufacturer as a *statement of conformance to* a technology standard and a requirement for all CSA Product Certification programs.

PICS codes are generated from the Test Plans. The Base.xml file lists all the Core feature PICS from the Matter Base Specifications and the application cluster PICS are listed in the respective TestPlan.xml files. Follow the steps below to generate and upload the PICS files.

1. Click on the following link to download the PICS XML files— <https://groups.csaiot.org/wg/members-all/document/folder/4120>
2. Click on the following link to use the PICS tool— [PICS Tool v1.6.4 matter 1.0 - Connectivity Standards Alliance \(csa-iot.org\)](#)
3. Load the Base.xml file by clicking on the **Browse** option. In case the following error is observed:



Base.xml: This XML PICS template is unapproved and has not been tested with this tool. To test new or updated PICS documents, please enable author mode and try again.

Enable author mode and retry uploading the XML file.

TO START: Drag-and-drop inside this box [or browse](#) to open one or more XML PICS files

Download [BDB](#), [Cluster](#) and [Matter](#) PICS using Causeway.

4. Load the XML file that is required for testing, e.g., Doorlock.xml.
5. Check the option for which the testing will be done for the DoorLock cluster. In the case of the Door Lock cluster to be tested in the Server mode, select the checkbox for DRLK.S. In case the cluster has to be tested in the Client mode, select the checkbox for DRLK.C.

PICS ITEMS – USAGE

DRLK.S	Does the device implement the DoorLock cluster as a server?	<input type="radio"/> <input checked="" type="checkbox"/> T/F
	§MATTER	
DRLK.C	Does the device implement the DoorLock cluster as a client?	<input type="radio"/> <input checked="" type="checkbox"/> T/F
	§MATTER	

6. Review all the attributes/commands that are supported by the DoorLock cluster and ensure the corresponding options are checked in the PICS tool.
7. Click on **Validate PICS**. Ensure that there are no warnings or errors. In case of any warnings or errors, revisit the options and check/uncheck the options as supported by the DUT.

Door lock Test Plan.xml

Click the filename to change.

154 Items Loaded
PICS File: Door lock Test Plan.xml
Template MD5: be38372d16f91c17957d5d1731af4fd
Template Status: Draft – Author Mode Enabled
Name: Door lock Test Plan

✓ Validate PICS Save XML Reset PICS Close PICS Edit PICS

Filter PICS
Item: X

Validation: Unvalidated Pass Warning Error

- Prior to the test execution, the user will have to load the relevant PICS file to list the required test cases. Depending on the PICS file loaded, the test suites list will be updated accordingly.

Version: TH Fall2023
Sha: 1c80214a

New Test Run

Test Details

Test Name: UI_Test_Run Description: (optional)

Operator: Alpha

Test Cases

python_tests SDK YAML Tests

Search test cases

Summary (01)

Test Suites

- FirstManualSuite (02)
 - TC-DLOG-3.1
 - TC-DRLK-2.10
 - TC-DRLK-3.2
 - TC-G-2.2
 - TC-G-2.3
 - TC-G-3.2
 - TC-GRPKEY-5.4
 - TC-I-3.2
 - TC-IDCM-2.2
- FirstChipToolSuite (08)
 - TC-DRLK-1.1
 - TC-DRLK-2.1 (Semi-automated)
 - TC-DRLK-2.4
 - TC-DRLK-2.5
 - TC-DRLK-2.6
- FirstAppSuite

Test Cases

- TC-DLOG-3.1
- TC-DRLK-2.10
- TC-DRLK-3.2
- TC-G-2.2
- TC-G-2.3
- TC-G-3.2
- TC-GRPKEY-5.4
- TC-I-3.2
- TC-IDCM-2.2

SDK YAML Tests

- FirstManualSuite
 - TC-DRLK-2.10 (01)
 - TC-DRLK-3.2 (01)
- FirstChipToolSuite
 - TC-DRLK-1.1 (01)
 - TC-DRLK-2.1 (Semi-automated) (01)
 - TC-DRLK-2.4 (01)
 - TC-DRLK-2.5 (01)
 - TC-DRLK-2.6 (01)

Start Clear Selection

9.1. Automated and Semi Automated Tests

9.1.1. Automated Test Cases

Click on the **SDK YAML Tests** tab. The automated and semi automated test cases will be listed in **FirstChipToolSuite**. The Automated test cases will be listed as the TC-<Cluster>-XX without any suffix, e.g., TC-DRLK-1.1. Automated test case execution will not require any manual intervention.

9.1.2. Semi Automated Test Cases

The Semi Automated test cases will be listed as TC-<Cluster>-XX(Semi-automated). During the Semi Automated test case execution, some of the steps will be executed automatically and the user will be prompted to perform a few steps as shown below in the screenshots. From the TH user interface, load the required PICS file to select the test cases, e.g., Doorlock Test Plan.xml.

Select the required Semi Automated test case to be executed and ensure other test cases are not selected. Take for example TC-ACE-1.6 as shown below:

New Test Run

Test Details

Test Name: UI_Test_Run Description: (optional)

Operator: Alpha

Test Cases

python_tests **SDK YAML Tests**

Search test cases:

Summary (01)

SDK YAML Tests

- FirstChipToolSuite
- TC-ACE-1.6 (Semi-automated) (01)
- TC-ACE-1.1
- TC-ACE-1.5 (Semi-automated)
- TC-ACFREMON-1.1
- TC-ACFREMON-2.1
- TC-ACL-1.1
- TC-ACL-2.1
- TC-ACL-2.2
- TC-ACL-2.3
- TC-ACL-2.4 (Semi-automated)
- TC-ACL-2.5 (Semi-automated)

Start Clear Selection

Bring up the DUT (All Clusters as Server) by sending the following command `./chip-all-clusters-app` on the Raspberry Pi terminal and click on the **Start** button.

During the Test execution, as the log gets updated, copy the newly generated node ID.

Test Execution

Logs

Test Suite Executing: FirstChipToolSuite
YAML Version: 19771ed7101321d68b87d05201d42d00adf5368f
Setting up chip_tool
New Node Id generated: 0xb1d2ee23dcf2f18b

chip-tool started: th-chip-tool with configuration: {'privileged': True, 'detach': True, 'network': 'host', 'name': 'th-chip-tool', 'command': 'tail -f /dev/null', 'volumes': {'/var/run/dbus/system_bus_socket': {'bind': '/var/run/dbus/system_bus_socket', 'mode': 'rw'}, PosixPath('/var/tmp'): {'bind': '/logs', 'mode': 'rw'}, PosixPath('/var/paa-root-certs'): {'bind': '/paa-root-certs', 'mode': 'ro'}}}

Step 10: TH sends a ViewGroup Command to the Groups cluster on Endpoint PIXIT.G.ENDPOINT over CASE with the GroupID 0x01050100
./chip-tool groups view-group 0x0105 1 0

Verify DUT sends a ViewGroupResponse with Status is set to NOT_FOUND on TH(chip-tool) Logs:
[1685009398.600925][39795:39797] CHIP:DMG: Received Command Response Data. Endpoint=0 Cluster=0x0000 0004 Command=0x01050100

PASS FAIL NOT APPLICABLE

Submit

Form the Chip-tool, execute the above command with node ID listed in the TH log. Save the Chip-tool logs in a text file. Verify the result in the Chip-tool log and select the applicable choice from the user prompt in the TH tool and select the **Submit** button.

Example:

```
docker exec -it th-sdk <popup command> <newly generated nodeID> <end-point id>
```

```
cd apps
```

```
docker exec -it th-sdk ./chip-tool groups view-group 0x0105 0xb1d2ee23dcf2f18b 0
```

Check for the response of the command in the Chip-tool log and compare with the expected response from the TH user prompt as shown below. In case both the responses match, click on **PASS** followed by the **Submit** button.

The screenshot shows the Test Project interface with the following details:

- Test Execution** tab is selected.
- Logs** panel displays the output of the Chip-tool log, including:
 - Test Suite Executing: FirstChipToolSuite
 - YAML Version: 19771ed7101321d68b87d05201d42d00adf5368f
 - Setting up chip_tool
 - New Node Id generated: 0xb1d2ee23dcf2f18b
 - chip-tool started: th-chip-tool with configuration: {privileged: True, 'detach': True, 'network': 'host', 'name': 'th-chip-tool', 'command': 'tail -f /dev/null', 'volumes': {'/var/run/dbus/system_bus_socket': {'bind': '/var/run/dbus/system_bus_socket', 'mode': 'rw'}, PosixPath('/var/paa-root-certs'): {'bind': '/paa-root-certs', 'mode': 'ro'}}}
 - [1685009398.600932][39795:39797] CHIP:TOO: Endpoint: 0 Cluster: 0x0000_0004 Command 0x0000_0001
 - [1685009398.600944][39795:39797] CHIP:TOO: ViewGroupResponse: {
 - [1685009398.600949][39795:39797] CHIP:TOO: status: 139
 - [1685009398.600955][39795:39797] CHIP:TOO: groupId: 261
 - [1685009398.600957][39795:39797] CHIP:TOO: groupName: [REDACTED]
 - [1685009398.600964][39795:39797] CHIP:DMG: ICR moving to [AwaitingDele...
 - [1698326717.243845][7:7] CHIP:DL: renamed tmp file to file (/tmp/chip_factory.ini)
 - [1698326717.244447][7:7] CHIP:DL: NVS set: chip-factory/unique-id = "91B51D08F4DB908D"
 - [1698326717.244556][7:7] CHIP:DL: writing settings to file (/tmp/chip_factory.ini-YsVtG)
- Test Cases** panel lists the steps of the test suite, all marked as **PASS**.
- Submit** button is visible at the bottom right of the logs panel.

At the end of the test execution, the user will be prompted to upload the Chip-tool logs that were saved in the previous step.

9.2. Python Tests

The Onboarding Payload Device Discovery test cases are listed under this option. Before executing the Python tests, bring up the DUT in the Chip-tool and save the discovery log. During the Python test execution, the user is prompted to input data such as QR code. Copy the data from the previously saved logs and provide the input. Follow the sequence below to execute the python_tests.

During the DUT bring-up, note down the QR code and save it for future use.

```
[1677044404.060044] [2528:2528] CHIP:DIS: Failed to advertise records: ../../examples/all-clusters-app/linux/third_party/connectedhomeip/src/inet/UDPEndPointImplSockets.cpp:411: OS Error 0x82000065: Network is unreachable
[1677044404.113596] [2528:2528] CHIP:DIS: mDNS service published: _matterc._udp
[1677044404.113714] [2528:2528] CHIP:IN: CASE Server enabling CASE session setups
[1677044404.113874] [2528:2528] CHIP:IN: SecureSession[0xaaaaaaaaaa3a0] - Allocated Type:2 LSID:44375
[1677044404.113960] [2528:2528] CHIP:SC: Allocated SecureSession (0xaaaaaaaaaa3a0) - waiting for Sigma1 msg
[1677044404.114024] [2528:2528] CHIP:SVR: Joining Multicast groups
[1677044404.114089] [2528:2528] CHIP:EVL: Emitting Startup event
[1677044404.114131] [2528:2528] CHIP:EVL: LogEvent event number: 0x0000000000000002 priority: 2, endpoint id: 0x0 cluster id: 0x0000_0028 event id: 0x0 Sys timestamp: 0x00000000012999A
[1677044404.114400] [2528:2528] CHIP:SVR: Server initialization complete
[1677044404.114409] [2528:2528] CHIP:SVR: Server initialized
[1677044404.114530] [2528:2528] CHIP:DLC: Device Configuration:
[1677044404.114630] [2528:2528] CHIP:DLC: Serial Number: TEST_SN
[1677044404.114758] [2528:2528] CHIP:DLC: Vendor Id: 65531 (0xFFFF)
[1677044404.114833] [2528:2528] CHIP:DLC: Product Id: 32769 (0x8001)
[1677044404.114887] [2528:2528] CHIP:DLC: Product Name: TEST_PRODUCT
[1677044404.114955] [2528:2528] CHIP:DLC: Hardware Version: 0
[1677044404.115080] [2528:2528] CHIP:DLC: Setup Pin Code (0 for UNKNOWN/ERROR): 20202021
[1677044404.115085] [2528:2528] CHIP:DLC: Setup Discriminator (0xFFFF for UNKNOWN/ERROR): 3840 (0xF00)
[1677044404.115121] [2528:2528] CHIP:DLC: Manufacturing Date: (not set)
[1677044404.115172] [2528:2528] CHIP:DLC: Device Type: 65535 (0xFFFF)
[1677044404.115249] [2528:2528] CHIP:SVR: SetupQRCode: [MT-24J042C00KA0648G0]
[1677044404.115310] [2528:2528] CHIP:SVR: Copy/paste the below URL in a browser to see the QR Code:
[1677044404.115358] [2528:2528] CHIP:SVR: https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A-24J042C00KA0648G0
[1677044404.115425] [2528:2528] CHIP:SVR: Manual pairing code: [3UQ27911232]
[1677044404.115490] [2528:2528] CHIP:DMS: Endpoint 0, Cluster 0x0000_0010 update version to e049b7aa
[1677044404.129314] [2528:2529] CHIP:DLC: TRACE: Bus acquired for name MATTER-3840
[1677044404.129872] [2528:2529] CHIP:DLC: CREATE: service object at /chipobole/09e0/service
[1677044404.130000] [2528:2529] CHIP:DLC: Create characteristic object at /chipobole/09e0/service/c1
[1677044404.130021] [2528:2529] CHIP:DLC: Create characteristic object at /chipobole/09e0/service/c2
[1677044404.130094] [2528:2529] CHIP:DLC: Create characteristic object at /chipobole/09e0/service
[1677044404.133909] [2528:2529] CHIP:DLC: CHIP_BT_P2_C /chipobole/09e0/service
[1677044404.133959] [2528:2529] CHIP:DLC: CHIP_ENABLE_ADDITIONAL_DATA_ADVERTISING is FALSE
[1677044404.109846] [2528:2528] CHIP:SVR: PlatformBlueZInit init success
[1677044404.150100] [2528:2528] CHIP:SVR: Cannot load binding table: ../../examples/all-clusters-app/linux/third_party/connectedhomeip/src/platform/Linux/KeyValueStoreManagerImpl.cpp:53: CHIP_Error 0x000000A0: Value not found in the persisted storage
[1677044404.150256] [2528:2528] CHIP:DLC: Updating services using commissioning mode 1
```

Select the python_tests tab for the test execution.

The screenshot shows the 'Test Project' interface with the 'New Test Run' screen. On the left, there's a sidebar with icons for Home, Test, Utility, and Settings. The main area has tabs for 'Test Details' and 'Test Cases'. Under 'Test Cases', the 'python_tests' tab is highlighted with a red box. The 'Summary (01)' panel on the right shows a hierarchical tree of test cases:

- python_tests
 - Onboarding Payload Test Suite
 - TC-DD-1.1
 - TC-DD-1.2
 - TC-DD-1.3
 - TC-DD-1.4

At the bottom, there are 'Start' and 'Clear Selection' buttons.

During the test execution the user is prompted for the QR code. Use the code that was saved earlier and proceed with the testing.

Test Project

Test Execution

Logs

Run Test Runner is Ready
TH Version: TH Fall2023
TH SHA: 1c80214a
TH SDK SHA: 19771ed
Test Run Executing
Test Suite Executing: Onboarding Payload Test Suite
Payload test suite setup

Please enter the QR code payload

MT:YNJV75HZ00KA0648G00

Submit

Abort

Test Cases

- python_tests [01]
- OnboardingPayloadTestSuite [04]
- TC-DD-1.1
 - Step1: Scan the DUT's QR code using a QR code reader
 - Step2.a: Verify the QR code payload version
 - Step2.b: Verify 8-bit Rendezvous Capabilities bit mask
 - Step2.c: Verify the 12-bit discriminator bit mask
 - Step2.d: Verify the onboarding payload contains a 27-bit Passcode
 - Step2.e: Verify passcode is valid
 - Step2.f: Verify QR code prefix
 - Step2.g: Verify Vendor ID and Product ID
 - Step4: Verify Custom payload support
- TC-DD-1.2
 - Step1: Verify the first digit of the pairing code
 - Step2: If the pairing code is 11 digits/the VID_PID flag is not set, verify...
 - Step2.b: If the pairing code is 21 digits the VID_PID flag is set, verify th...
 - Step3: Verify the 'check digit' of the pairing code (digit 11 or 21)
- TC-DD-1.3
 - Step1: Power up the DUT and put the DUT in pairing mode and bring th...
 - Step3.a: Verify the NFC code payload version
 - Step3.b: Verify 8-bit Rendezvous Capabilities bit mask
 - Step3.c: Verify the 12-bit discriminator bit mask

© Connectivity Standards Alliance

9.3. Manual Tests

During the manual test case execution, the user is prompted for an action for each test step as shown below.

Test Project

Test Execution

Logs

Run Test Runner is Ready
TH Version: TH Fall2023

Step 1b: TH reads the DoorLockAlarm event from DUT
./chip-tool doorlock read-event door-lock-alarm 1 1
Via the TH (chip-tool), verify TH receives the DoorLockAlarm event.
-In that event the AlarmCode is set to LockJammed(0) and priority is set to Critical.

PASS
 FAIL
 NOT APPLICABLE

Submit

Abort

Test Cases

- SDK YAML Tests [01]
- FirstManualSuite [01]
- TC-DRLK-2.10
 - Precondition
 - Step 1a: Trigger the DUT to generate D...
 - Step 1b: TH reads the DoorLockAlar...
 - Step 2a: Trigger the DUT to generate D...
 - Step 2b: TH reads the DoorStateChang...
 - Step 3a: TH sends the Lock Door com...
 - Step 3b: TH reads the LockOperation e...
 - Step 3c: TH sends the Unlock Door co...
 - Step 3d: TH reads the LockOperation e...
 - Step 3e: TH sends the Lock Door com...
 - Step 3f: TH reads the LockOperationE...
 - Step 4c: TH sends Unlock Door Comm...
 - Step 4d: TH reads the LockOperationError event from DUT
 - Step 5a: TH sends Set User Command to DUT with the following value...
 - Step 5b: TH reads the LockUserChange event from DUT
 - Step 5c: TH send Set Week Day Schedule Command to DUT with the f...
 - Step 5d: TH reads the LockUserChange event from DUT
 - Step 5e: TH sends Set Credential Command to DUT with the following ...
 - Step 5f: TH reads the LockClearChange event from DUT

Event (by LockJammed scenario)
LockJammed scenario).

DoorLockAlarm Event (by LockJammed scenario)

© Connectivity Standards Alliance

After the Manual pairing of the DUT, execute the command displayed on the prompt as shown below.

```
Example: ./apps/chip-tool doorlock read-event door_lock-alarm 1 1
```

Save the Chip-tool logs in a text file. Validate the chip tool log and select the applicable choice from the user prompt in the TH tool and select the **Submit** button. At the end of the test execution, the user is prompted to upload the Chip-tool logs that were saved in the previous step.

9.4. Simulated Tests

Simulated tests must be executed when the DUT is considered as a Client. The simulated test cases will be listed in **FirstAppSuite** under the **SDK YAML Tests** tab.

The screenshot shows the 'Test Project' interface with the 'New Test Run' screen. On the left, there's a sidebar with icons for Home, Test, Utility, and Settings. The main area has tabs for 'Test Details' (Test Name: UI_Test_Run, Description: (optional)), 'Test Cases' (with sub-tabs 'python_tests' and 'SDK YAML Tests' - the latter is highlighted with a red box), and 'Summary (01)'. The 'Test Suites' section lists 'FirstManualSuite', 'FirstChipToolSuite', and 'FirstAppSuite' (which is also highlighted with a red box). The 'Test Cases' section lists various test cases like TC-CC-3.4, TC-CC-4.5, etc., with TC-CC-3.4 checked. The 'Summary' panel on the right shows the selected test cases: 'SDK YAML Tests' > 'FirstAppSuite' > 'TC-CC-3.4'.

During the execution of these tests, the user is prompted for an action to be performed on the device as shown in the following screenshot.

Follow the instructions provided in the user prompt to complete the test execution.

The screenshot shows the Test Project interface with a central 'Test Execution' panel. On the left, there's a sidebar with icons for Home, Test, Utility, and Settings. The main area has tabs for 'Test Cases' and 'Logs'. A modal dialog is open, prompting the user to take an action on the Controller: 'DUT sends MoveToHue command to TH an Hue with _TransitionTime 300'. It lists three options: PASS, FAIL, or NOT APPLICABLE. Below the dialog, the logs show a series of commands and responses related to the certification process.

IMPORTANT: Currently the selection will be done automatically by TH based on the test execution result. In the future the User Prompt will be updated to properly represent this behavior.

9.5. SDK Python Tests

9.5.1. Run Tests Inside SDK Docker Container

Some automated Python scripts are available inside the docker of the TH.

E.g.: TC_ACE_1_3.py, TC_ACE_1_4.py , TC_CGEN_2_4.py , TC_DA_1_7.py , TC_RR_1_1.py TC_SC_3_6.py

Follow the instructions below to execute the test cases.

9.5.1.1. Prerequisite

1. A directory containing the PAA (Product) roots that will be mounted as /paa_roots.
2. Run the following commands from the Raspberry Pi terminal.

```
cd certification-tool
./backend/test_collections/matter/scripts/update-paa-certs.sh
```

3. After execution of the above commands ensure that the PAA's are available locally at /var/paa-root-certs .

9.5.1.2. Mapped Volumes

The following host directories are mapped into the cert-bins container:

- `/root/python_testing` → `/home/ubuntu/certification-tool/backend/test_collections/matter/sdk_tests/sdk_checkout/python_testing`
- `/paa-root-certs` → `/var/paa-root-certs`
- `/credentials/development` → `/var/credentials/development`

9.5.1.3. Placeholders for Steps

Device-specific configuration is shown as shell variables. **PLEASE REPLACE THOSE WITH THE CORRECT VALUE** in the steps below.

- **\$PATH_TO_PAA_ROOTS**: Path on host where PAA roots are located. Failure to provide a correct path will cause early failure during commissioning (e.g., `/var/paa-root-certs/`)
- **\$DISCRIMINATOR**: Long discriminator for DUT (e.g., 3840 for Linux examples)
- **\$SETUP_PASSCODE**: Setup passcode for DUT (e.g., 20202021 for Linux examples)
- **\$WIFI_SSID**: SSID of Wi-Fi AP to which to attempt connection



Currently, `WIFI_SSID` with special characters or empty spaces is not supported.

- **\$WIFI_PASSPHRASE**: Passphrase of Wi-Fi AP to which to attempt connection
- **\$BLE_INTERFACE_ID**: Interface ID for BLE interface (e.g., 0 for default, which usually works)
- **\$THREAD_DATASET_HEX**: Thread operational dataset as a hex string (e.g., output of dataset active -x in OpenThread CLI on an existing end-device)

9.5.1.4. Common Steps

Factory-reset the DUT

```
docker run -v $PATH_TO_PAA_ROOTS:/paa_roots -v
/var/run/dbus/system_bus_socket:/var/run/dbus/system_bus_socket -v
mapped_data_model:/root/python_testing/data_model -v /home/ubuntu/certification-
tool/backend/test_collections/matter/sdk_tests/sdk_checkout/python_testing:/root/python_testin
g -v $(pwd):/launch_dir --privileged --network host -it connectedhomeip/chip-cert-bins:<SDK
SHA RECOMMENDED>
```

This downloads a Docker image with the test environment, and runs the environment including mounting the PAA trust store in `/paa_roots` and mounts the local Avahi socket so that Avahi in the VM can run against its host.

- You will be shown a # root prompt



The first time running docker will be SLOW (around 5 minutes) due to the need to download data. Every other run after that will be instant.

9.5.1.5. For On-Network Pairing

Execute the following command:

```
rm -f admin_storage.json && python3 python_testing/scripts/sdk/TC_RR_1_1.py --discriminator $DISCRIMINATOR --passcode $SETUP_PASSCODE --commissioning-method on-network --paa-trust-store -path /paa_roots --storage-path admin_storage.json
```

To test this against a Linux target running on the same network as the host:

```
clear && rm -f kvs1 && ./chip-all-clusters-app --discriminator 3842 --KVS kvs1 --trace_decode 1
```



- The \$DISCRIMINATOR to be used will be 3842 in this example.
- The **rm -f kvs1** is a factory reset.

9.5.1.6. For BLE+Wi-Fi Pairing

Execute the following command in the docker for the BLE+Wi-Fi pairing:

```
rm -f admin_storage.json && python3 python_testing/scripts/sdk/TC_RR_1_1.py --discriminator $DISCRIMINATOR --passcode $SETUP_PASSCODE --commissioning-method ble-wifi --paa-trust-store -path /paa_roots --storage-path admin_storage.json --wifi-ssid $WIFI_SSID --wifi-passphrase $WIFI_PASSPHRASE --ble-interface-id $BLE_INTERFACE_ID
```

9.5.1.7. For BLE+Thread Pairing

Execute the below command in the docker for the BLE+Thread pairing:

```
rm -f admin_storage.json && python3 python_testing/scripts/sdk/TC_RR_1_1.py --discriminator $DISCRIMINATOR --passcode $SETUP_PASSCODE --commissioning-method ble-thread --paa-trust-store -path /paa_roots --storage-path admin_storage.json --thread-dataset-hex $THREAD_DATASET_HEX --ble-interface-id $BLE_INTERFACE_ID
```

9.5.1.8. Post-Test Steps

Factory reset the DUT again → The test fills tons of stuff and the device will be in an odd state of ACL's. This will be fixed once there is ample time to clean up after the test is completed by sending commands to, for example, remove the fabrics joined.

9.5.1.9. Possible Issues

- Failing at Step 9 during execution of TC_RR_1_1:
 - Some DUT's have an incorrectly-configured UserLabel cluster where the backend is not implemented due to SDK example issues where some examples have the backend and others do not. This will fail at the last step ("Step 9: Fill UserLabel clusters on each endpoint"), with FAILURE writes. To override the test not to run this step, you can add "**--bool-arg skip_user_label_cluster_steps:true**" to the command line of **TC_RR_1_1.py**, at the end.
 - Not having the **\$PATH_TO_PAA_ROOTS** set properly when starting the docker or not having PAA roots certificates at that path.
 - Follow the instructions for item 2 in [Section 9.5.1.1, Prerequisite](#).

Common Test Failures

The documents in this [link](#) are intended to be used to help root-cause common test failures, especially in cases where the underlying cause of the failure may not be immediately obvious from the test step or expected outcomes.

9.5.2. Run Tests on the TH User Interface

Some automated Python scripts are available in TH User Interface.

To execute the tests, the parameters **discriminator**, **setup_code** and **pairing_mode** need to be filled in the device configuration parameters (**dut_config**).

To configure specific/custom parameters, please edit the project configuration to include the parameters in the session (**test_parameters**).

Project configuration example:

```
{  
    ...  
    "dut_config": {  
        "discriminator": "3840",  
        "setup_code": "20202021",  
        "pairing_mode": "onnetwork",  
        "chip_tool_timeout": null,  
        "chip_tool_use_paa_certs": false  
    },  
    "test_parameters": {  
        "paa-trust-store-path": "/credentials/development/paa_roots",  
        "storage-path": "admin_storage.json"  
    }  
    ...  
}
```

9.5.2.1. Test Parameters for SDK Python Tests

PIXIT Support

PIXIT type parameters must be filled in the **test_parameters** section. The following example will be used to define the following parameters:

```
PIXIT.ACE.APPENDPOINT:1  
PIXIT.ACE.APPDEVTYPEID:256  
PIXIT.ACE.APPCLUSTER:OnOff  
PIXIT.ACE.APPATTRIBUTE:OnOff
```

Project configuration example:

```
{
  ...
  "test_parameters": {
    "paa-trust-store-path": "/credentials/development/paa_roots",
    "storage-path": "admin_storage.json",
    "int-arg": "PIXIT.ACE.APPENDPOINT:1 PIXIT.ACE.APPDEVTYPEID:256",
    "string-arg": "PIXIT.ACE.APPCLUSTER:OnOff PIXIT.ACE.APPATTRIBUTE:OnOff"
  }
  ...
}
```

The above example will be used to define the following arguments when running the test:

```
--int-arg PIXIT.ACE.APPENDPOINT:1 PIXIT.ACE.APPDEVTYPEID:256 --string-arg
PIXIT.ACE.APPCLUSTER:OnOff PIXIT.ACE.APPATTRIBUTE:OnOff
```

Test Parameters Examples

Access the spreadsheet via the [Verification Steps Document](#) and review the information provided. Based on this data, create the parameters set as requested.

Below are some specific examples assembled from data obtained from the spreadsheet.

1. TC-ACE-1.4

Sample command to run manually inside docker

```
python3 TC_ACE_1_4.py --discriminator 3840 --passcode 20202021 --commissioning
-method on-network --storage-path admin_storage.json --int-arg
PIXIT.ACE.APPENDPOINT:1 PIXIT.ACE.APPDEVTYPEID:256 --string-arg
PIXIT.ACE.APPCLUSTER:OnOff PIXIT.ACE.APPATTRIBUTE:OnOff --paa-trust-store-path
/credentials/development/paa-root-certs/
```

Arguments to be used while executing using UI (use product specific values)

```
"test_parameters": {
  "int-arg" : "PIXIT.ACE.APPENDPOINT:1 PIXIT.ACE.APPDEVTYPEID:256",
  "string-arg": "PIXIT.ACE.APPCLUSTER:OnOff PIXIT.ACE.APPATTRIBUTE:OnOff"
}
```

2. TC-SC-7.1

Sample command to run manually inside docker

```
python3 TC_SC_7_1.py --bool-arg post_cert_test:true --qr-code MT:-
<24J0CEK01KA0648G00> --storage-path admin_storage.json --paa-trust-store-path
```

Arguments to be used while executing using UI (use product specific values)

```
"test_parameters": {  
    "bool-arg": "post_cert_test:true",  
    "qr-code": " MT-24J042C00KA0648G00"  
}
```

9.5.2.2. Test suites

TH expects the SDK Python Tests to follow a certain template. New tests are being written with this template and the old tests are being updated to conform to it. The tests are divided in 3 test suites:

New Project Version: Sha:
New Test Run
Test Details
Test Name: UI_Test_Run Description: (optional) Operator
Test Cases
Onboarding Payload Tests SDK YAML Tests Mandatory SDK Python Tests **SDK Python Tests** Search test cases Summary
Test Suites
Python Testing Suite Python Testing Suite - Old script format
Test Cases
TC-ACE-1.3 TC-ACE-2.1
TC-ACE-2.2 TC-ACL-2.2
TC-ACL-2.11 TC-BOOLCFG-2.1
TC-BOOLCFG-3.1 TC-BOOLCFG-4.1
TC-BOOLCFG-4.2 TC-BOOLCFG-4.3
TC-BOOLCFG-4.4 TC-BOOLCFG-5.1
Please Select test cases
Start Clear Selection

1. Python Testing Suite

- For test cases that follow the expected template and have a commissioning first step.
- The user will be asked to make sure that the DUT is in Commissioning Mode at the start of the test suite setup and then the DUT will be commissioned.
- The commissioning will be kept throughout the execution of all its tests.

2. Python Testing Suite - No commissioning

- For test cases that follow the expected template but don't have a commissioning first step.
- The selected tests will be executed without commissioning the DUT.
- The user will be asked to make sure that the DUT is in Commissioning Mode at the start of each test.

3. Python Testing Suite - Old script format

- For test cases that don't follow the expected template yet.
- The user will be asked to make sure that the DUT is in Commissioning Mode at the start of each test.
- The user will also be asked if the DUT should be commissioned at the start of each test. The DUT will be commissioned depending on the user's answer.

The screenshot shows the 'Test Execution' interface of a software tool. On the left sidebar, there are icons for Home, Test, Utility, and Settings. The main area displays a list of test cases under 'Test Cases': 'SDK Python Tests' (01), 'Python Testing Suite - Legacy' (01), and 'TC-ACE-1.2'. Below these are buttons for 'Start Python test' and 'Show test logs'. A modal dialog box is centered over the list, titled 'Make sure the DUT is in Commissioning Mode'. It contains two radio buttons: 'DONE' (selected) and 'FAILED'. At the bottom right of the modal is a blue 'Submit' button. In the background, the log pane shows command-line output related to commissioning and test execution. The top right corner of the interface shows 'Version: main - dev Sha: 089e1d1' along with a bell icon and a sun icon.

This screenshot shows the same 'Test Execution' interface as the previous one. The modal dialog has changed to 'Should the DUT be commissioned to run this test case?'. It contains two radio buttons: 'YES' (selected) and 'NO'. The background log pane shows the same commissioning and test execution logs as the first screenshot. The top right corner shows 'Version: main - dev Sha: 089e1d1' and the status bar indicates 'User chose prompt option DONE'.

9.5.2.3. Reuse commissioning information

This allows users to perform multiple test run executions without the need to perform the commissioning step in every test run execution. The TH is now storing the last commissioning information, so a prompt will presented asking user to reuse those previous commissioning information or if he wants to perform a new commissioning procedure.

New Project ▾

Version: spring2025
Sha: Unknown

Home Test Utility Settings

Test Execution

Test Cases

- SDK Python Tests [0]
- Python Testing Suite [0]

Logs

Run Test Runner is Ready
TH Version: spring2025
TH SHA: Unknown
TH SDK SHA: 017e8bb5

Test Run Executing

Start Python test
Commissioning, already done
TH0 writes ACL all view on PIXIT.ACE.TESTENDPOINT
TH1 reads EP0 descriptor - expect SUCCESS
TH2 reads EP0 descriptor - expect SUCCESS
TH3 reads EP0 descriptor - expect SUCCESS
TH0 writes ACL TH1 view on EP0
TH1 reads EP0 descriptor - expect SUCCESS
TH2 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH3 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH0 writes ACL TH2 view on EP0
TH1 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH2 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH3 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH0 writes ACL TH3 view on EP0
TH1 reads EP0 descriptor - expect UNSUPPORTED_ACCESS
TH2 reads EP0 descriptor - expect UNSUPPORTED_ACCESS

Do you want to reuse the previous commissioning information?
If you select NO, a new commissioning will be performed

YES
 NO

Submit

4d53e721b2

```
#####
#posixPath('/home/ubuntu/certification-tool/backend/test_collections/matter/sdk_tests/support/performance_tests/scripts/sdk/TC_COMMISSIONING_1_0.py'): {'bind': '/root/python_testing/scripts/sdk/TC_COMMISSIONING_1_0.py', 'mode': 'rw'}, PosixPath('/home/ubuntu/certification-tool/backend/test_collections/matter/sdk_tests/support/performance_tests/scripts/sdk/accessory_manager.py'): {'bind': '/root/python_testing/scripts/sdk/accessory_manager.py', 'mode': 'rw'}, PosixPath('/home/ubuntu/certification-tool/backend/test_collections/matter/sdk_tests/support/performance_tests/scripts/sdk/simulated_accessory.py'): {'bind': '/root/python_testing/scripts/sdk/simulated_accessory.py', 'mode': 'rw'}}}
User prompt: "Make sure the DUT is in Commissioning Mode"
User chose prompt option DONE
User prompt: "Do you want to reuse the previous commissioning information?"
```