AN1118: Certifying Zigbee® 3.0 Devices

This document provides guidelines for certifying Zigbee 3.0 devices. It describes how to set up the Silicon Labs Zigbee Test Harness and provides details on using the Zigbee Test Tool and Test Harness for internal pre-testing, along with troubleshooting tips.

KEY POINTS

- Overview of the Connectivity Standards Alliance Certification Process
- Zigbee test harness setup
- Using the Zigbee test tool and test harness
- Troubleshooting tips

Note: As of 2021 Zigbee Unified Test Harness (ZUTH) is the official test tool for the CSA. ZUTH is owned and supported by the CSA, not Silicon Labs. Contact the CSA with any questions.

Introduction

This document provides guidelines for certifying Zigbee 3.0 devices. It includes an overview of the process with tips for a successful outcome. It describes how to set up the Silicon Labs Zigbee Test Harness, including information about the Zigbee Test Tool firmware. It then provides details on using the Zigbee Test Tool and Test Harness for internal pre-testing, including a full example. Tips on trouble-shooting testing problems are covered. Finally, a list of resources is provided. Note that the official documents from Connectivity Standards Alliance may require membership for full access.

1.1 Definitions

The following abbreviations and acronyms are used throughout this document.

BDB: Base Device Behavior.

CSA: Connectivity Standard Alliance.

DoC: Declaration of Conformity.

DUT: Device under test.

PICS: Protocol Implementation Conformance Statement. The PICS will present a matching between DUT's behavior and its items.

PIXIT: Protocol Implementation Extra Information for Testing. PIXIT generally indicates manufacturer-specific information. In ZTT software, PIXIT also contains items to ensure the software's functioning.

TH or ZTH: Zigbee Test Harness. THrx denotes the router role of a test harness. THex denotes the end device role. THcx denotes the coordinator role.

ZTT: Zigbee Test Tool. If not specified, ZTT usually refers to the software that executes test scripts. ZTT firmware relates to the ZTT dongle.

ZC: Zigbee Coordinator.

ZED: Zigbee End Device.

ZR: Zigbee Router.

1.2 Documentation Conventions

Internal references to documents or websites are in the form of a cross-reference **[R#]**, where R# is a specific item in the reference list (see section 6 Reference Material).

Overview of the Connectivity Standards Alliance Certification Process

Silicon Labs provides EmberZNet, a Zigbee-Compliant Platform, but the Connectivity Standards Alliance also provides members a path for product certification as Zigbee-Certified Products. This section summarizes the Zigbee certification process provided by the Connectivity Standards Alliance. The Connectivity Standards Alliance may change these details at any time. If you have questions about Zigbee certification or the certification process, contact the Connectivity Standards Alliance directly at certification@zigbee.org.

2.1 Getting Started

The first step of the certification process is to submit your product to an authorized test service provider. A list of authorized providers can be found on the Connectivity Standards Alliance website **[R1]**.

Note: Although not required, many companies do internal testing before sending the product to an authorized test service provider. Procedures for in-house testing are documented in section 4 Using the Zigbee Test Tool and Test Harness for In-House Pre-Testing.

The second step in the process is to submit an application online using the Connectivity Standards Alliance Certification Web Tool (http://zigbeecertifiedproducts.knack.com/zigbee-certified). If you don't already have a Connectivity Standards Alliance user account (separate from your Workspace account), you can sign up as a new user by clicking the "Sign Up" link. When requesting a new account, be sure to sign up with a company e-mail address (not Gmail, Yahoo, 163.com, or other free e-mail domains).

For detailed instructions on creating and submitting an application in the Certification Tool, a user information guide is available on **[R1]**. After you complete and submit your application, the Connectivity Standards Alliance will review it and send you either an approval notice or feedback via e-mail.

2.2 Documentation Tips and Pointers

Have all documents ready:

- To ensure prompt processing of your application, remember to upload all required documents before submitting your application.
- All applications should include completed Declaration of Conformity (DoC) and Protocol Implementation Conformance Statement (PICS) documents. Protocol Implementation Extra Information for Testing (PIXIT) specifications are included in PICS documentations. For Zigbee 3.0 applications, multiple PICS documents are required (one for base device behavior plus additional documents for each cluster). These may be uploaded to the Certification Tool in a single ZIP archive.

Ensure consistency across all documents submitted:

A common error is inconsistent information across the web application, DoC, PICS, and test report. To save on processing time, double-check these entries:

- All hardware/software version fields must match the versions declared in the test report.
- The compliant platform declared in the application must be consistent with the DoC and test report (or the original certified application, for Certification by Similarity requests).

Tips for completing the DoC:

- Use the latest DoC template. It can be found on **[R1]**.
- Remember to have both the applicant and test house signature blocks signed and dated.

Tips for completing the PICS:

- Each Zigbee standard has its own PICS document, available with its respective specification document package, available on **[R2]**.
- Brief instructions for completing PICS documents are found at the beginning of each document template.
- Make sure that all mandatory PICS items are supported.

Setting up the Silicon Labs Zigbee Test Harness

This section describes how to set up and use the Silicon Labs Zigbee 3.0 test harness. Two options for the ZTT dongle hardware are:

- The Silicon Labs WSTK with BRD4162 radio board
- The Silcon Labs Thunderboard Sense 2

3.1 Requirements

In order to set up the Silicon Labs Zigbee 3.0 test harness (ZTT) you will need the following:

Hardware (ZTT Dongle):

The hardware that interfaces the ZTT to the Zigbee network as a Test Harness is called the ZTT Dongle in this document. The hardware can be purchased at the following:

- EFR32MG Zigbee and Thread Starter Kit (WSTK) (Part - SLWSTK6000B):
  https://www.silabs.com/development-tools/wireless/zigbee/efr32mg-zigbee-thread-starter-kit
- Thunderboard Sense 2 Sensor-to-Cloud Advanced IoT Kit (TBS2) (Part - SLTB004A):
  https://www.silabs.com/products/development-tools/thunderboard/thunderboard-sense-two-kit

Software:

- ZTT (Zigbee 3.0 Test Tool) software version 1.0.2.4 or higher for Windows can be downloaded from the Zigbee website at **[R3]**.
- A ZTT license can be purchased from Zigbee online at:
  https://www.regonline.com/Register/Checkin.aspx?EventID=1849737
- Silicon Labs Simplicity Studio can be downloaded from the Silicon Labs website here:
  http://www.silabs.com/products/development-tools/software/simplicity-studio
- The latest ZTT dongle firmware can be downloaded from the Zigbee website at **[R3]**.

ZTT Dongle Driver:

The driver configures the attached ZTT Dongle as a serial port (appears as COM port) interface that the ZTT software can use. The WSTK and TBS2driver is installed as part of the Simplicity Studio installation.

Programming Tools:

The above hardware does not come with the ZTT Dongle firmware (explained in the next section) pre-programmed. You must program the firmware on the hardware using following tools The WSTK and TBS2 boards have a built-in J-Link to program the EFR32 MG12 parts using Simplicity Commander (provided as part of Simplicity Studio).

3.2 About the ZTT Firmware

The ZTT firmware binary is built for the WSTK EFR32MG12 boards (BRD4162A) and TBS2 EFR32MG12 boards (BRD4166A. Binaries are provided for the following use cases:

- Cordinator/Router: For test cases where the test harness should operate as a router or coordinator
- End Device: For test cases where the test harness should operate as an end device

The naming format used for the firmware files are TRaC_Z3_<role>_<board> where:

- <role>: Role of the node, the firmware can be used by the ZTT for test case execution in following roles:
  - “ZCR”: can be used as coordinator or router by the ZTT
  - “ZED”: can be used only as end device by the ZTT
- <board>: board on which the firmware runs
  - 4161A: WSTK radio board
  - 4166A: TBS2 radio board

The firmware files are provided in following image formats:

- .S37: Standard Motorola S-Record format, used for a single application or bootloader image.
- GBL: Silicon Labs proprietary format for application images, designed to work with the Silicon Labs Gecko bootloader.

The gbl files are provided so that you can update the ZTT dongle firmware using a serial terminal tool that supports the xmodem protocol.

The following binaries are provided for the WSTK with EFR32MG12 and the test roles:

Binary	Platform	Test	Description
TRaC_Z3_ZCR_<board>.s37	MG12	BDB or Cluster	ZTT firmware binary file that can be programmed to the EFR32MG12 daughter board (BRD4162A) on a WSTK or Thunderboard Sense 2 (BRD4166A) using the Simplicity Commander programming tool. Refer to section 3.3.1 Programming the Firmware File.
TRaC_Z3_ZCR_<board>.gbl	MG12	BDB or Cluster	ZTT firmware GBL file that can be programmed with the uart-xmodem-MG12 bootloader using the procedure in section 3.3.2 Using the Xmodem Protocol.
TRaC_Z3_ZED_<board>.s37	MG12	BDB	ZTT firmware (ZED) binary file that can be programmed to the EFR32MG12 daughter board (BRD4162A) on a WSTK or Thunderboard Sense 2 (BRD4166A) using the Simplicity Commander programming tool. Refer to section 3.3.1 Programming the Firmware File.
TRaC_Z3_ZED_<board>.gbl	MG12	BDB	ZTT firmware GBL file that can be programmed with the uart-xmodem-MG12 bootloader using the procedure in section 3.3.2 Using the Xmodem Protocol.
bootloader-uart-xmodem-MG12- combined.s37	MG12	-	Bootloader binary that must be programmed to the EFR32MG12 daughter board (BDR4162A) on a WSTK or Thunderboard Sense 2 (BRD4166A) using the Simplicity Commander programming tool. Refer to section 3.3.1 Programming the Firmware File.

The following commands implemented in the binaries are not included in the current ZTT API Reference. For usage see the command's help.

Command Group	Command Name	Description
custom	lookup	Looks up a short address from the neighbor table given an IEEE address string
custom	resetBootloader	Resets the dongle and enters the bootloader. This allows loading the .ebl file through a terminal emulator with xmodem support.

3.3 Loading the ZTT Firmware

Because Simplicity Studio is an essential tool required for using the built-in J-Link on a WSTK board or TBS2 board, install Simplicity Studio 5 before following the procedures in this section.

3.3.1 Programming the Firmware File

The WSTK/TBS2 ZTT Dongle firmware files and bootloader files are downloaded using Simplicity Commander as follows.

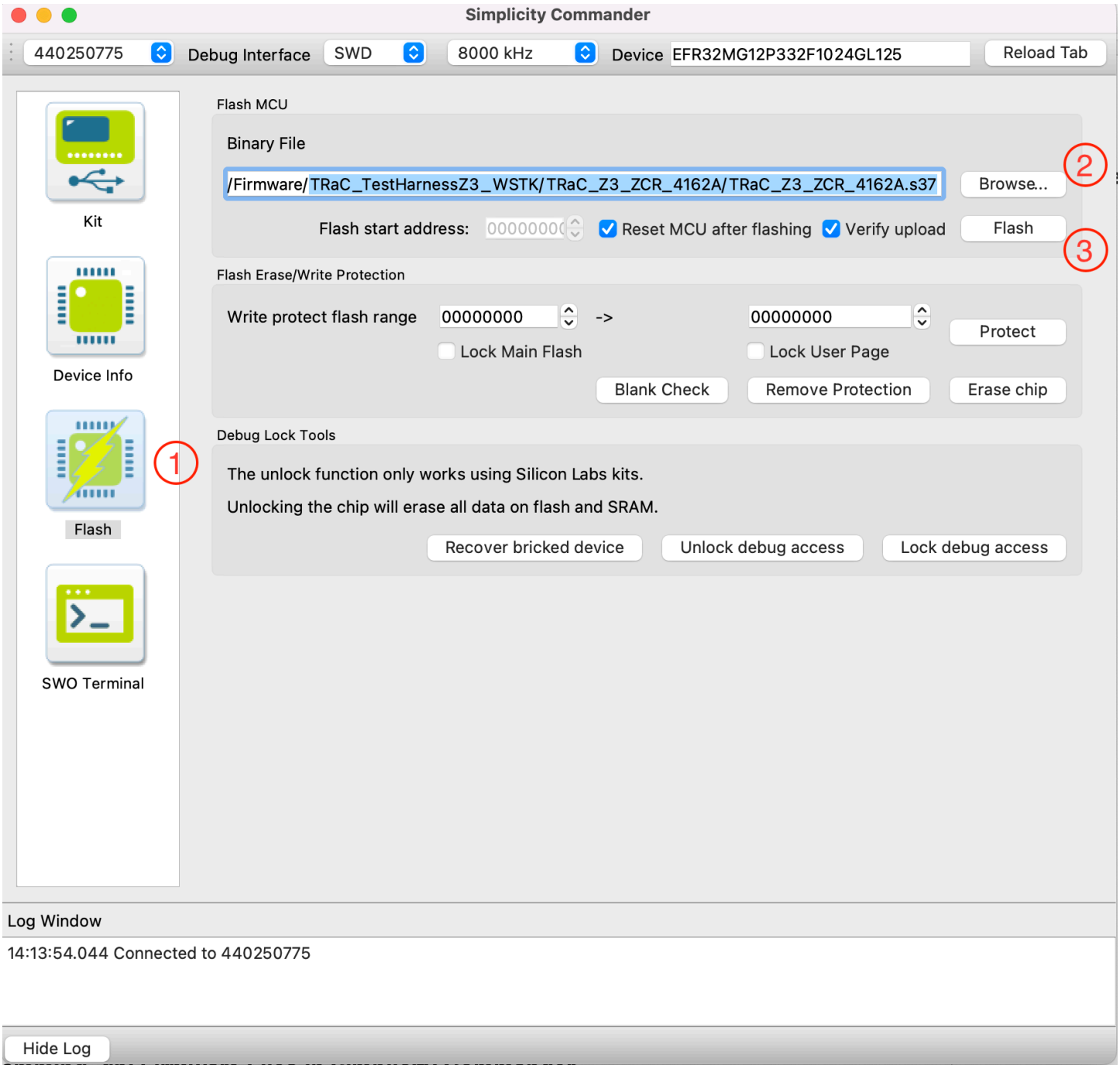


Figure 3-1. Reprogramming .gbl Firmware Files in Simplicity Commander

1. Connect the USB cable from the WSTK board (with BDR4162A daughter board) to the computer. Note the “J-Link SN:” serial number that appears on the onboard LCD.
2. Launch Simplicity Commander (commander.exe) from its Simplicity Studio installation directory. Generally, the path looks like:

```
C:\SiliconLabs\SimplicityStudio\v5\developer\adapter_packs\commander\commander.exe
```

3. In the Simplicity Commander graphical user interface (shown in the previous figure), in the Adapter bar select the J-Link Device serial number and click **Connect**. This changes to **Disconnect** when successfully connected.
4. In the target bar click **Connect**. This changes to **Reconnect** when successfully connected.
5. Click the large **Flash** control in the left panel (1 in the previous figure).
6. Click **Browse** next to the **Binary File** field (2) and locate the WSTK bootloader file, for example: `bootloader-uart-xmodem-MG12-combined.s37`.
7. Click **Flash** (3). The bootloader file downloads.
8. Click **Browse** again and locate the WSTK ZTT dongle firmware file, for example

```
TRaC_Z3_ZCR_4162A.s37
```

9. Click **Flash**. The firmware file downloads.

3.3.2 Using the Xmodem Protocol

If a ZTT dongle that has a bootloader already programmed and the running ZTT Firmware needs a firmware upgrade, it can be done using a serial tool that supports the xmodem protocol. This option is only useful when the user does not have access to the above programming method using the ISA3 or Simplicity Studio.

The steps are as follows:

1. Open a terminal emulator that supports x-modem protocol to send a file and set the following:
   - Com port = that of the connected test harness.
   - Baud rate= 115200
   - Data = 8 bit
   - Stop = 1 bit
   - Parity = none
   - Flow control = none

   If all the settings are correct, when you press <CR or ENTER >, a test harness prompt is returned, such as:

```
TRaC_Z3_ZCR_4162A
```

2. To enter the bootloader, enter:

```
TRaC_Z3_ZCR_4162A>custom resetBootloader <CR>
```

   If the command is successful, a menu such as the following is displayed:

```
Gecko Bootloader v1.1.0
1. upload gbl
2. run
3. ebl info
BL >
```

3. Enter 1. The following is displayed:

```
begin upload
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
```

4. Send the .gbl file using the xmodem protocol from the terminal.
5. Once the transfer is complete, enter 2 to run the firmware.

3.4 Setup Procedure

An overview of the procedure for setting up your own Zigbee test harness is as follows:

1. Once the steps in section 3.3 Loading the ZTT Firmware are complete, connect the ZTT dongle to the computer.
2. Test the presence of the ZTT firmware by connecting to the dongle using a terminal such as Tera Term with the ZTT Com port (Baud =115200, Data=8bit, Stop=1, Parity=None, FlowControl=None).

   When you press Enter, you should see a command prompt such as `TRaC_Z3_ZCR_4162A>`. Disconnect and close the terminal before the next step.

3. Launch the ZTT software.
4. Enter the license for the ZTT software.
5. Discover the ZTT dongle as your test harness.
6. Use the ZTT software to test your Zigbee 3.0 devices.

For more information about testing see document **[R11]**, available on the Zigbee website.

Using the Zigbee Test Tool and Test Harness for In-House Pre-Testing

In this section, an example of test setup will be presented, followed by explanations of PICS and PIXIT selection. The section ends with a complete execution of a cluster test case to serve as a quick start.

4.1 Test Setup

It is recommended to conduct In-house pre-testing in a shield room or in a shield box. The idea is to provide a clean RF environment and to avoid interfering with the DUT's performance. A shield box is illustrated in the following figure.

Figure 4-1. A Shield Box

Also, the official test specification requires deploying a sniffer to capture over-the-air packets for verification on certain test steps. The following figure shows a sniffing device connected to an ISA3 debugger. It is also feasible to use the built-in sniffing capability from the DUT with WSTK connection.

Figure 4-2. Sniffing Device Setup

Some BDB test cases require support by multiple test harnesses with various roles (for example THr1, THr2, THe1, and so forth). To save time on role-swapping, it is convenient to use an USB hub, and connect test harnesses with fixed roles to the hub if TH availability is not a concern. The following figure shows such a connection.

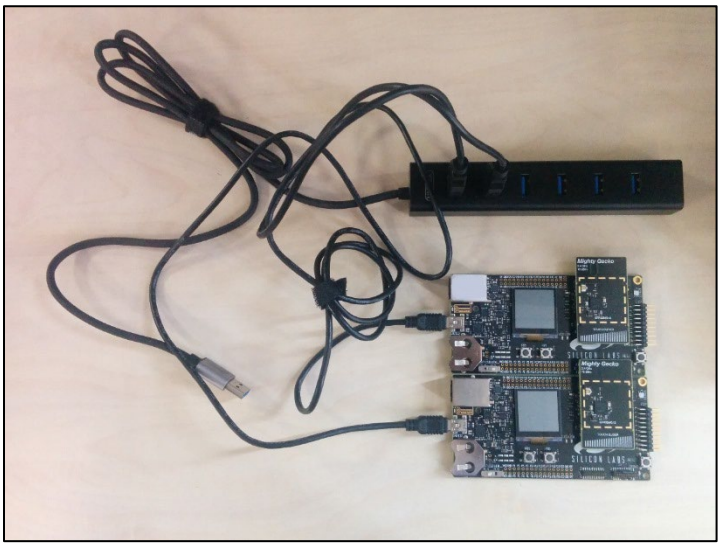


Figure 4-3. A USB Hub with Two Test Harnesses Connected

The lighting reference design (https://www.silabs.com/documents/public/user-guides/UG252.pdf) servers as a DUT, and is shown in the following figure.

Figure 4-4. EFR32 Lighting Reference Design Connected to a WSTK Board

The following figure shows the complete device setup for testing a lighting reference design and built-in sniffing capability. Along with test harnesses, the DUT is enclosed in the shield box.

Figure 4-5. The Complete Device Setup for Lighting Reference Design

Several consoles/graphical interfaces could be opened for DUT's behavior investigation, over-the-air packets verification and standalone debugging. They are the UART console for test harnesses, virtual UART console for DUT (Figure 4-6), and an analyzer interface for sniffing (Figure 4-7).

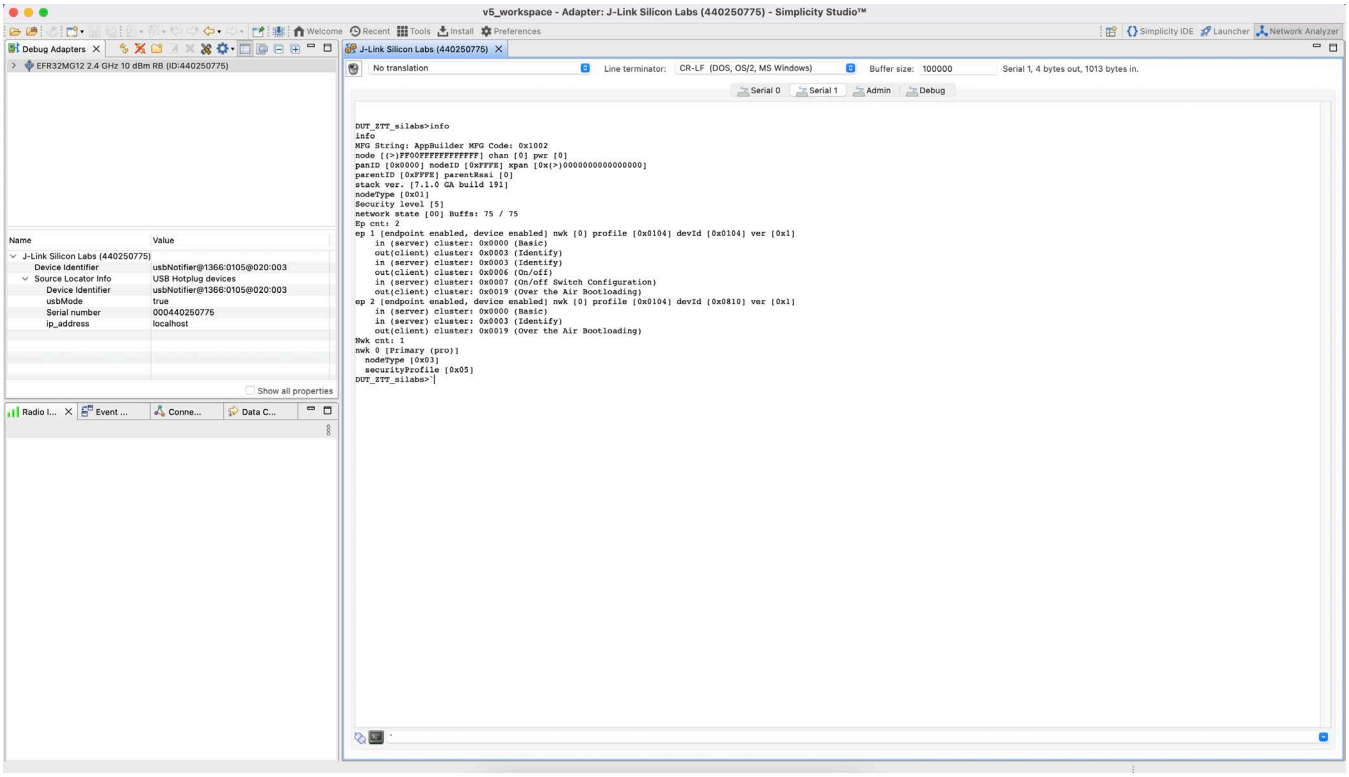


Figure 4-6. DUT's Virtual Console

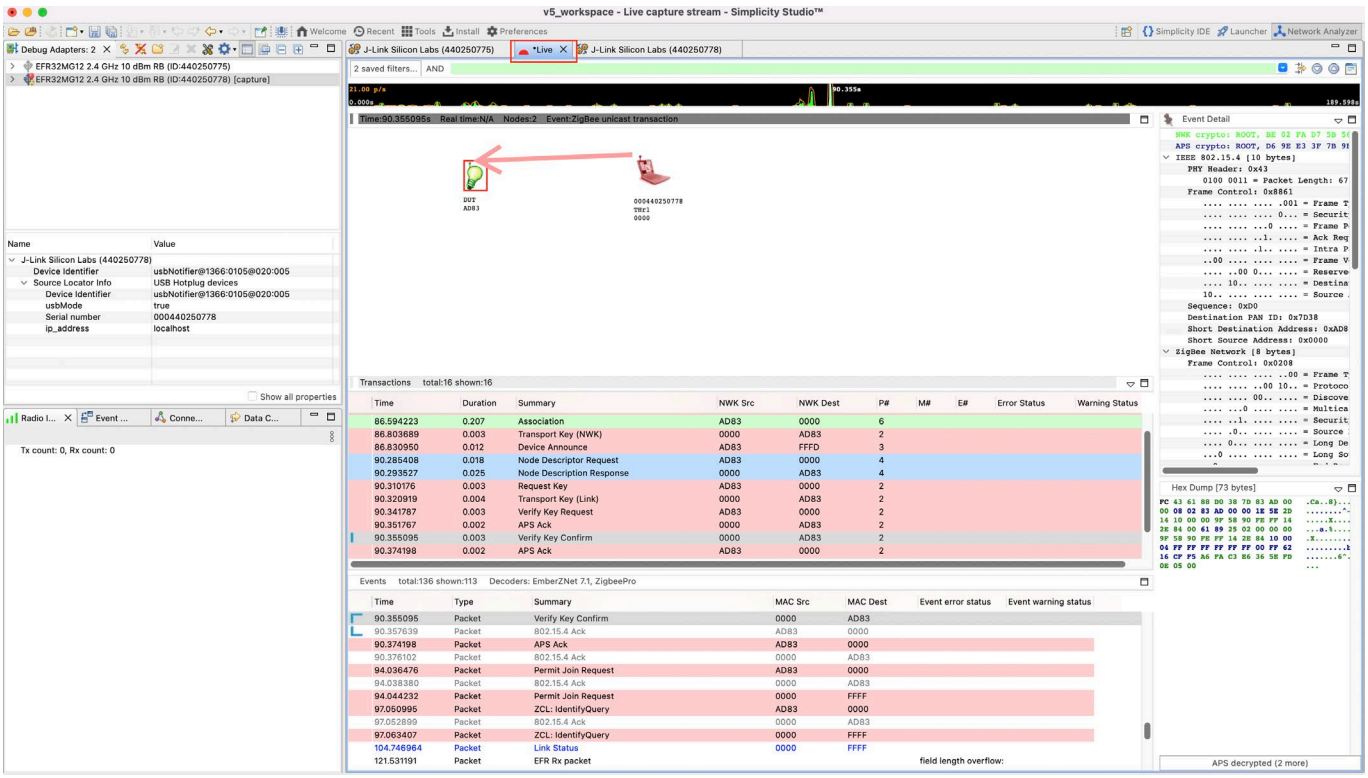


Figure 4-7. Sniffing GUI in Network Analyzer

Simplicity Studio integrates a virtual console and a network analyzer, available through the Tools drop down menu in the launcher perspective. To open the console for a TH, a tool like PuTTY could be used. Make sure to use the parameters specified in section 3.3.2 Using the Xmodem Protocol.

4.2 Completing BDB PICS and PIXIT Items

The items in PICS and PIXIT items determine which test cases are run on a particular device. Hence, it is vital to build a comprehensive understanding of DUT's behavior. To complete the items, refer to two official documents: *16-02828-012-PRO-BDB-v3.0.1-Specification*(**[R6]**, BDB-Spec-Doc for short) *16-02829-015-PRO-BDB-v3.0.1-PICS* (**[R7]** BDB-PICS-Doc for short). Both PICS and PIXIT are included in the BDB-PICS-Doc. Three complete BDB-PICS example documents for Z3ColorControlLight, Z3Gateway and Z3SmartOutlet reference designs are included as supplemental information to this application note. The zip file containing these examples can be accessed through the EmberZNet SDK's Getting Started Application Note list in Simplicity Studio.

The following figure shown an example PICS table, including five columns:

1. **Item Number:** The unique number matches the one in the ZTT PICS tab.
2. **Feature:** A brief description of the related item.
3. **Reference:** Points each item to a detailed specification in BDB-Spec-Doc.
4. **Status:** Shows if an item is mandatory (M), optional (O), or prohibited (X), based on whether another item reflects the DUT's behavior.
5. **Support:** Indicates if the feature reflects the DUT's behavior.

5.1 [ZLT] ZigBee logical device types

Item number	Feature	Reference	Status	Support
ZLT1	Is the logical device type specified as a ZigBee coordinator?	[R1]/2.5.4.5.1	O.1	Yes/No
ZLT1.1	Does the node encompass the role of the Trust Center?	6.1	ZLT1: M ZLT2: X ZLT3: X	Yes/No
ZLT1.2	Does the node form a centralized security network?	6.1	ZLT1: M ZLT2: X ZLT3: X	Yes/No
ZLT1.3	Does the node NOT attempt to join another network?	6.1	ZLT1: M	Yes/No
ZLT2	Is the logical device type specified as a ZigBee router?	[R1]/2.5.4.5.2	O.1	Yes/No
ZLT2.1	Can the router node join another network?	6.1	ZLT2: M	Yes/No
ZLT2.2	Does the node form a distributed network?	6.1	ZLT1: X ZLT2: O ZLT3: X	Yes/No
ZLT3	Is the logical device type specified as a ZigBee end device?	[R1]/2.5.4.5.5	O.1	Yes/No
ZLT3.1	Can the end device node join another network?	6.1	ZLT3: M	Yes/No
ZLT3.2	Is the end device sleepy?	-	ZLT3: O	Yes/No
ZLT4	Can the node switch between ZLT1 and ZLT2 types under application control.	6.1	O.1	Yes/No

Figure 4-8. A Screenshot of PICS Items

4.2.1 Deciding if the DUT Supports a PICS item

Whether or not the DUT supports a PICS item can be decided in two ways:

- Using Feature and Status
- Using Application Configuration (component/plugin) Settings

Using the "Feature" and "Status" Description

Take ZLT1 in Figure 4-8 as an example. Its feature says, "*Is the logical device type specified as a ZigBee coordinator*?" It could be straightforwardly answered with a No if the device is an end-device. Notice in the Status column, certain conditions must be fulfilled. For instance, in ZLT1.2, the status indicates that this item is prohibited (X) if the DUT satisfies ZLT2 or ZLT3.

Extra care also needs to be taken when dealing with items with an O.1 status, such as ZLT1. It usually means only one item can be chosen from a group of items marked by O.1. There are notes under a PICS table explaining each O.x status. For example, the notes for O.1 are "*A node SHALL support one of ZLT1 or ZLT2 or ZLT3 or (ZLT1 and ZLT2, switchable under application control)*".

A status of M (mandatory) denotes a must-have feature in the DUT. For example, MRD1 (shown in the following figure) must be supported by the DUT, which can be explained by the one-to-one mapping of *Active_EP_req* and *Active_EP_rsp* illustrated in the figure. The keyword *SHALL* has the same effect as Required or Mandatory. Therefore, any item defined by SHALL become a must-have feature. If only BDB-PICS-Doc is considered, it is applicable throughout the document that an M-status indicates a Yes-support.

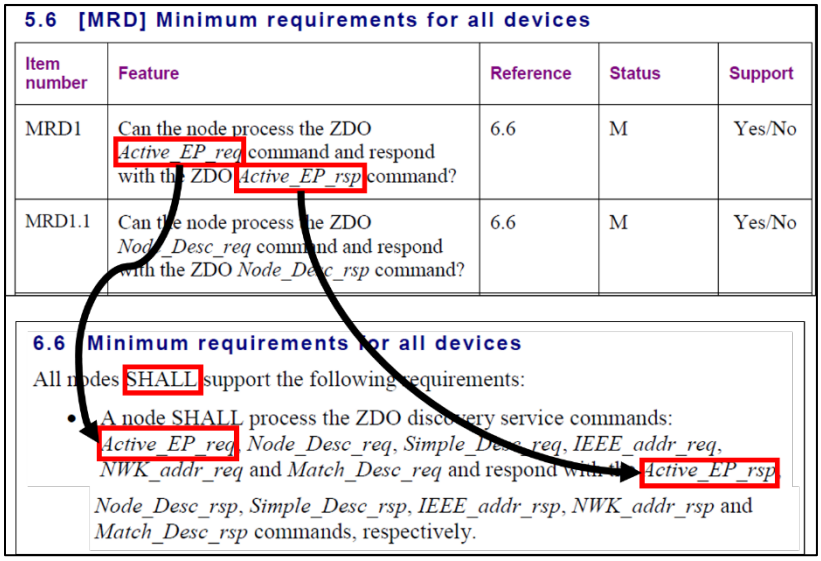


Figure 4-9. Mandatory Item Correlation Between BDB-PICS-Doc Section 5.6 and BDB-Spec-Doc Section 6.6

Using Application Configuration (component/plugin) Settings

Some BDB items are application-dependent, and they are usually marked by O (i.e., optional) in the status column. Therefore, component/plugin settings need to be investigated. Take ZLT2.2 in Figure 4-8 as an example. The Silicon Labs Smart Outlet reference design acts as a router, but it does not form a distributed network. This could be determined by investigating the Application Configuration in

Simplicity Studio. If neither *Network Creator* nor *Network Creator Security* components/plugins are enabled in the Smart Outlet, item ZLT2.2 should be a No. This is shown in the following figures.

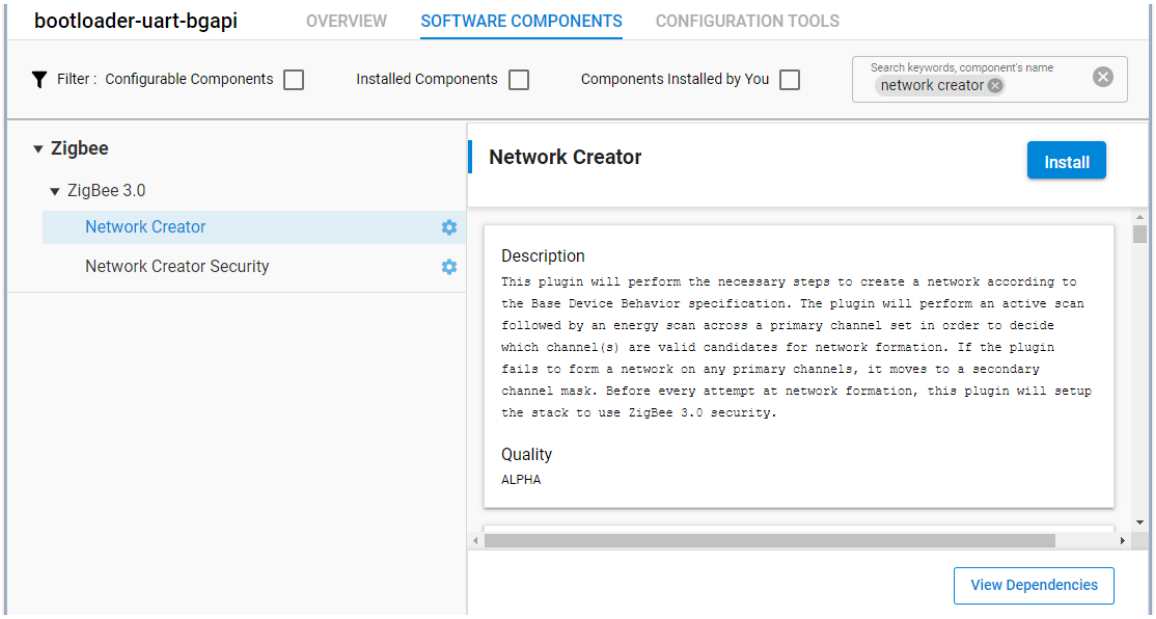


Figure 4-10. Z3SmartOutlet .slcp File

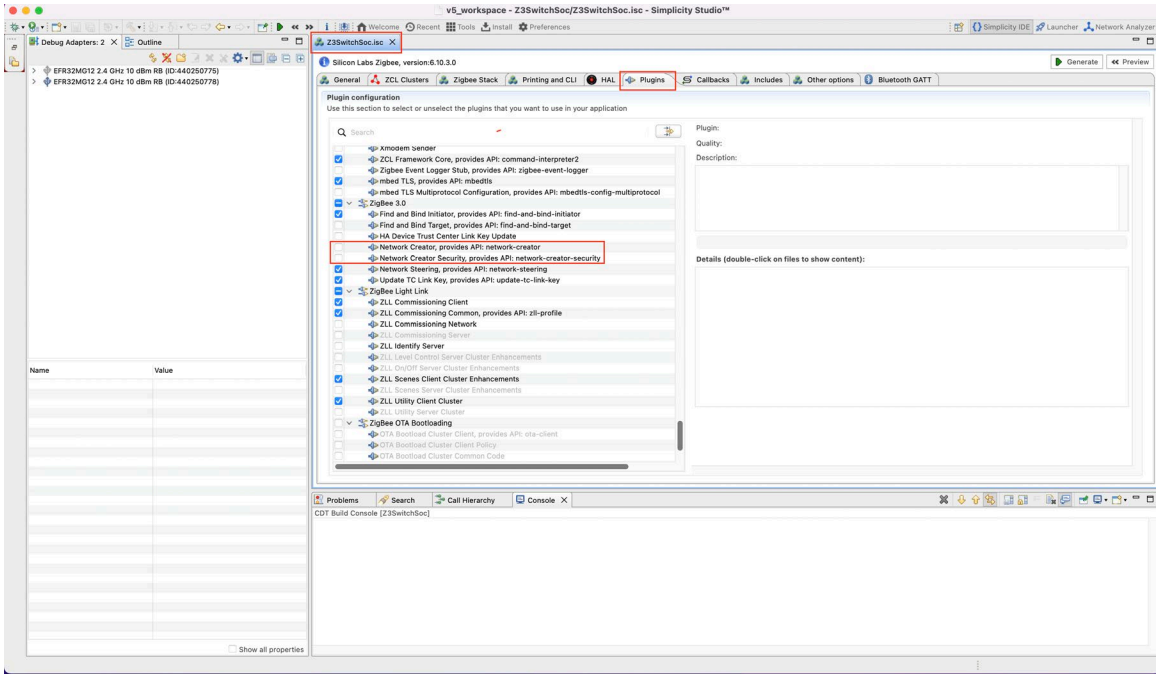


Figure 4-11. Z3SmartOutlet .isc File

4.2.2 Completing the PIXIT Items

There are three major classes of PIXIT items in BDB-PICS-Doc: IA (*internal attributes*), CC (*commissioning combinations*) and M (*miscellaneous*). Users could leave CC and M items as No as they wouldn't affect ZTT procedures, but it is always best to understand the behavior with respect to CC and M items. The BDB-Spec-Doc can help with completing PIXIT items.

As most of IA items are mandatory, we use IA item completion as an example. The following figure presents all the IA items. The reference column points each item to the BDB-Spec-Doc as PICS items. For example, IA1 support equals to 0xffff in the Z3Gateway reference design, since its behavior matches the specification "*If* `bdbCommissioningGroupID` *is equal to 0xffff, any bindings will be created as unicast*".

Item number	Feature	Reference	Status	Support
IA1	*bdbCommissioningGroupID*: What is the list of groups the node is able to use for finding & binding?	5.3.1	M	*"0xffff, List of group IDs"*
IA2	*bdbJoinUsesInstallCodeKey*: Does the Trust Center policy require all nodes to join using an install code?	5.3.6	ZLT1: M	*"True, False"*
IA3	*bdbPrimaryChannelSet*: What is the primary channel set?	5.3.10	M	*"0x00000000, Channel mask"*
IA4	*bdbScanDuration*: What is the scan duration?	5.3.11	M	*"8-bit integer"*
IA5	*bdbSecondaryChannelSet*: What is the secondary channel set?	5.3.12	M	*"0x00000000, Channel mask"*
IA6	*bdbTCLinkKeyExchangeAttemptsMax*: What is the maximum number of attempts a node will try to exchange its Trust Center link key?	5.3.14	ZLT2: M ZLT3: M	*"8-bit integer"*
IA7	*bdbTCLinkKeyExchangeMethod*: What is the Trust Center link key exchange method?	5.3.15	ZLT2: M ZLT3: M	*"0x00, 0x01"*
IA8	*bdbTrustCenterNodeJoinTimeout*: What is the Trust Center node join timeout?	5.3.16	ZLT1: M	*"8-bit integer"*
IA9	*bdbTrustCenterRequireKeyExchange*: Does the Trust Center's policy require a node to exchange its initial link key with a new link key generated by the Trust Center?	5.3.17	ZLT1: M	*"True, False"*

Figure 4-12. IA Items (PIXIT)

Item IA6 relates to EmberZNet stack implementation. The value `bdbTcLinkKeyExchangeAttemptsMax` in the stack is 1, and it indicates the DUT would try once for link-key exchange. An additional item on ZTT software does not appear on BDB-PICS-Doc. This item, known as IA5B, asks for clarification of "*DUT has a* `bdbSecondaryChannelSet` *!= 0 (bool)*". By default, this value is 0. However, it should be set to 1, indicating the DUT does have a secondary channel set.

4.3 Completing a Cluster's PICS and PIXIT Items

The official PICS documents are created per cluster. The PICS documents for individual clusters can be found in **[R2]**. The layout of those documents is similar to BDBs, except that test specifications are integrated in the docs. To better present the PICS and PIXIT selection, CSA also provides XML files to be easily customized. As for the PICS documents, the XML files should be created per cluster, and they can be downloaded from the sub-folders of **[R12]**. Example XML files for the Z3ColorControlLight cluster are included as supplemental information to this application note. The zip file containing these examples can be accessed through the EmberZNet SDK's Getting Started Application Note list in Simplicity Studio.

The following text presents an example of the Basic cluster in XML format. Notice that in the `<support>` tag, `true` should selected if the feature is supported, and vice versa.

```
<!-- General cluster information -->
       <name>Basic</name>
       <clusterId>0x0000</clusterId>
       <picsRoot>B</picsRoot>
<!-- Cluster usage -->
       <usage>
              <picsItem>
                     <itemNumber>B.S</itemNumber>
                     <feature>Does the device implement the basic cluster as a server?</feature>
                     <reference>3.2.2</reference>
                     <status>O</status>
                     <support>true</support>
              </picsItem>
              <picsItem>
                     <itemNumber>B.C</itemNumber>
                     <feature>Does the device implement the basic cluster as a client?</feature>
                     <reference>3.2.3</reference>
                     <status>O</status>
                     <support>false</support>
              </picsItem>
      </usage>
```

It is straightforward to map applications in Simplicity Studio to a clusters' PICS items. The following figure illustrates such a mapping using the On/off cluster as an example. In Simplicity Studio, open the application project and navigate to *ZCL clusters*. Click **On/off.** You should be able to find the cluster role as *Server* and view the implemented attributes and commands in detail. In the *Attributes* and *Commands received* sections of the PICS document, the item should be supported if the related attributes/commands are ticked in Simplicity Studio. The following figure shows the Attributes and Commands in Simplicity Studio correlated to the PICS items.

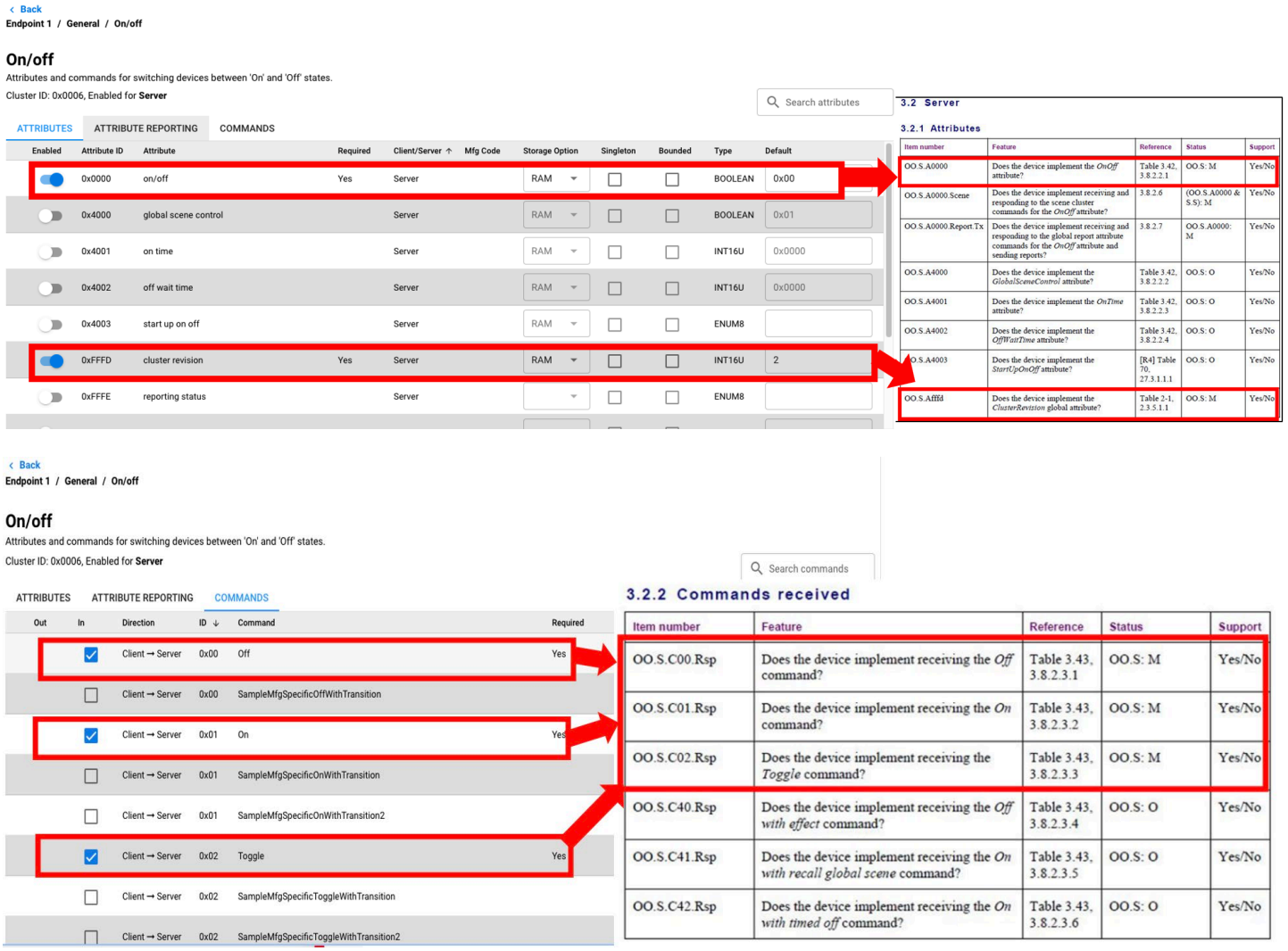


Figure 4-13. Mapping the On/Off Cluster's Attributes and Commands to PICS items

The PICS document also contains references for each item that point to related sections in *07-5123-06-zigbee-cluster-library-specification* **[R8]**. The following figure presents the detail with respect to `OO.S.A0000.Scene` and OO.S.A000.Report.Tx. As Z3ColorControlLight implements Scenes cluster as a server, the item `OO.S.A0000.Scene` is a mandatory feature. The attribute reporting section indicates OnOff attribute should be reportable, and therefore `OO.S.A0000.Report.Tx` is mandatory as well.

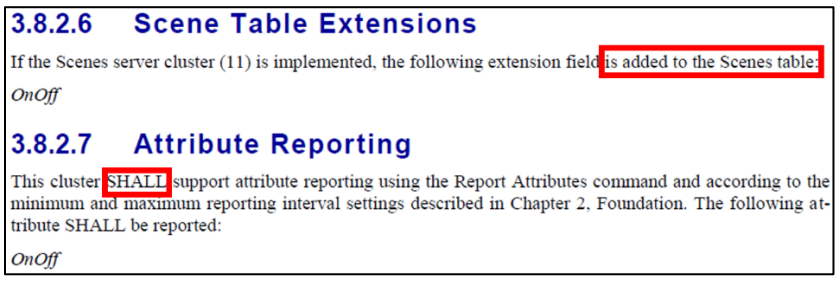


Figure 4-14. Cluster-Related Sections in the ZCL Specification

PIXIT items are enclosed in each clusters' PICS document. Not all the clusters have PIXIT items. For example, the Color Control cluster contains PIXIT items regarding hue, whereas the Basic cluster has none. The following text presents PIXIT examples of the Color Control cluster using XML format. By referring to section 5.2.2.2.1 of **[R8]** and CIE1931 model (https://upload.wikimedia.org/wikipedia/commons/3/3b/CIE1931xy_blank.svg), we could calculate the approximate X coordinate by: 0.7 * 65536 = 45875(Dec) or 0xB333(Hex).

```
<!-- PIXIT -->
 <pixit>
   <pixitItem>
     <itemNumber>CC.PIXIT01.1</itemNumber>
     <feature>Approximate X coordinate for a red hue?</feature>
     <specification>0xB333</specification>
   </pixitItem>
 <pixit>
```

4.4 Selecting Suitable Test Cases Based on the PICS

Once the PICS items are completed, suitable test cases need to be selected. Over-selection of test cases could cause unnecessary failures. However, under-selection of test cases incurs incomplete validation of conformity.

Note that the Zigbee Test Tool will build up a list of tests for your device based on the PICS and PIXIT data provided. While in many cases these will be the complete suite of tests, you should check the Zigbee documentation to verify that these are all of the test you are required to run.

CSA provides cross references for test case selection. The BDB PICS and Cluster PICS to test case cross references are described below.

4.4.1 BDB PICS to Test Case Cross Reference

A cross reference spreadsheet can be found at **[R9]**. The following figure illustrates the test cases for PICS item GRC1. Test case CN-NSA-TC-04 covers PICS item GRC1 (indicated by C), while test case CN-NSA-TC-03 does not. By scanning through the spreadsheet, you should be able to find a suitable test case to cover for the applicable PICS items.

	PICS to Test Case Mapping for: BDB PICS (15-0282) and BDB test specification (14-0439)				
Item number	Item description	Reference	Status	CN-NSA-TC-03	CN-NSA-TC-04
GRC1	Does the node support network steering?	6.5	M		C

Figure 4-15. Cross Reference for BDB Test Cases

4.4.2 Clusters PICS to Test Case Cross Reference

The PICS document for individual clusters contains a cross reference table. For example, the on/off cluster PICS document (*docs-15-0310-05-pfnd-0x0006-OnOff-Cluster-Test-Specification)* presents the table in shown in the following figure.

5 Annex A: PICS to test case cross reference

5.1 Server

	Test case						
PICS	OO-TC-01G	OO-TC-01S	OO-TC-02S	OO-TC-03S	OO-TC-04S	OO-TC-05S	OO-TC-06S
OO.S	X	X	X	X	X	X	X
OO.S.A0000		X	X	X	X	X	X
OO.S.A0000.Scene					X		
OO.S.A0000.Report.Tx						X	
OO.S.A4000		X		X			
OO.S.A4001		X		X			
OO.S.A4002		X		X			
OO.S.A4003		X					X
OO.S.Afffd	X						
OO.S.C00.Rsp			X	X	X	X	X
OO.S.C01.Rsp			X	X	X	X	X
OO.S.C02.Rsp			X				
OO.S.C40.Rsp				X			
OO.S.C41.Rsp				X			
OO.S.C42.Rsp				X			

Figure 4-16. Cross Reference for On/Off Cluster

The figure shows that seven cases in total could be used for testing. However, at least one case should not be chosen for the Z3ColorControlLight reference design. As shown in section 4.2 Completing a Cluster's PICS and PIXIT Items, PICS item `OO.S.A4000`, `OO.S.A4001` and `OO.S.A4002` have not been configured as Yes, whereas the figure shows that tests in OO-TC-03S will be conducted against these attributes (shown by the 'X' mark). In other words, if OO-TC-03C is executed, these three unimplemented attributes would cause a failure. The test script for OO-TC-03C, running in ZTT software, could be investigated to prove it tries to read the value of attribute 0x4000 (`OO.S.A4000`). A code snippet is presented below (the original script can be downloaded from **[R10]**). In step 1b shown below, attribute 0x4000 is collected via "*zcl global read 0x0006 0x4000*".

```
print step 1b
      print comment TH CLIENT unicasts a ZCL read attributes command frame to DUT SERVER to read the
OnOff and GlobalSceneControl attributes
        ……
        TH_CLIENT > zigbee command raw zcl global read 0x0006 0x4000
        TH_CLIENT > zigbee command raw send [NWK:SHORTADDRESS] 1 [PIXIT:ENDPOINT]
        zigbee expect packet [PIXIT:MAXIMUMTIMEOUT] ""ReadAttributesResponse: Cluster 0x0006 Attrib-
ute   0x4000"" {   cs=0,   cluster=0x0006, command=0x01,   attributeeidentifier=0x4000,   attributesta-
tusread=0x00, datatype=0x10, value=0x01 }
```

4.5 Example of Using ZTT to Run a Test Case

This section uses the on/off cluster test as an example. The DUT is based on Z3ColorControlLight reference design. The following two figures show the implemented attributes and commands for the on/off cluster.

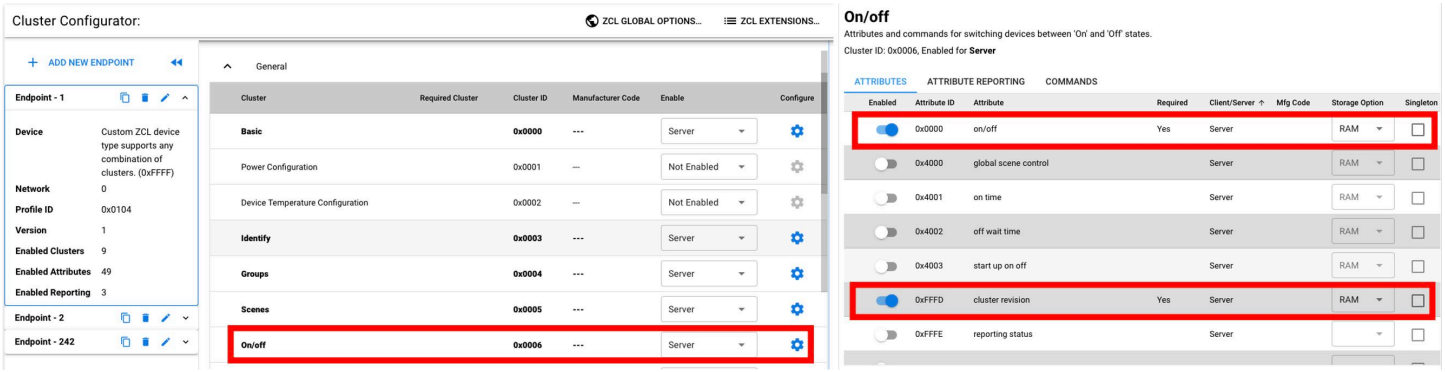


Figure 4-17. Implemented Attributes

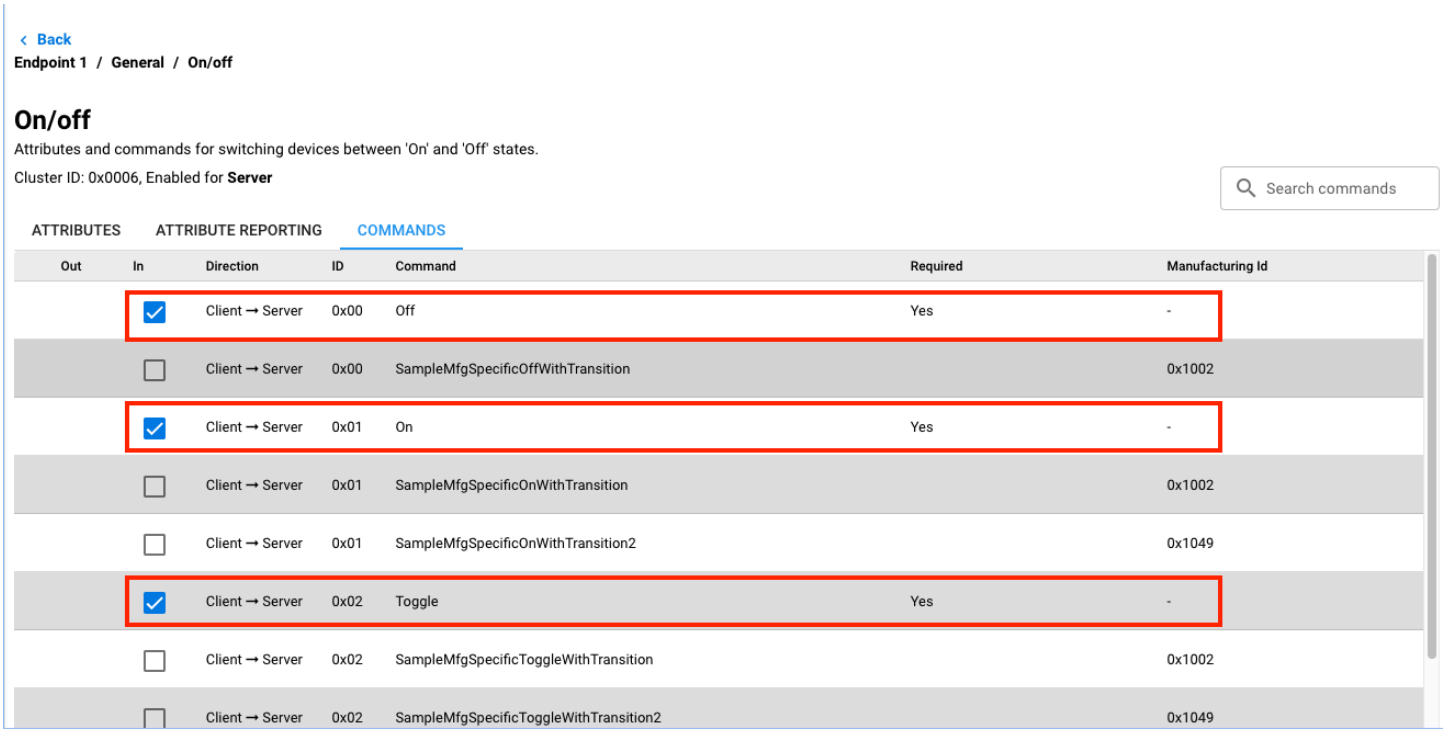
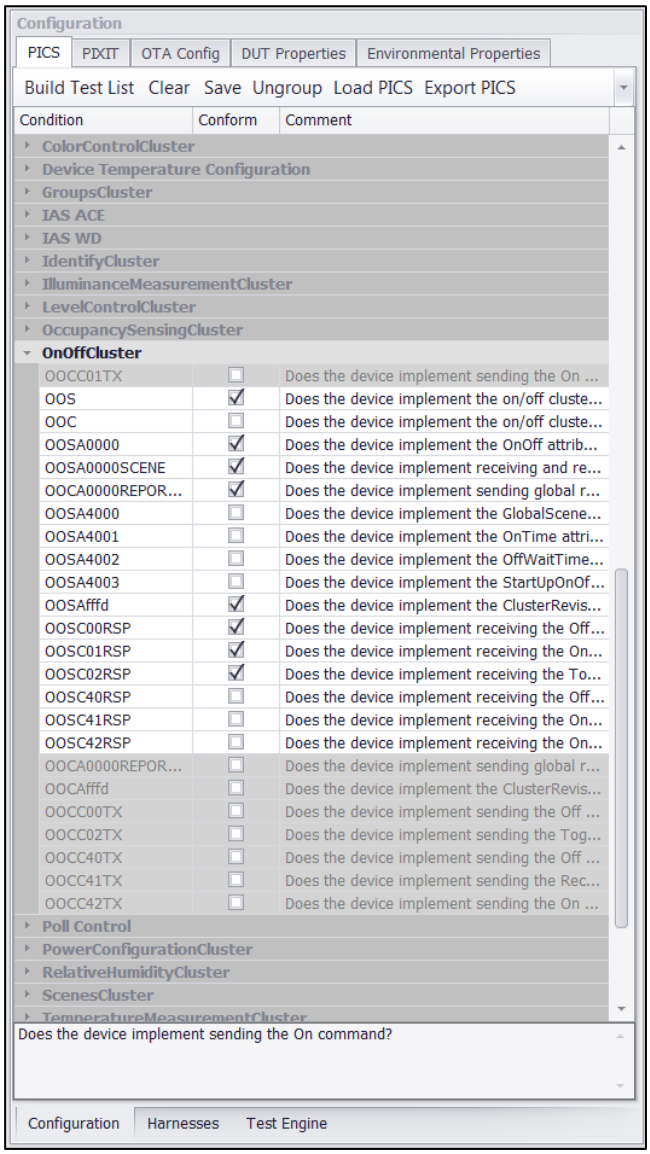


Figure 4-18. Implemented Commands

These attributes and commands can be mapped to the PICS items (based on *docs-15-0310-05-pfnd-0x0006-OnOff-Cluster-Test-Specification*) using the method introduced in section 4.2 Completing a Cluster's PICS and PIXIT Items. The following figure illustrates the PICS items (a) configured in the ZTT software. The common PIXIT items (b) also need to be filled to make sure ZTT operates correctly.

a)



b)

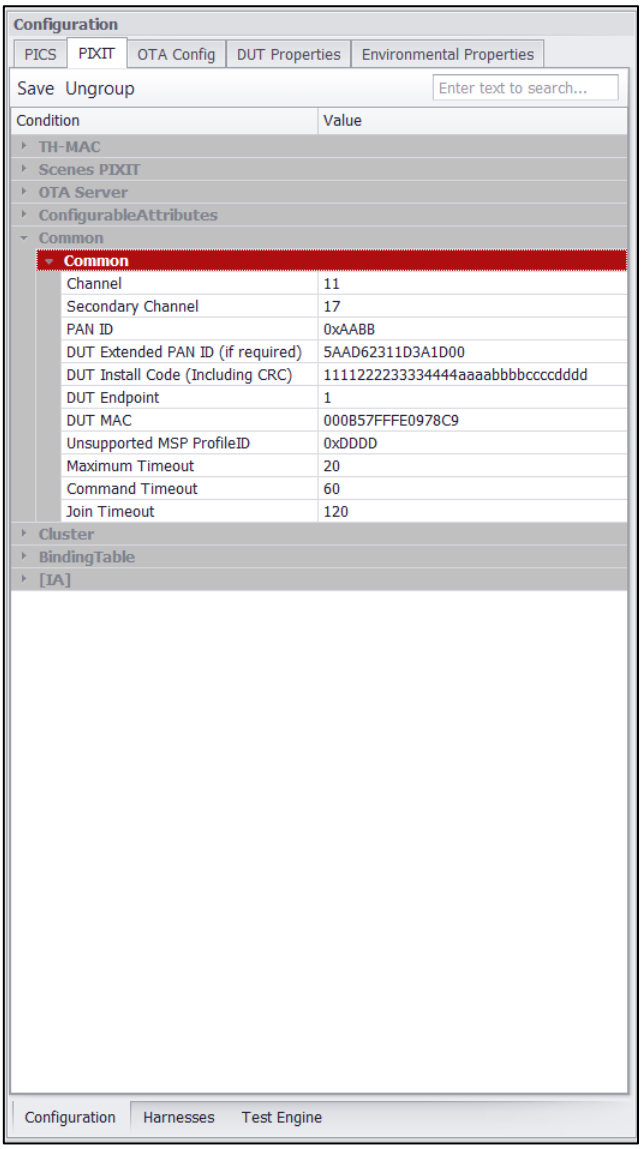


Figure 4-19. PICS and PIXIT Configuration in ZTT

As the on/off cluster is implemented as a type 1 server, it must join a TH network whose role is a client. The client role assignment is made in ZTT software's Test Harness configuration window, shown in the following figure. When running cluster tests, it is recommended to use the Router/Coordinator test harness firmware.

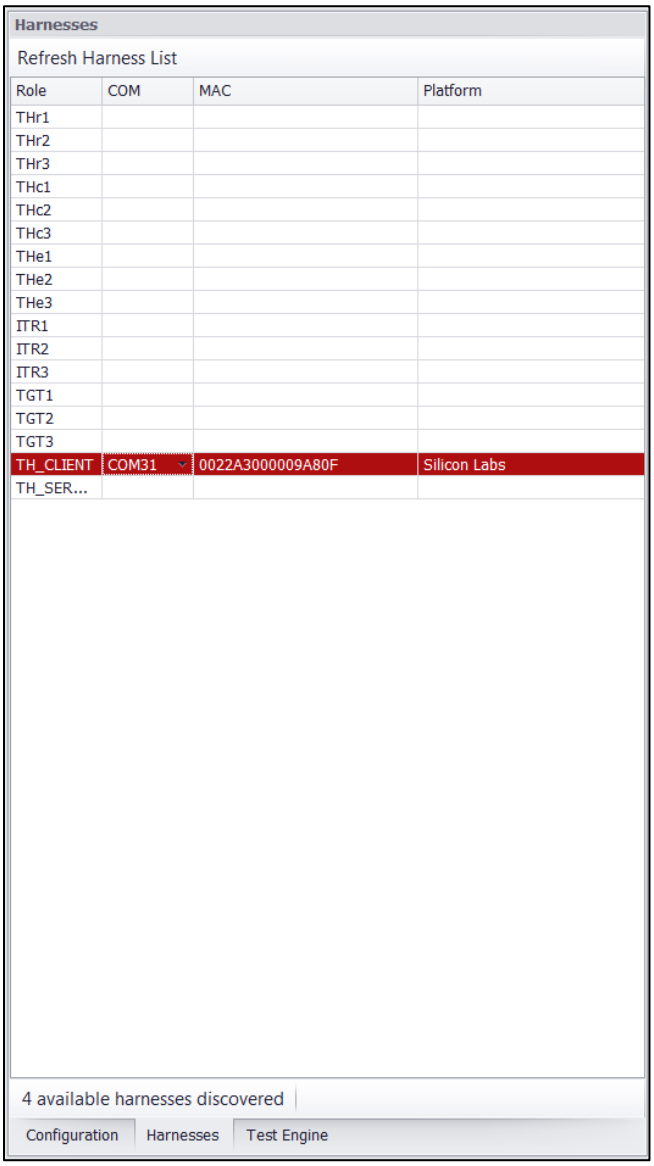


Figure 4-20. TH configuration

Follow the method described in section 4.4 Selecting Suitable Test Cases Based on the PICS to pick the suitable test cases based on the PICS items, then add those test cases to ZTT's Test Engine as shown in the following figure. It should be noted that the perquisite of

running the following tests is that DUT has already joined the network created by TH_Client. Tips for joining DUT can be found in section 5.4 Tips for Testing a Custom Test Case.

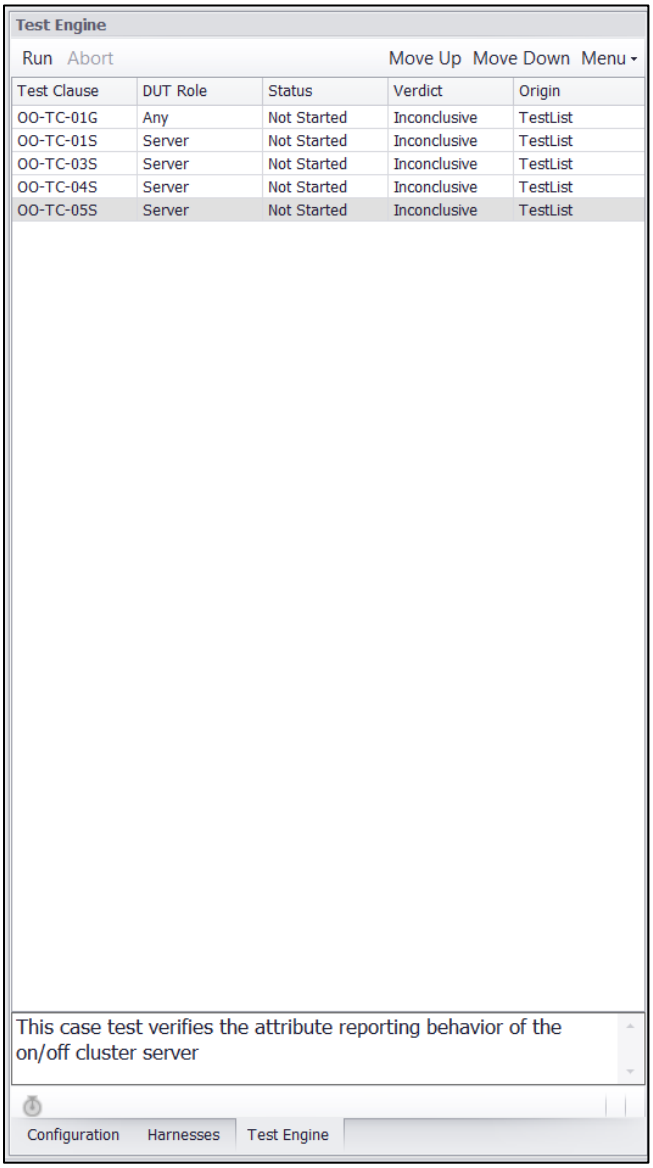
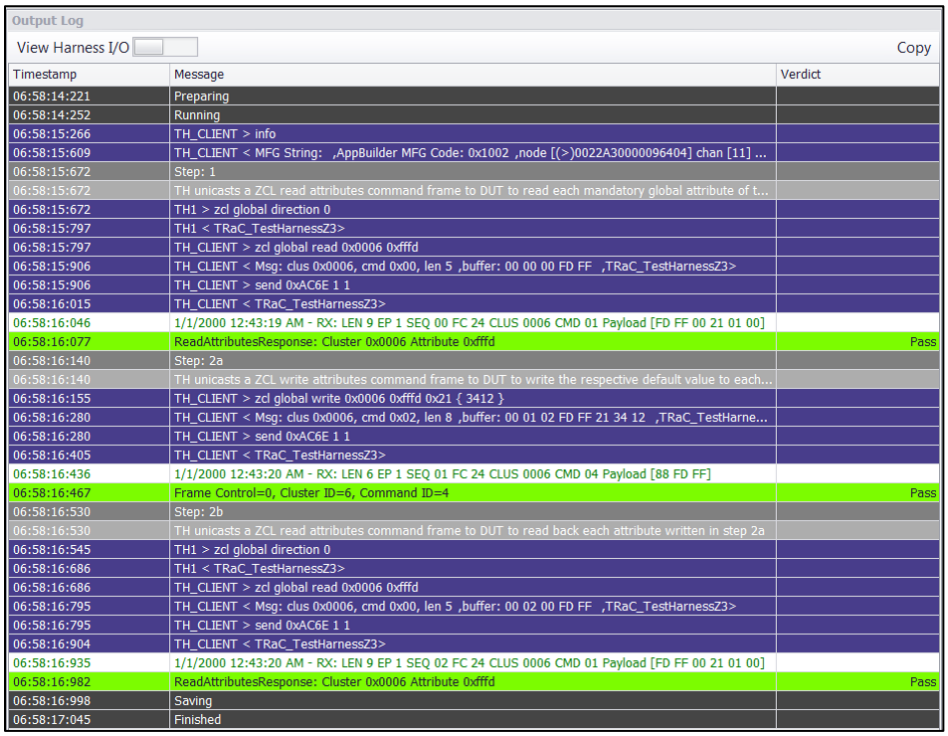


Figure 4-21. Test Cases to be Executed in ZTT

Click **[Run]** in the test engine tab. The test cases are executed sequentially. An example result is shown in the following figure. The figure also shows a list of debug information. When investigating a failure cases, you can start by clicking **[Copy]** to duplicate the debug information and pasting it into any text editor for script and payload checking.


Output Log

View Harness I/O Copy

Timestamp	Message	Verdict
06:58:14:221	Preparing	
06:58:14:252	Running	
06:58:15:266	TH_CLIENT > info	
06:58:15:609	TH_CLIENT < MFG String: ,AppBuilder MFG Code: 0x1002 ,node [(>)0022A30000096404] chan [11] ...	
06:58:15:672	Step: 1	
06:58:15:672	TH unicasts a ZCL read attributes command frame to DUT to read each mandatory global attribute of t...	
06:58:15:672	TH1 > zcl global direction 0	
06:58:15:797	TH1 < TRaC_TestHarnessZ3>	
06:58:15:797	TH_CLIENT > zcl global read 0x0006 0xfffd	
06:58:15:906	TH_CLIENT < Msg: clus 0x0006, cmd 0x00, len 5 ,buffer: 00 00 00 FD FF ,TRaC_TestHarnessZ3>	
06:58:15:906	TH_CLIENT > send 0xAC6E 1 1	
06:58:16:015	TH_CLIENT < TRaC_TestHarnessZ3>	
06:58:16:046	1/1/2000 12:43:19 AM - RX: LEN 9 EP 1 SEQ 00 FC 24 CLUS 0006 CMD 01 Payload [FD FF 00 21 01 00]	
06:58:16:077	ReadAttributesResponse: Cluster 0x0006 Attribute 0xfffd	Pass
06:58:16:140	Step: 2a	
06:58:16:140	TH unicasts a ZCL write attributes command frame to DUT to write the respective default value to each...	
06:58:16:155	TH_CLIENT > zcl global write 0x0006 0xfffd 0x21 { 3412 }	
06:58:16:280	TH_CLIENT < Msg: clus 0x0006, cmd 0x02, len 8 ,buffer: 00 01 02 FD FF 21 34 12 ,TRaC_TestHarne...	
06:58:16:280	TH_CLIENT > send 0xAC6E 1 1	
06:58:16:405	TH_CLIENT < TRaC_TestHarnessZ3>	
06:58:16:436	1/1/2000 12:43:20 AM - RX: LEN 6 EP 1 SEQ 01 FC 24 CLUS 0006 CMD 04 Payload [88 FD FF]	
06:58:16:467	Frame Control=0, Cluster ID=6, Command ID=4	Pass
06:58:16:530	Step: 2b	
06:58:16:530	TH unicasts a ZCL read attributes command frame to DUT to read back each attribute written in step 2a	
06:58:16:545	TH1 > zcl global direction 0	
06:58:16:686	TH1 < TRaC_TestHarnessZ3>	
06:58:16:686	TH_CLIENT > zcl global read 0x0006 0xfffd	
06:58:16:795	TH_CLIENT < Msg: clus 0x0006, cmd 0x00, len 5 ,buffer: 00 02 00 FD FF ,TRaC_TestHarnessZ3>	
06:58:16:795	TH_CLIENT > send 0xAC6E 1 1	
06:58:16:904	TH_CLIENT < TRaC_TestHarnessZ3>	
06:58:16:935	1/1/2000 12:43:20 AM - RX: LEN 9 EP 1 SEQ 02 FC 24 CLUS 0006 CMD 01 Payload [FD FF 00 21 01 00]	
06:58:16:982	ReadAttributesResponse: Cluster 0x0006 Attribute 0xfffd	Pass
06:58:16:998	Saving	
06:58:17:045	Finished	

Figure 4-22. Output Log Example

Troubleshooting

If a test case fails, first check that the test case has been correctly executed before investigating the application code. The following sections describe some common situations that could cause a failure, and some tips on using ZTT.

5.1 Verify the Device Joins the Network

The DUT could be conducting network-steering after power-up. A full network-steering cycle is as follows:

```
NWK Steering State: Scan Primary Channels and use Install Code
NWK Steering State: Scan Secondary Channels and use Install Code
NWK Steering State: Scan Primary Channels and Use Centralized Key
NWK Steering State: Scan Secondary Channels and Use Centralized Key
NWK Steering State: Scan Primary Channels and Use Distributed Key
NWK Steering State: Scan Secondary Channels and Use Distributed Key
```

Assuming the THr and the DUT start network-steering at the same time, the DUT first tries joining the distributed network in the state of "*Scan Primary Channels and use Install Code*". From the DUT point of view, this will fail. Testers could observe from the ISA3 debugger that the steering continues. However, the ZTT thinks the device is successfully joined and progresses to next step. This will eventually cause no response from the device since it is not even in the network. In most test cases, the ZTT prompts with a question like "*Does DUT start network steering*", and waits for a response. Make sure DUT is joined before giving a positive answer.

5.2 Tests Against Sleepy End Devices

The following are some suggestions when testing sleepy end devices.

Sleepy devices may fail some BDB test cases when a long pending time is instigated on the ZTT software. Testers could shorten the long poll interval to prevent the DUT from sleeping too fast. The following figure illustrates the place to modify the long poll interval.

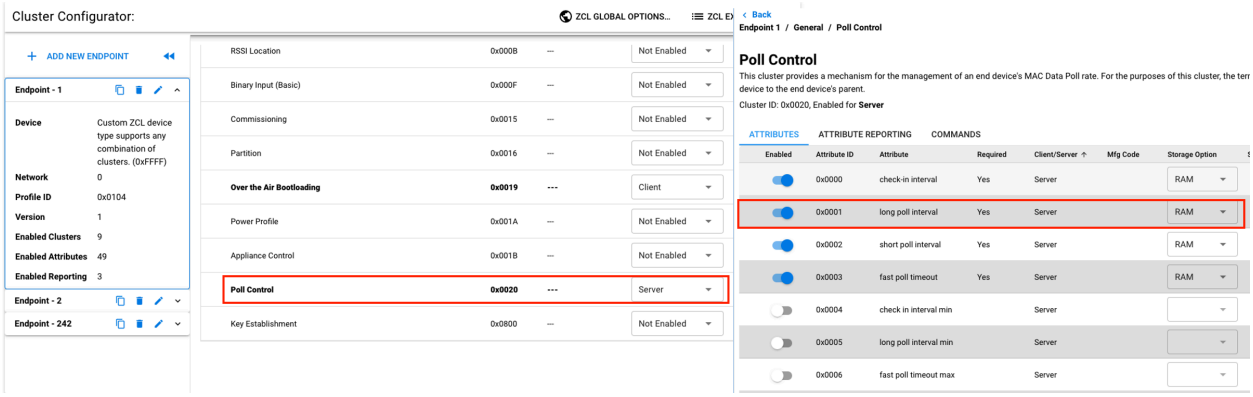
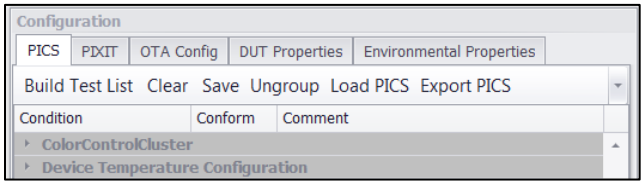


Figure 5-1. Long Poll Interval Modification

5.3 Tips for Using the Zigbee Test Tool

The Zigbee Test Tool Configuration page is shown in the following figure. The buttons are in the row under the tabs.



- Before starting, clear the previously loaded ZTT PICS settings (**Clear**).
- Do not forget to save from time to time (**Save**).
- If you need to share the PICS with a third party use the **Export PICS** button.
- If somebody made the test plan for you use the **Load PICS** button to load it.

5.4 Tips for Testing a Custom Test Case

This section illustrates building a custom test case for the joining procedure. The ZTT user's guide **[R11]** provides brief descriptions and examples of how to run a custom script. Following those guidelines, you can paste the following text into a new script to make the DUT join a network created by TH_Client. Analogously, to join a network created by TH_Server, replace the name `TH_Client` with `TH_Server` in the text.

```
TH_CLIENT > zigbee command raw info
    print step 1
    TH_CLIENT > zigbee command raw network leave
    TH_CLIENT > zigbee command raw option binding-table clear
    TH_CLIENT > zigbee command raw keys clear
    TH_CLIENT > zigbee command raw reset
    prompt wait 5

    print step 2
    TH_CLIENT > zigbee command raw network extpanid { 5AAD62311D3A1D8B }
    TH_CLIENT > zigbee command raw plugin test-harness z3 set-device-mode 0x00

    print step 2a
    TH_CLIENT > zigbee command raw plugin network-creator mask set 1 0
    TH_CLIENT > zigbee command raw plugin network-creator mask set 2 0
    TH_CLIENT > zigbee command raw plugin network-creator mask add 1 11
    TH_CLIENT > zigbee command raw plugin test-harness z3 set-pan-id 0xAABB
    TH_CLIENT > zigbee command raw plugin network-creator start 0

    print step 2b
    TH_CLIENT > zigbee command raw network broad-pjoin 0x00
    TH_CLIENT > zigbee command raw plugin network-steering start 1

    prompt continue Is DUT joined the network?
    prompt verdict Should the test case pass?
```

Reference Material

Note: Official documents from Connectivity Standards Alliance may require membership for full access.

[R1] Certification resources for members:

https://groups.csa-iot.org/wg/members-all/document/folder/31

[R2] PICS document for individual clusters (included in each Cluster Test Specification folder):

https://groups.csa-iot.org/wg/zwg-bdb/document/folder/44

[R3] Zigbee 3.0 Test Tool:

https://groups.csa-iot.org/wg/members-all/document/folder/289

[R4] Certification Web Tool:

https://zigbeecertifiedproducts.knack.com/zigbee-certified

[R5] Zigbee Alliance BDB documents:

https://groups.csa-iot.org/wg/members-all/document/folder/1658

[R6] 16-02828-012-PRO-BDB-v3.0.1-Specification:

https://groups.csa-iot.org/wg/members-all/document/24019

(preview link) https://groups.csa-iot.org/wg/members-all/document/previewpdf/24019

[R7] 16-02829-015-PRO-BDB-v3.0.1-PICS:

https://groups.csa-iot.org/wg/members-all/document/24023

(preview link) https://groups.csa-iot.org/wg/members-all/document/previewpdf/24023

[R8] Zigbee cluster library specification:

https://groups.csa-iot.org/wg/members-all/document/23019

(preview link) https://groups.csa-iot.org/wg/members-all/document/previewpdf/23019

[R9] BDB PICS to test cases cross reference:

https://groups.csa-iot.org/wg/members-all/document/9123

[R10] ZTT v1.0.2.6 Official Test Scripts.zip (included in ZTT 1.0.2.6 package):

https://groups.csa-iot.org/wg/members-all/document/19806

[R11] ZTT user guide (included in ZTT 1.0.2.6 package):

https://groups.csa-iot.org/wg/members-all/document/19806

[R12] Clusters test specification parent folder:

https://groups.csa-iot.org/wg/zwg-bdb/document/folder/44

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com