AN1387: Backing Up and Restoring a Z3 Green Power Combo Gateway

This document describes how to use the the backup and restore feature in a Z3GatewayGPCombo scenario application.

This document is up to date with Zigbee EmberZNet Release 8.0.0.

KEY POINTS

- Working principle of host-NCP backup and restore
- Components used in the function
- Token file format
- Saving and restoring the EUI64 token
- Building the examples
- Testing the feature
- Limitations

1 Backup/Restore Basic Working Principle

The backup and restore of a gateway work on the principle that all the tokens on the operational NCP are taken as backup by the host and then restored on the new NCP hardware. Tokens are read and saved on the Host.

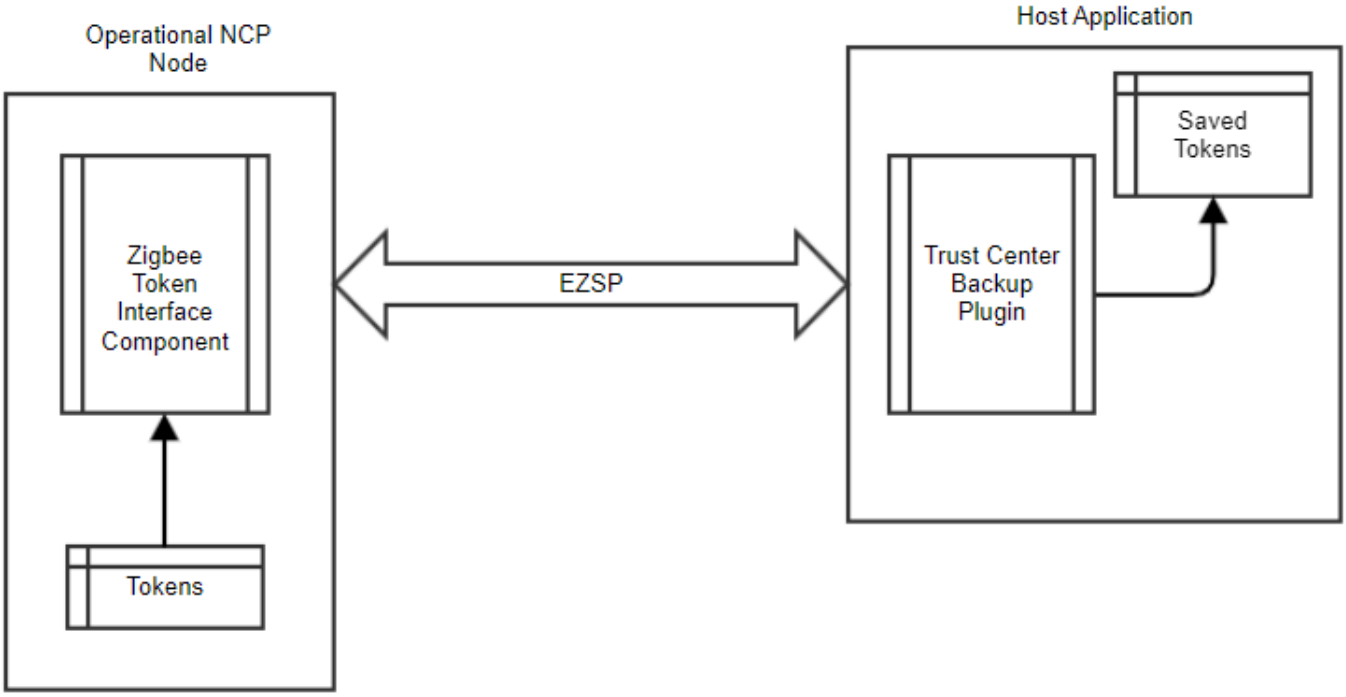


Figure 1-1. NCP Tokens are Saved on the Host

The host then reads the tokens saved on the host in the above step and updates the new NCP hardware tokens with them.

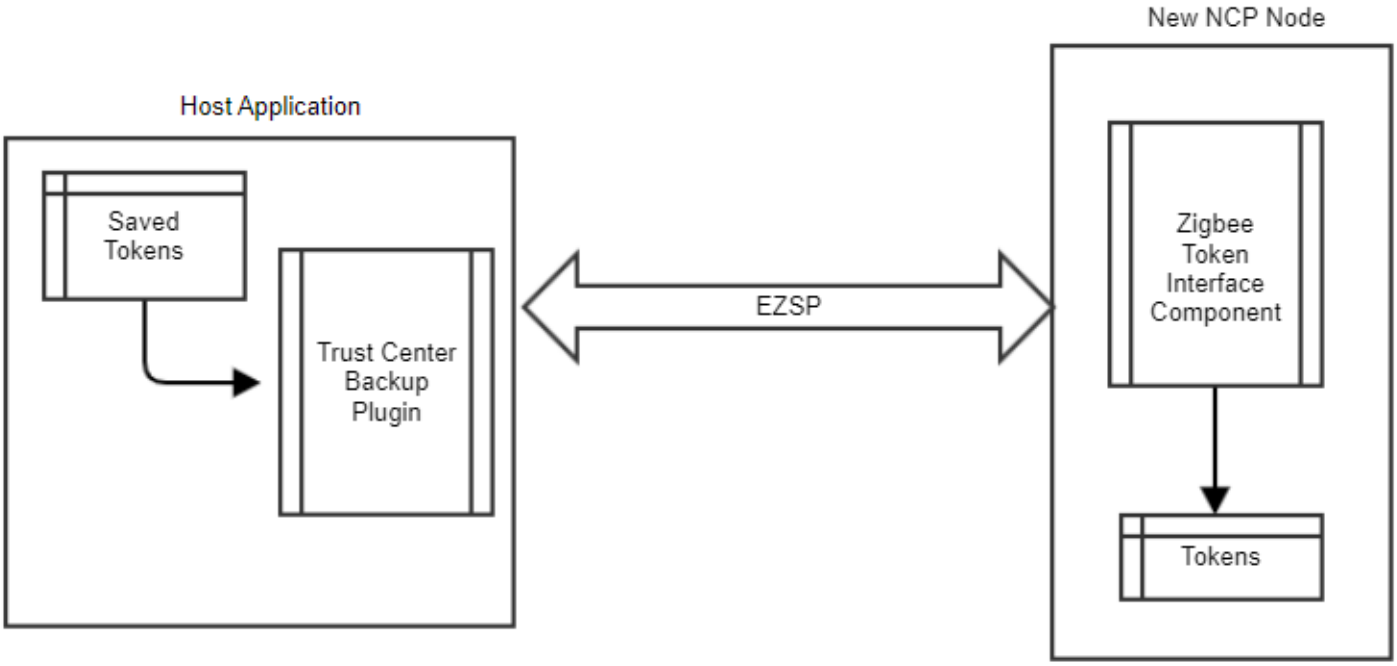


Figure 1-2. Host Updates New Hardware

2 Components

Two main components are used, one on the NCP and one on Host.

2.1 On the NCP Side

The Zigbee **Token Interface** component is used on the NCP side, which requires the **Token Manager** component be installed. **Token Manager using NVM3** should be installed as well. SimEE2 is not supported. If it is used, the APIS in the **Token Interface** component will be stubbed and will return a status of SL_STATUS_NOT_AVAILABLE.

The Zigbee **Token Interface** component provides token access through EZSP commands from the host to read and write the token on the NCP node. The APIs provided are:

```
uint8_t sli_zigbee_stack_get_token_count(void)

sl_status_t sli_zigbee_stack_get_token_info(uint8_t index, sl_zigbee_token_info_t *tokenInfo)

sl_status_t sli_zigbee_stack_get_token_data(uint32_t token, uint32_t index, sl_zigbee_token_data_t
*tokenData)

sl_status_t sli_zigbee_stack_set_token_data(uint32_t token, uint32_t index, sl_zigbee_token_data_t
*tokenData)
```

The component also provides a strong implementation of following function to supply an EUI64 for a restore scenario:

```
void sl_zigbee_get_restored_eui64(sl_802154_long_addr_t eui64)
```

Finally, it supplies a reset function to reset the NCP from the host after all the tokens have been restored. With that, the new NCP node gets initialized with the operational EUI64, the network data from restored tokens, and all working RAM tables are populated.

2.2 On the Host Side

The Zigbee **Trust Center Backup** component is used on the host side. This component is primarily designed to restore a trust center by re-forming the saved network with the same credentials on different hardware. The component performs a backup and restore using the token APIs described in the previous section. These APIs have their EZSP counterparts when called from the host side. The component provides two functions that are implemented to get the token information from the NCP node, save it to a file on the host, and write it back on the new NCP hardware.

As this is designed to restore the entire token system including node EUI, the children on this network do not need to re-establish binding in order to re-form the network.

The following function reads token information and saves it to a file.

```
sl_status_t sl_zigbee_af_trust_center_backup_save_tokens_to_file(const char* filepath)
```

The following function reads the saved file and uses token interface APIs to write tokens to the NCP.

```
sl_status_t sl_zigbee_af_trust_center_backup_restore_tokens_from_file(const char* filepath)
```

To use the file operations and the above routines on a POSIX system, set the component configuration SL_ZIGBEE_AF_PLUGIN_TRUST_CENTER_BACKUP_POSIX_FILE_BACKUP_SUPPORT to 1.

An EZSP reset function is available to reset the NCP node to initialize the restored token data. The host uses this to reset the NCP after restoring all the tokens, so that the network data gets restored on the working RAM tables.

```
void sl_zigbee_ezsp_reset_node(void)
```

Three CLIs are available (one each for the above functions) on the Host as part of the Trust Center Backup component to test the backup and restore:

```
plugin trust-center-backup backup-tokens <file name to store the tokens>

plugin trust-center-backup restore-tokens <the file with the stored tokens>

plugin trust-center-back reset-node
```

3 Host Token File Format

The `sl_zigbee_af_trust_center_backup_save_tokens_to_file` API uses the following format to store all the tokens read from the operational NCP in a binary file.

```
// The binary file format to save the tokens are
//
// Number of Tokens (1 byte)
// Token0 (4 bytes) Token0Size(1 byte) Token0ArraySize(1 byte) Token0Data(Token0Size * Token0Array-
Size)
// :
// :
// TokenM (4 bytes) TokenMSize(1 byte) TokenMArraySize(1 byte) TokenMData(TokenMSize * TokenMArray-
Size)
//
```

Similarly, the function `sl_zigbee_af_trust_center_backup_restore_tokens_from_file` reads the same file in the same format order and writes the tokens to the NCP.

4 Saving and Restoring the Operational EUI64

NCPs in the field may have their EUI64 already programmed in the factory. These cannot be reprogrammed by the standard token restoration mechanism, but rather require special handling.

An additional stack token is created with an NVM3 key name `NVM3KEY_STACK_RESTORED_EUI64`. It is part of the znet token group. This is initialized to blank (0xFFFFFFFFFFFFFFFF) and used to store the EUI of the operational node that is being backed up to the host in the function `sl_zigbee_af_trust_center_backup_save_tokens_to_file`.

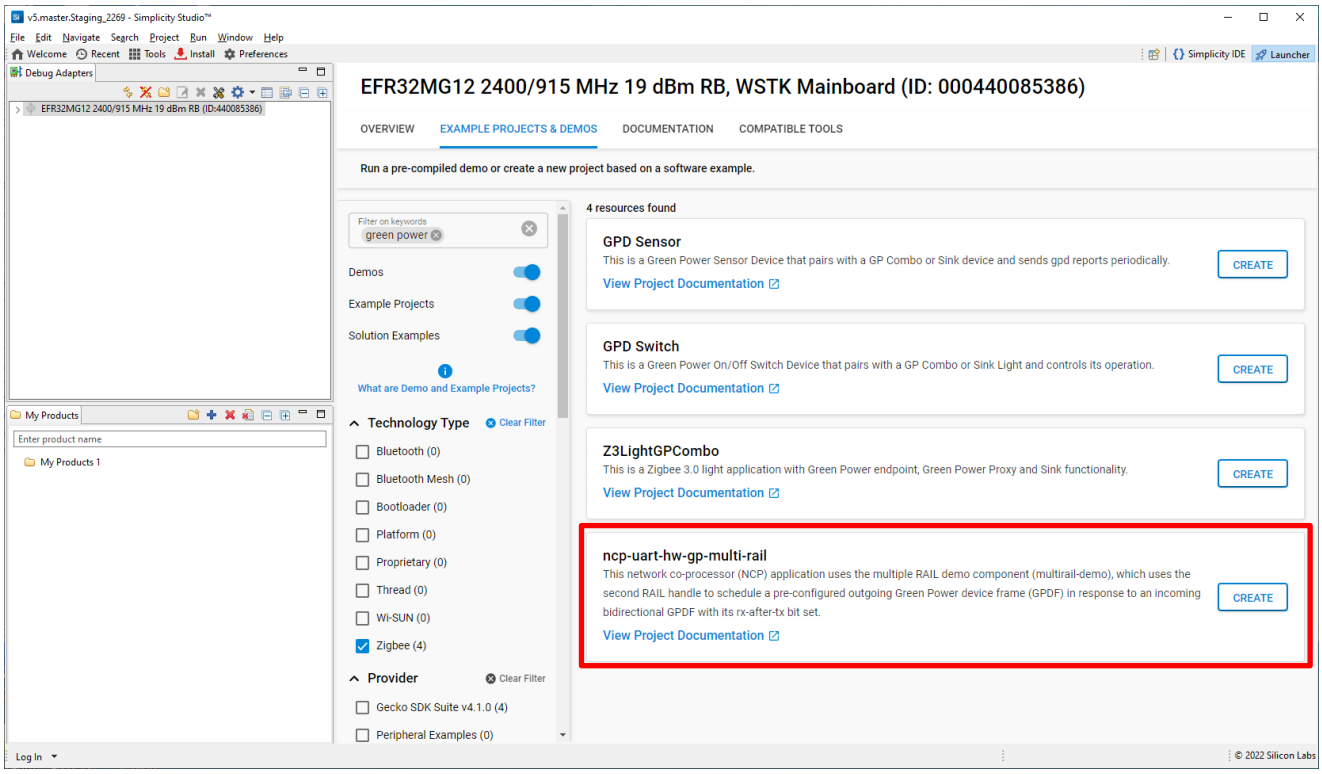
When restoring, the host writes this token on the NCP. The NCP has a strong implementation of `zigbee_get_restored_eui64` as part of the **Token Interface** component. It supplies any restored EUI64 during NCP boot up. The new NCP firmware then starts to use the restored EUI64 instead of its own factory-programmed one.

5 Building the Sample Applications

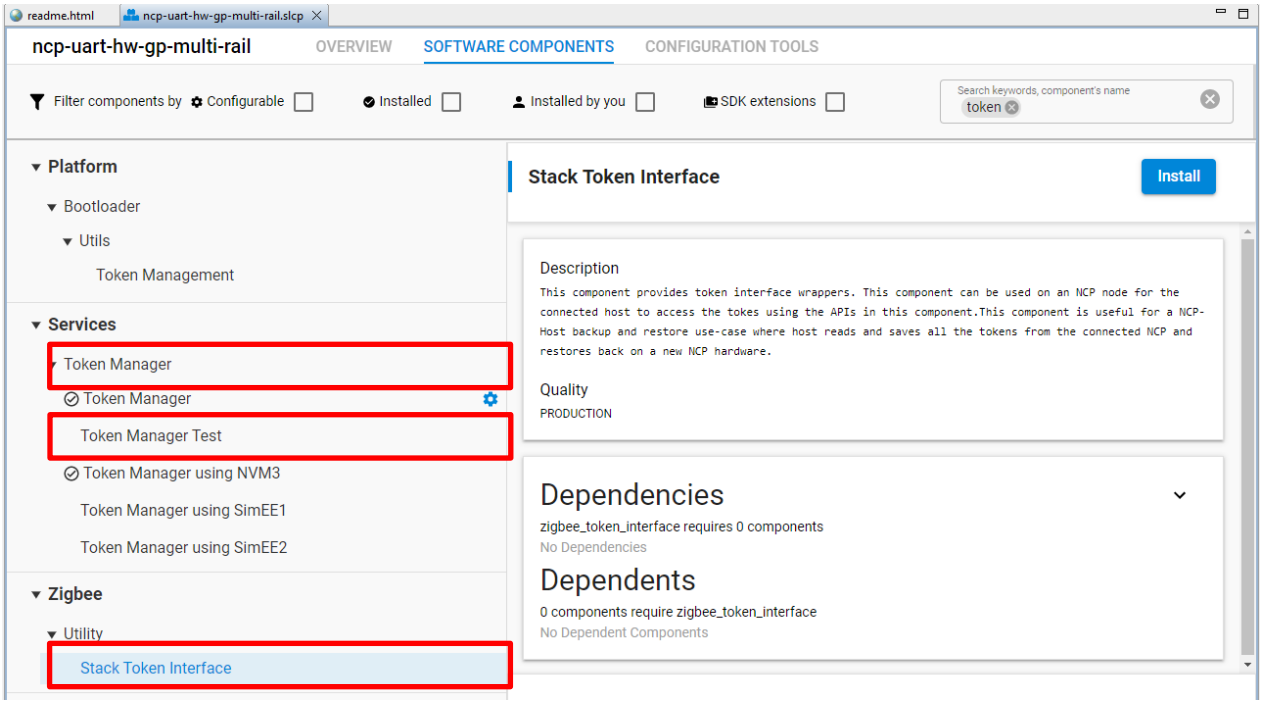
This section describes how to build the sample applications supplied in the Simplicity SDK to test the Z3GatewayGPCombo Backup/Restore feature using Simplicity Studio. If you are not familiar with creating and modifying examples, see *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK 7.0 and Higher*.

5.1 NCP Application

1. Start a project using the following example: ncp-uart-hw-gp-multi-rail.slcp



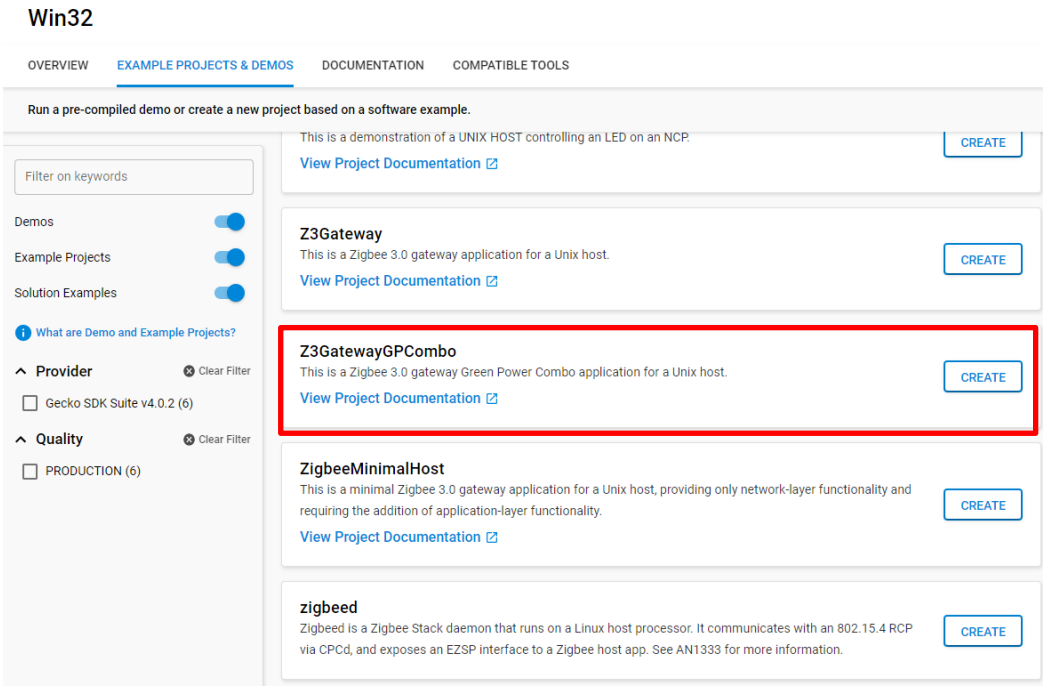
2. On the SOFTWARE COMPONENTs tab install the following, if not already installed:
   - **Stack Token Interface**
   - **Token Manager**
   - **Token Manager using NVM3**



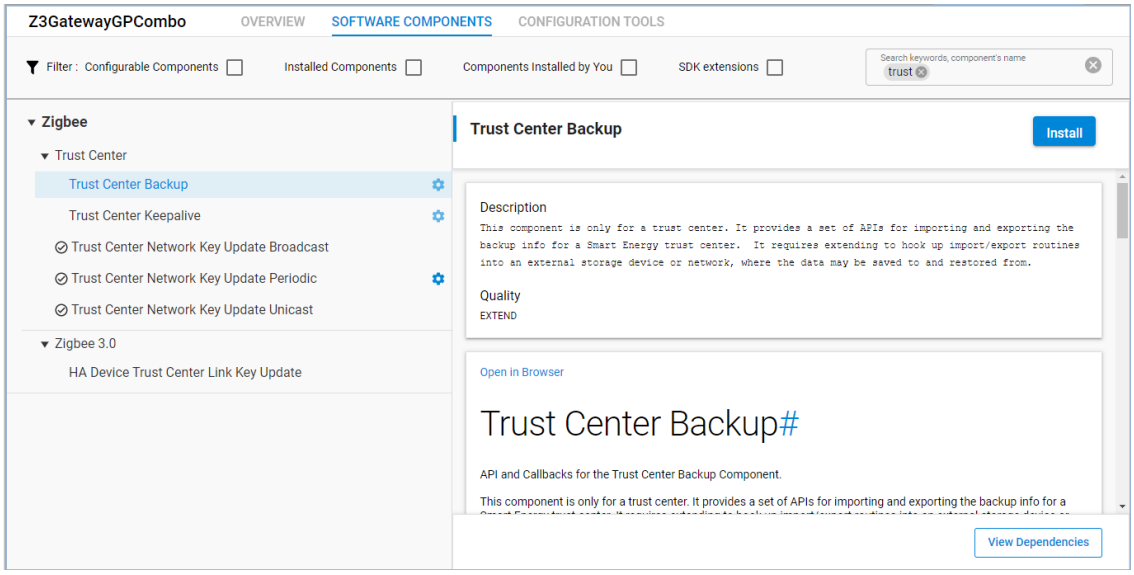
3. Build the NCP application.

5.2 Host Application

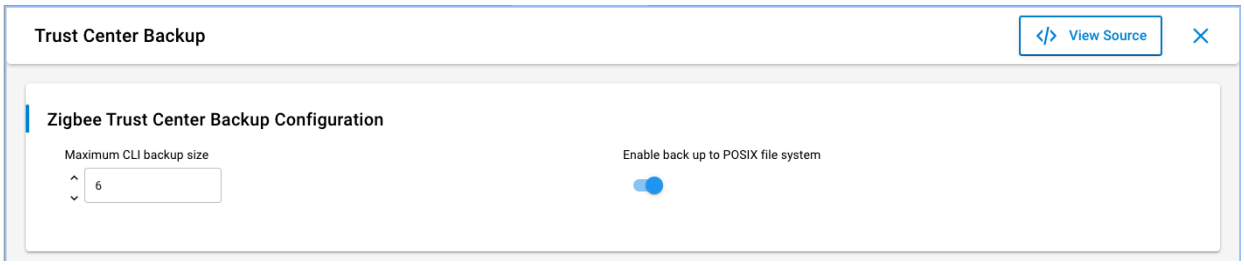
1. Start a project using the following host example: Z3GatewayGPCombo.



2. On the SOFTWARE COMPONENTS tab, install the **Trust Center Backup** component.



3. Click **Configure**. Turn on **Enable back up to POSIX file system**.



4. Build the host application.

6 Testing the Feature

For this test, both the source NCP device and the NCP device to be restored must have the same firmware version installed.

1. Flash the NCP application to a radio board, start the host and connect to the NCP on the serial port. The following example is for a Linux system with a WSTK mainboard on the USB port enumerated as /dev/ttyACM0.

   ```
   ./Z3GatewayGPCombo -n 0 -p /dev/ttyACM0 -b 115200
   ```

2. Form a network using the Network Creator component CLI.

   ```
   plugin network-creator start 1
   ```

You should see something like this on the host terminal.

```
Z3GatewayGPCombo>plugin network-creator start 1
plugin network-creator start 1
NWK Creator: Form: 0x00
NWK Creator Security: Start: 0x00
NWK Creator: Form. Channel: 20. Status: 0x00
NWK Creator: Stop. Status: 0x00. State: 0x00
EMBER_NETWORK_UP 0x0000
Green Power Server Stack Status Callback status = 90
Green Power Translation Table Stack Status Callback status = 90
NWK Steering stack status 0x90
```

3. Once the host has formed a network, the `info` command displays essential network information.

   ```
   Z3GatewayGPCombo>MFG String: AppBuilder MFG Code: 0x1049
   node [(>)0807060504030201] chan [20] pwr [3]
   panID [0xDFEA] nodeID [0x0000] xpan [0x(>)7C3E9F7B090105B6]
   ezsp ver 0x08 stack type 0x02 stack ver. [7.0.0 GA build 0]
   nodeType [0x01]
   Security level [5]
   network state [02] Buffs: 250 / 255
   Ep cnt: 2
   ```

4. Join a Z3 Switch or a GPD switch to the above host. This will be used to demonstrate that other network devices are still part of the network after the NCP restore operation.
5. Print and record the child table, key table, proxy table, and sink table, to verify that the values are unchanged after the restore operation.
6. (optional) You might also do some operations to see the RX message confirming the network messages. Observe the network traffic on a sniffer.
7. Use the following CLI to back up the token data to a file *saved_token_data.nvm*.

   ```
   plugin trust-center-backup backup-tokens saved_token_data.nvm
   ```

   The following success message should be displayed. You can also see the file *saved_token_data.nvm* in the folder from where the Z3GatewayGPCombo host is running.

   ```
   plugin trust-center-backup backup-tokens saved_token_data.nvm
   Z3GatewayGPCombo>Opening file 'saved_token_data.nvm'
   SUCCESS Staus = 0
   ```

8. Reset the host using the command `reset`, which closes the connection to the NCP and exits the host application.
9. Restart the host and connect another NCP node that is mass-erased and flashed with the same NCP firmware version as the source NCP using the command above. The new NCP may be on another serial port.
10. Use the `info` command to show the new NCP's EUI64, which will be different to the one you recorded in the first NCP.
11. Restore the tokens from the saved file.

    ```
    plugin trust-center-backup restore-tokens saved_token_data.nvm
    ```

    A message like the following means the new NCP tokens are updated with the saved tokens from the *saved_token_data.nvm* file.

    ```
    Z3GatewayGPCombo>Opening file 'saved_token_data.nvm'
    SUCCESS Staus = 0
    ```

12. Reset the NCP with the following command, which initializes it with the restored token data that will restore the network operations.

```
plugin trust-center-backup reset-node
```

The result of the reset looks like this, ending with a network event of EMBER_NETWORK_UP.

```
Z3GatewayGPCombo>plugin trust-center-backup reset-node
plugin trust-center-backup reset-node
Z3GatewayGPCombo>ERROR: ezspErrorHandler 0x21
ERROR: ezspErrorHandler 0x28
ezsp ver 0x08 stack type 0x02 stack ver. [7.0.0 GA build 0]
Ezsp Config: set address table size to 0x0002:Success: set
Ezsp Config: set TC addr cache to 0x0002:Success: set
Ezsp Config: set MAC indirect TX timeout to 0x1E00:Success: set
Ezsp Config: set max hops to 0x001E:Success: set
Ezsp Config: set tx power mode to 0x8000:Success: set
Ezsp Config: set supported networks to 0x0001:Success: set
Ezsp Config: set stack profile to 0x0002:Success: set
Ezsp Config: set security level to 0x0005:Success: set
Ezsp Value : set end device keep alive support mode to 0x00000003:Success: set
Ezsp Policy: set binding modify to "allow for valid endpoints & clusters only":Success: set
Ezsp Policy: set message content in msgSent to "return":Success: set
Ezsp Value : set maximum incoming transfer size to 0x00000052:Success: set
Ezsp Value : set maximum outgoing transfer size to 0x00000052:Success: set
Ezsp Config: set binding table size to 0x0002:Success: set
Ezsp Config: set key table size to 0x0004:Success: set
Ezsp Config: set max end device children to 0x0006:Success: set
Ezsp Config: set aps unicast message count to 0x000A:Success: set
Ezsp Config: set broadcast table size to 0x0023:Error: set: 0x35
Ezsp Config: set neighbor table size to 0x0010:Success: set
NCP supports maxing out packet buffers
Ezsp Config: set packet buffers to 255
Ezsp Config: set end device poll timeout to 0x0008:Success: set
Ezsp Config: set zll group addresses to 0x0000:Success: set
Ezsp Config: set zll rssi threshold to 0xFFD8:Success: set
Ezsp Config: set transient key timeout to 0x012C:Success: set
Ezsp Endpoint 1 added, profile 0x0104, in clusters: 8, out clusters 17
Ezsp Endpoint 242 added, profile 0xA1E0, in clusters: 1, out clusters 1
SinkTable Init..
EMBER_NETWORK_UP 0x0000
Green Power Server Stack Status Callback status = 90
Green Power Translation Table Stack Status Callback status = 90
NWK Steering stack status 0x90
```

`info` shows the same details as those from the previous operational network.

13. Check the operation of the Z3Switch that was joined earlier and the already commissioned gpd-switch. They should start working without any rejoin.

7 Limitations

- This feature only applies to the NVM3 stack token group.
- The UART-based EZSP token data transfer relies on the EZSP transport mechanism. It does not have any data integrity as part of the host reading out the tokens and saving them to the file.
- The file format is evolving. The current custom format is as documented.
- The security and integrity of the file that stores the token data is not implemented.
- The backup subroutine on the host runs a continuous loop to collect the token data. This implies the following:
  - Backup cannot be a continuous process over time, as some of the tokens may get rewritten based on network activity between the start and end of the backup.
  - The one-time backup process is recommended for a time when no other network activity such as joining is happening, ideally just before the gateway is taken down for replacement. In this way the backup contains the latest operational token set before replacement.
  - Host application developers may choose to back up those tokens that do not change over a period of time to avoid the continuous loop and thereby to improve host performance. Use the above host subroutines as a starting point reference.
- A reset on the NCP is required after the restore because most of the operational RAM tables only get initialized from the token in the power-on reset path.
- This feature does not back up or restore any ephemeral RAM data such as neighbor table or incoming frame counters, as these are built back upon incoming packets after the network is restored.

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