PSOC™ Control C3M/P8, C3M/P7 MCUs provisioning specification

About this document

Scope and purpose

This application note provides essential information for enabling the provisioning process of Infineon's PSOC™ Control devices. It details the RAM application, Edge Protect Tools, and policy information.

Intended audience

This document is intended for users with advanced knowledge of the various components involved in the PSOC™ Control provisioning process.

This document presumes that you are already familiar with AN241270 – Getting started with PSOC™ Control C3M/P8, C3M/P7 MCUs on ModusToolbox™ software.

Table of contents

About this document . . . 1

Table of contents . . . 2

1 Introduction . . . 3

1.1 Provisioning overview . . . 3

1.2 Re-provisioning . . . 5

2 Certificates . . . 6

2.1 OEM certificate . . . 6

2.2 OEM CSR . . . 7

3 Policy . . . 10

3.1 Debug policy . . . 11

3.2 Boot policy . . . 13

3.3 DFU Policy . . . 14

3.4 Re-provisioning policy . . . 14

3.5 OEM keys . . . 15

4 Edge Protect Tools . . . 16

4.1 Overview . . . 16

4.2 Device communication . . . 17

4.3 Edge Protect Tools package for PSOC™ Control MCU . . . 17

4.3.1 Packaged assets . . . 19

5 DLM (Downloadable Module) package . . . 20

5.1 RAM application (RAM_APP) . . . 20

5.2 DLM (Downloadable Module) package . . . 20

5.3 Creating a DLM package . . . 23

5.4 Download and launch of the DLM package . . . 23

5.4.1 Launch flow: BootROM DFU . . . 24

A Appendix . . . 25

A.1 RAM application (DLM) status codes . . . 25

A.2 Edge Protect Tools supported commands . . . 27

A.3 CBOR and COSE format . . . 28

References . . . 29

Glossary . . . 30

Revision history . . . 33

Trademarks . . . 34

Disclaimer . . . 35

1 Introduction

Provisioning is the process of injecting authorized keys and/or policy configuration into a device. Original equipment manufacturers (OEMs) use the provisioning process to load OEM Root of Trust (RoT) keys and OEM policies into the PSOC™ Control device. The provisioning of OEM keys, certificates, and policies transfers device ownership from Infineon to the OEM.

PSOC™ Control devices can be provisioned using Infineon's Edge Protect Tools package, which creates and downloads a provisioning packet into the device. This provisioning packet includes a RAM application capable of performing the provisioning, along with input parameters for the RAM application, such as the policy file and keys.

This document provides detailed information about the various types of certificates, policies, RAM application, and downloadable module (DLM) packages, all of which are included in or generated by Edge Protect Tools.

This document is not a programming specification. Refer to AN238998 - PSOC™ Control C3 MCU programming specification.

1.1 Provisioning overview

The provisioning process enables the OEM to configure several boots and debug related policy configurations. As a primary step, the OEM is required to ‘own’ the device before making any changes to the Infineon default policies on the device. To take ownership of the device, the OEM needs to generate its own keys and certificate, and provision the device with these, as shown in Step 1 of Figure 1. This step can be performed whenever the OEM needs to change the root of trust (RoT) keys on the device, even while the device is in the development lifecycle stage (LCS). After the transfer of ownership is completed, the OEM must then use the OEM_ROT_KEY to perform further provisioning of the device for the purpose of policy modification as shown in Step 2 of Figure 1. There are no restrictions on the number of times the device can be provisioned in the development lifecycle stage. Therefore, provisioning can be utilized to carry out the following operations on the device:

- Replace keys and certificates
- Replace the OEM policy file

The device can then be provisioned 'one' time to transition into the production lifecycle by using the production certificate. After the device is moved into the production lifecycle stage, it cannot be returned to the development LCS and the OEM_ROT_KEY can no longer be changed through provisioning. In the production lifecycle stage (LCS), you only have a single opportunity to perform key revocation, which allows you to replace the existing OEM ROT keys with new ones. This operation is irreversible and can only be performed once.

1 Introduction

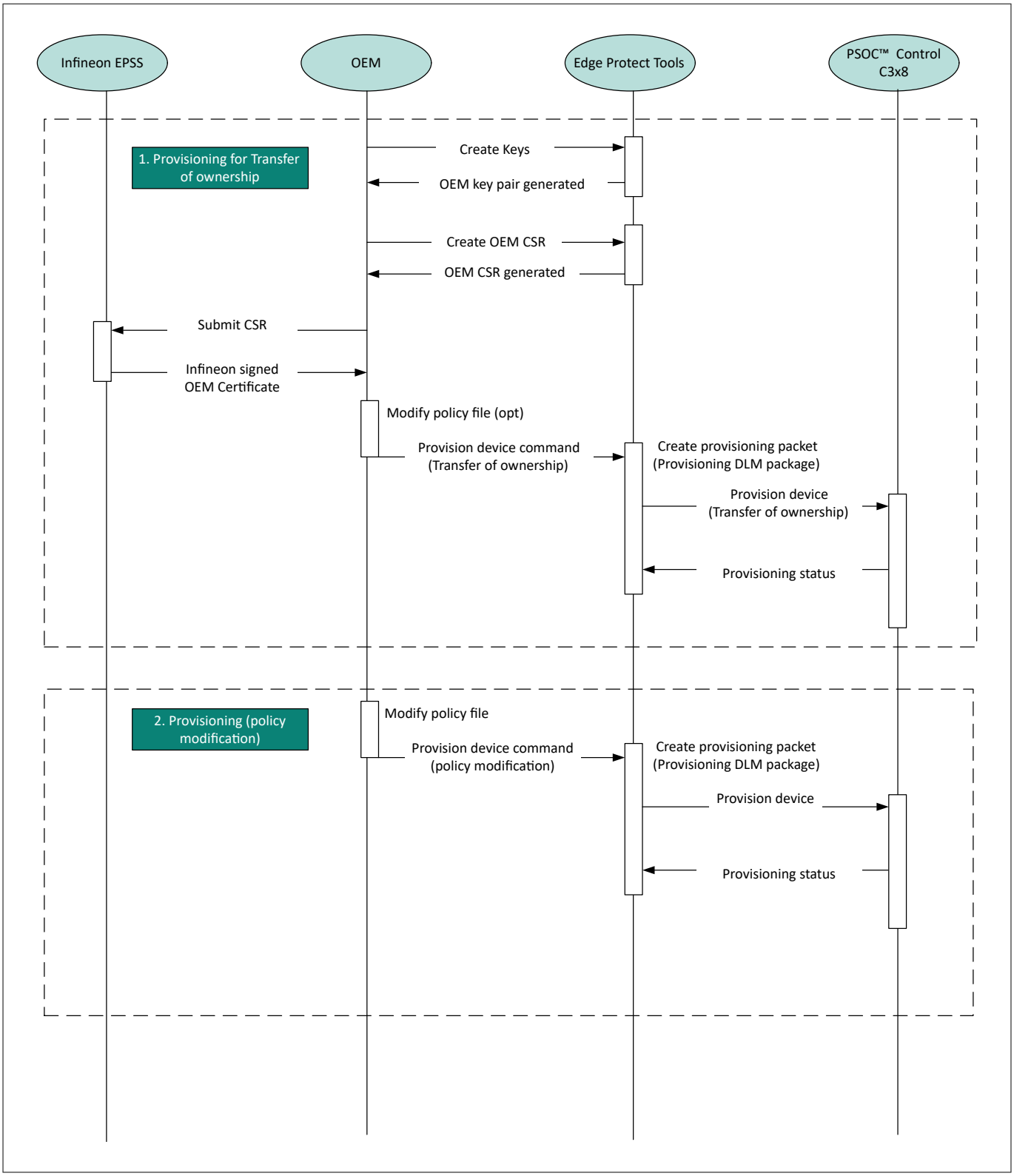


Figure 1 Provisioning overview

***Note**: To program an eFuse, VDDIO0 must be at 2.5 V ±10%. See the datasheet for more details.*

1.2 Re-provisioning

While in the production lifecycle stage, the device can only be re-provisioned if the re-provisioning policy for the specific policy field has been configured accordingly. For more details, see Re-provisioning policy.

2 Certificates

A digital certificate or a public-key certificate can be used by any entity to establish the ownership of a public key. Certificates contain a public key, information about the public key, identity of its owner, and the signature of the certificate authority (CA) that verifies the content of the certificate. A certificate signing request (CSR) is a request sent from the entity that owns the public key to the certificate authority to verify the information provided in the CSR and generate the certificate.

There are different kinds of certificates and CSR that are used in the context of PSC3M/P8, PSC3M/P7 devices.

Certificate	Purpose
OEM certificate	This contains the OEM_ROT_PUBLIC_KEY, which is signed by the Infineon CA and provisioned by the OEM to transfer the ownership of the device. See OEM certificate
OEM CSR	This is a self-signed certificate signing request sent from the OEM to the Infineon CA to generate the OEM certificate. See OEM CSR

For PSC3M/P8, PSC3M/P7 devices, the certificates use the CBOR/COSE format. The device ownership is transferred to the OEM when the device is provisioned with the OEM certificate. This certificate is signed by Infineon through Edge Protect Signing Service and is verified by the device after provisioning. This process ensures that only Infineon approved OEM certificate can be provisioned on the device. OEM certificate contains the OEM root of trust public Key. The OEM root of trust private key must be protected by the OEM and used during provisioning of the device to sign the OEM policy, and also to sign the user application images before programming, in case secure boot is enabled.

2.1 OEM certificate

The OEM certificate contains the OEM_ROT_PUBLIC_KEY and enables the OEM to take ownership of the device. This certificate must be signed by the IFX_ROT_KEY, and generated by Infineon's Edge Protect Signing Service (EPSS) based on the CSR uploaded by the OEM. For more detailed instructions on generating the OEM CSR and OEM certificate, see AN242836 – Getting started with PSOC™ Control C3M/P8, C3M/P7 C3 security.

The OEM certificate can be of two types:

- **Development certificate:** The development OEM certificate contains the OEM_DEV_PUBLIC_KEY that is used by the OEM during the development LCS of the device. In the development lifecycle, the device can be provisioned multiple times to change the policy and/or keys, and can also be restored to Infineon default policy settings. When the development certificate is used, the keys are stored in a mutable memory and the lifecycle of the device remains unchanged. Development OEM keys do not need to be secured and can be replaced at any time by the OEM
- **Production certificate:** The production OEM certificate is used to transition the device from the development to production LCS. This certificate must contain the production OEM_ROT_PUBLIC_KEY that will be stored in the immutable memory region. Two sets of the OEM_ROT_KEYs are supported to enable key revocation in the event of a security breach. The production OEM ROT keys, used for production devices, must be securely stored and used by the OEM. These keys cannot be replaced by provisioning

Table 1 OEM certificate fields

Item	Field	Example value	Data type	Description
1	VERSION	1	String	Certificate version
2	TEMPLATE_TYPE	OEM_CERT	String	Certificate template
3	CERTIFICATE_NAME	OEM_CSR	String	Certificate name

(table continues...)

Table 1 (continued) OEM certificate fields

Item	Field	Example value	Data type	Description
4	TYPE	DEVELOPMENT	String	Certificate type. Applicable values: DEVELOPMENT, PRODUCTION
5	ISSUER	Dummy Corp.	String	Certificate issuer
6	ID	ABCDEF	HEX	Certificate (unique) ID
7	DATE	2023-09-19 15:24:47	String	The date of creation of the certificate
8	ALGORITHM	ES521	String	Key algorithm. Applicable values: ES256, ES384, ES521
9	SIGNER_ID	Dummy_OEM	String	Certificate signer ID
10	PUBLIC_KEY_0	049A0BFAC2727E7...	HEX	OEM_ROT public key 0
11	PUBLIC_KEY_1	049A0BFAC2727E7...	HEX	OEM_ROT public key 1
12	OEM	Dummy_OEM	String	OEM name
12	PROJECT	Dummy_Project	String	Project name
13	PROJECT_NUMBER	123	String	Project number

The OEM certificate generated by Infineon's EPSS will be in the COSE format. A JSON representation of the same is shown in the following code block:

OEM certificate in JSON format

```
{
"VERSION": "1",
"TEMPLATE_TYPE": "OEM_CERT",
"CERTIFICATE_NAME": "OEM_CSR",
"TYPE": "DEVELOPMENT",
"ISSUER": "Dummy Corp.",
"ID":h'ABCDEF',
"DATE":"2023-09-19 15:24:47",
"ALGORITHM": "ES521",
"SIGNER_ID": "555",
"PUBLIC_KEY_0":
h'04C55A85F02FDF5D2256397C7C154DAF2AECDBB56403F4ED279D47AACE1D9270063BFB09AB8CFFE8DC2172DC354D37
7495ECFE2FE0C8A55E3B8F521F79FF874CBB',
"PUBLIC_KEY_1":
h'04504D54E877726B856CF59940CCE458DA85F805FE1AC4CDEF692B598B3CE11E772C841F2F6C09998090E5D85C468B
BE85431FE4EFBA6AED8D9777E2E46A92C84D',
```

2.2 OEM CSR

The OEM certificate signing request (CSR) must be generated by the OEM with the OEM_ROT_PUBLIC_KEY and other relevant information. The CSR is signed by the OEM_ROT_PRIVATE_KEY. OEM CSR can be generated using

the Edge Protect Tools. The OEM CSR must be submitted to Infineon's Edge Protect Signing Service (EPSS) to generate the OEM certificate.

Table 2 OEM CSR fields

Item	Field	Example value	Data type	Description
1	VERSION	1	String	Certificate version
2	TEMPLATE_TYPE	OEM_CSR	String	Certificate type
3	CERTIFICATE_NAME	–	–	Not applicable
4	OEM	Dummy_OEM	String	OEM name
5	PROJECT	Dummy_Project	String	Project name
6	PROJECT_NUMBER	123	String	Project number
7	ISSUER	Dummy Corp.	String	Certificate issuer
8	SIGNER_ID	555	String	Signer ID
9	ALGORITHM	ES521	String	Key algorithm. Applicable values: ES256, ES384, ES521
10	PUBLIC_KEY_0	"keys/ oem_public_key_0.pem"	String	Path to the ECDSA P-256 public key or a 65-byte hex string
11	PUBLIC_KEY_1	"keys/ oem_public_key_1.pem"	String	Path to the ECDSA P-256 public key or a 65-byte hex string
12	TYPE	DEVELOPMENT	String	Certificate type. Applicable values: DEVELOPMENT, PRODUCTION
13	DATE	2023-09-19 15:24:47	String	The date of creation of the certificate in the format: YYYY-MM-DD hh:mm:ss. If not provided, the system time will be used
14	ID	ABCDEF	HEX	Certificate (unique) ID

2 Certificates

The OEM CSR generated by the Edge Protect Tools will be in the CBOR Object Signing and Encryption (COSE) format. A JSON representation of the same is shown in the following code block:

```
{
"VERSION": "1",
"TEMPLATE_TYPE": "OEM_CSR",
"CERTIFICATE_NAME": "",
"OEM": "Dummy_OEM",
"PROJECT": "Dummy_Project",
"PROJECT_NUMBER": "123",
"ISSUER": "Dummy Corp.",
"SIGNER_ID": "555",
"ALGORITHM": "ES521",
"PUBLIC_KEY_0": "keys/oem_public_key_0.pem",
"PUBLIC_KEY_1": "keys/oem_public_key_1.pem",
"TYPE": "DEVELOPMENT",
"DATE": "2023-09-19 15:24:47",
"ID": h'ABCDEF'
}
```

3 Policy

The boot configuration of PSOC™ Control devices is dictated by a policy file. All policy configurations are applied during device boot up by the Infineon BootROM (ROM_BOOT + FLASH_BOOT). The policy file is a JSON configuration file that defines the settings/configurations i.e., input parameters for device provisioning. Refer to AN242836 – Getting started with PSOC™ Control C3M/P8, C3M/P7 MCUs security - Policy file for list of input parameters that can be provisioned to device.

In the Infineon factory, the device is provisioned with a set of default policy configurations. This default policy is packaged as a part of the Edge Protect Tools package and contains different sections that can be broadly classified into the three OEM policy sections:

- Debug policy
- Boot policy
- DFU policy

The policy file also includes a Keys section where the user OEM_ROT_KEY and device parameters are defined. The following figure shows what different input parameters are part of the policy file.

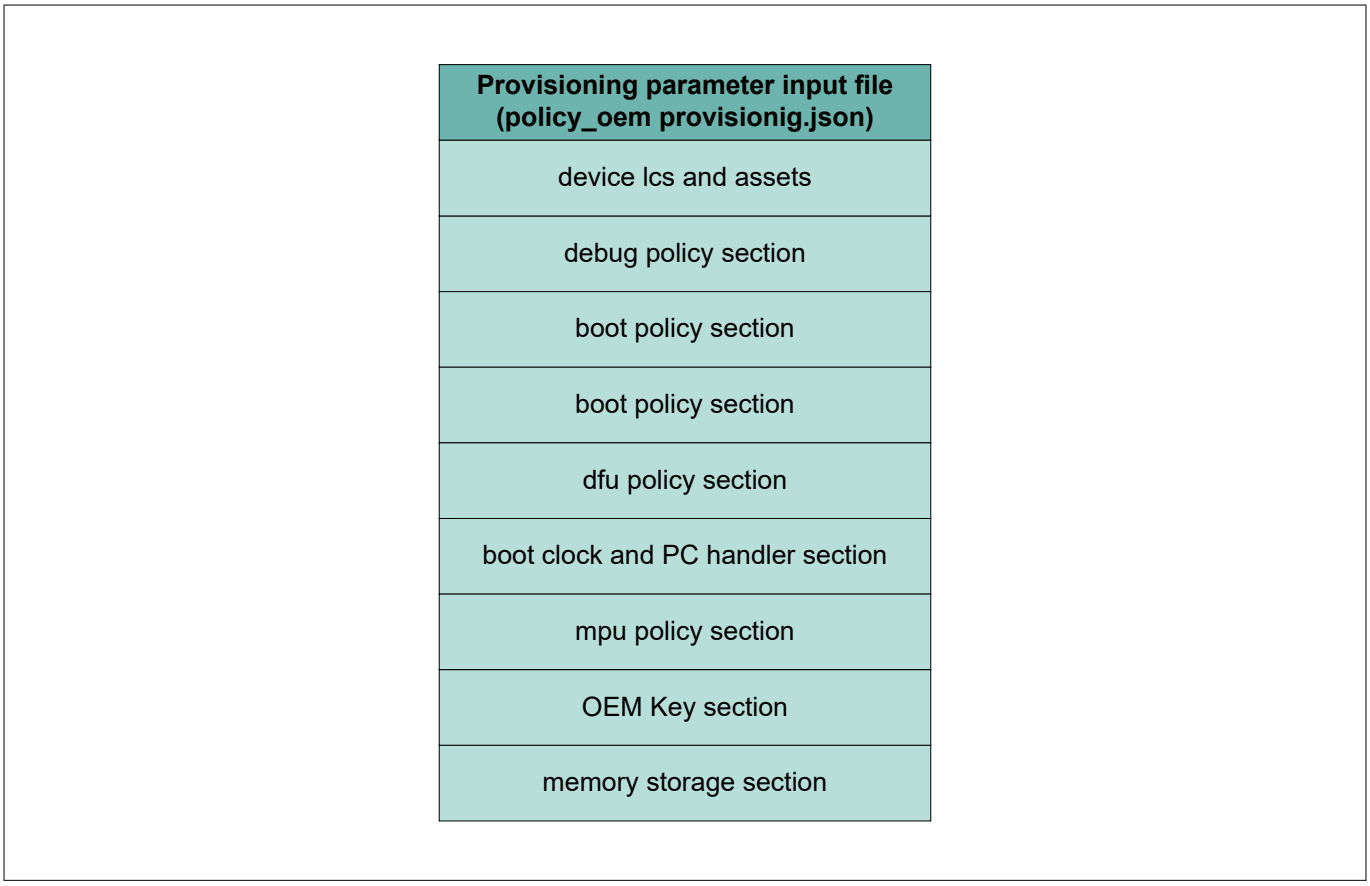


Figure 2 OEM policy file layout

Once the Edge Protect Tools package is initialized on the host PC, see the JSON policy in the `<application-directory>/policy/policy_oem_provisioning.json` file. This file can be edited using any text editor to customize the policy.

The policy file is converted into a CBOR packet that contains all provisioning data by Edge Protect Tools. This packet is then signed accordingly to the COSE format and the final packet is sent to the device during provisioning by Edge Protect Tools. The RAM application on the device then receives the packet, verifies the packet's signature, and processes the CBOR payload to get data for the provisioning. Typically, the RAM application processes provisioning data to convert it to the asset format stored on the device.

3.1 Debug policy

The debug policy configuration is used by the OEM to control access to debug ports through two mechanisms:

- Direct configuration
  - Enable/disable debug ports
  - Infineon's default policy enables all debug configurations
- Authenticated debug
  - Debug ports can be temporarily enabled using a debug token
  - Token must be cryptographically signed with authorized OEM RoT keys
  - Provides secure, controlled debug access when ports are disabled

***Note**: While Infineon's default policy enables all debug configurations, OEMs can implement authenticated debug for enhanced security by requiring valid debug tokens to access disabled ports.*

The authenticated debug works only when the debug field is set to “Permanently Disable” in the policy file. When set to “Disable”, the debug ports are permanently disabled and cannot be enabled even with debug token.

The default configurations in a policy file are shown in the following code block:

3 Policy

Debug default policy configuration

```
"debug": {
      "cpu": {
        "ap_cm33": {
          "description": "Configures the CM33 debug access port availability in the SECURE and
NORMAL_PROVISIONED LCS",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "listen_window": {
          "description": "Configures the listen window time to acquire the CM33 debug access
port in the SECURE LCS",
          "applicable_conf": "100 ms, 20 ms, 1 ms, 0 ms",
          "value": "100 ms"
        },
        "cm33_dbg": {
          "description": "Configures the invasive debug of CM33 core",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "cm33_nid": {
          "description": "Configures the non-invasive debug of CM33 core",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "cm33_spid": {
          "description": "Configures the secure privileged invasive debug of CM33 core",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "cm33_spnid": {
          "description": "Configures the secure privileged non-invasive debug of CM33 core",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "cm33_secure": {
          "description": "Enables the CM33 secure AP interface",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        }
      },
      "system": {
        "ap": {
          "description": "Configures the System debug access port availability in the SECURE
and NORMAL_PROVISIONED LCS",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
        },
        "secure": {
          "description": "Enables the system secure AP interface",
          "applicable_conf": "Enable, Disable, Permanently Disable",
          "value": "Enable"
```

```
        },
        "mmio_restrictions": {
          "description": "MMIO restrictions for System AP",
          "applicable_conf": "No restrictions, SRSS_GENERAL2 only, No access",
          "value": "No restrictions"
        },
        "debug_pins": {
          "description": "Configures debug pins and interface",
          "comment_1": "0000 - 2 pins (TMS, TCLK), JTAG is disabled",
          "comment_2": "0011 - 2 pins (TMS, TCLK), JTAG is enabled",
          "comment_3": "0101 - 4 pins (TMS, TCLK, TDO, TDI)",
          "comment_4": "0110 - 5 pins (TMS, TCLK, TDO, TDI, TRSTN)",
          "comment_5": "1001 - SWJ pins disabled, JTAG is enabled",
          "comment_6": "1010 - SWJ pins disabled, JTAG is disabled",
          "applicable_conf": "0000, 0011, 0101, 0110, 1001, 1010",
          "value": "0000"
        }
      }
    },
```

3.2 Boot policy

Table 3 Boot policy

Field	Description
boot_cfg_id	Selects the boot configuration mode. SIMPLE_APP is a basic single-image boot, while SECURE_APP is a single image boot with secure boot enabled in the BootROM to validate the application. Similarly, the DUAL_BANK_SIMPLE_APP, DUAL_BANK_SECURE_APP, and PROT_FW provide progressively stronger security and update capabilities.
boot_dual_bank_enable	When true, enables flash dual-bank mode, allowing a new firmware image to be staged in one bank while the current firmware runs in the other — enabling atomic, fail-safe updates.
boot_bank_ctr_offset	The offset within each flash bank where the 32-bit bank-swap counter is stored. Used by the bootloader to determine which bank holds the most recently updated firmware
boot_hash_locking	When true, uses a faster hash comparison for boot authentication instead of full ECDSA signature verification. Recommended only where boot time is critical and the hash is stored securely.
boot_auth_images	Controls which application images are authenticated at boot. ALL_MOVE authenticates all images and also moves them from Shared Secondary Slot (SSS) into execution memory after a successful DFU update.
boot_auth_signature	The elliptic-curve signature algorithm used for image authentication. ES512 (ECDSA with P-521) provides the highest security level.
boot_app_layout	Defines up to 5 application memory slots (address + size). Secure images use the 0x32xxxxxx address range; non-secure images use 0x22xxxxxx. Unused slots are set to 0x00000000.
boot_od_clk	The CM33 core clock frequency during boot. Higher frequencies reduce boot time but increase power consumption.

3.3 DFU Policy

DFU (Device Firmware Update) is a mechanism that allows the firmware on a device to be updated over a wired communication interface (e.g., UART, SPI, I2C, etc.) without requiring physical access to debug/programming pins. It enables field upgrades, bug fixes, and feature additions after deployment.

To enable it, change "value": false to "value": true:

DFU policy default configurations

```
"dfu": {
  "dfu_enable": {
    "description": "Indicates whether Device Firmware Update is enabled",
    "value": true
  }
}
"mmio_restrictions": {
"description": "MMIO restrictions for System AP",
"applicable_conf": "No restrictions, SRSS_GENERAL2 only, No access",
"value": "No restrictions"
},
"debug_pins": {
"description": "Configures    debug    pins and interface",
"comment_1":    "0000    - 2 pins    (TMS,    TCLK),    JTAG is disabled",
"comment_2":    "0011    - 2 pins    (TMS,    TCLK),    JTAG is enabled",
"comment_3":    "0101    - 4 pins    (TMS,    TCLK,    TDO, TDI)",
"comment_4":    "0110    - 5 pins    (TMS,    TCLK,    TDO, TDI, TRSTN)",
"comment_5":    "1001    - SWJ pins disabled,    JTAG is enabled",
"comment_6":    "1010    - SWJ pins disabled,    JTAG is disabled",
"applicable_conf": "0000, 0011, 0101, 0110, 1001, 1010",
"value": "0000"
}
}
}
```

3.4 Re-provisioning policy

When the PSC3M/P8, PSC3M/P7 devices transition to the production lifecycle stage, only a single key revocation is permitted through the re-provisioning process. During initial provisioning, the OEM provisions two OEM RoT keys — this ensures that if one key is compromised, the other remains available for OEM application validation. By default, BootROM uses **OEM_ROT_KEY_0** as the primary key for OEM application validation.

If **OEM_ROT_KEY_0** is compromised, the OEM can re-provision the policy to designate **OEM_ROT_KEY_1** as the primary key, replacing OEM_ROT_KEY_0 for BootROM-based OEM application validation. Similarly, for the RAM application, **IFX_ROT_KEY_1** will be used for authentication in place of IFX_ROT_KEY_0.

To re-provision a specific asset, set its corresponding field value to true in the policy. The default policy configurations are shown in the following code block:

Re-provisioning policy default configurations

```
{
  "policy": {
    "platform": "psoc_c3",
    "version": 1.0,
    "type": "reprovisioning"
  },
  "revoke": {
    "description": "Key revocation",
    "applicable_conf": "IFX_ROT_KEY_0, OEM_ROT_KEY_0, MPN_ROT_KEY_0, PROT_FW_ROT_KEY_0",
    "value": "OEM_ROT_KEY_0"
  },
  "public_key": {
    "description": "Path to the public key used to verify this policy signature",
    "value": "../keys/oem_rot_pub_key_0.pem"
  }
}
```

3.5 OEM keys

The OEM can specify their own RoT keys, referred to as 'user keys,' in the OEM policy file. These keys are stored in secure flash memory during the provisioning process. These keys are used for signing OEM images for secured boot and signing the debug tokens used for authenticated debug.

The default policy configurations are shown in the following code block:

User keys default configurations

```
  "keys": {
       "oem_rot_public_key_0": {
         "description": "Path to the OEM RoT public key 0",
         "value": "../keys/oem_rot_pub_key_0.pem"
       },
       "oem_rot_public_key_1": {
         "description": "Path to the OEM RoT public key 1",
         "value": "../keys/oem_rot_pub_key_1.pem"
       },
       "oem_debug_public_key": {
         "description": "Path to the OEM debug public key",
         "value": "../keys/oem_debug_pub_key.pem"
       },
       "oem_dev_public_key": {
         "description": "Path to the OEM RoT dev public key",
         "value": "../keys/oem_dev_pub_key.pem"
       }
     }
```

4 Edge Protect Tools

4.1 Overview

Edge Protect Tools is a Python-based package that provides a command-line interface for interacting with the MCUs. It supports various functionalities, including provisioning, device integrity checks, RMA transition, and more. This tool is implemented in the Python programming language and uses several third-party Python modules to manage configuration files (`JSON`), object files (`ELF`), and perform cryptographic operations.

The tool can be accessed using any shell-based command-line interface. However, Infineon recommends using modus-shell, which is included in the ModusToolbox™ installation package, for seamless interaction with the Edge Protect Tools package.

When executing commands that require MCU interaction, such as provisioning, the tool prepares the necessary provisioning packets and communicates with the device using programming tools. It supports tools like OpenOCD via the SWD interface or the DFU Host Tool via serial interfaces, such as I^2C, SPI, or UART.

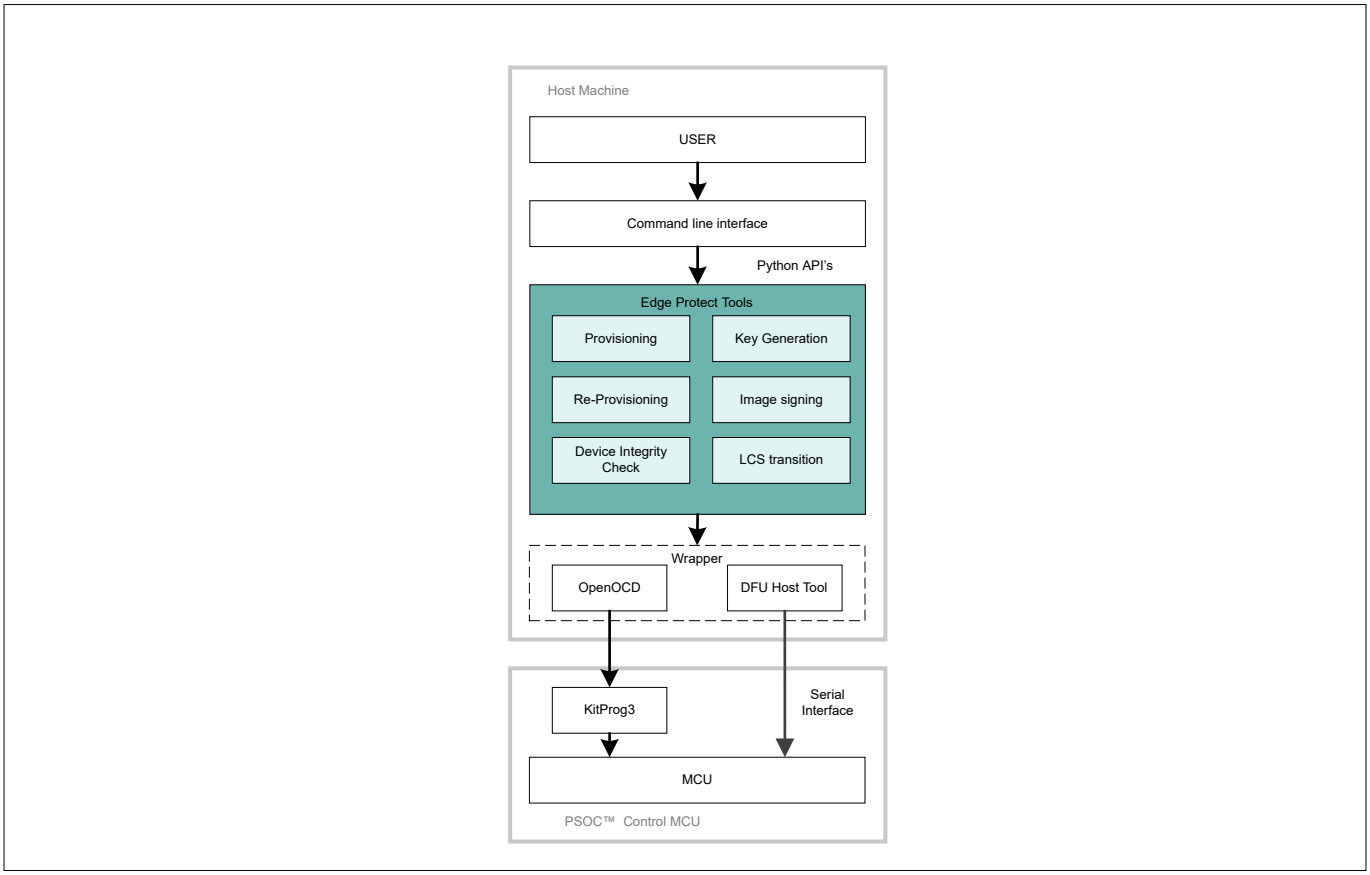


Figure 3 Edge Protect Tools overview

The operations supported by the tool include:

- **Provisioning**: Injecting keys and policies into the device
- **Re-provisioning**: Modifying specific configurations after provisioning
- **Key pair generation**: Creating key pairs for provisioning and image signing
- **Certificate generation**: Generating certificates for debugging and RMA transition
- **Signing images**: Signing images with the OEM key

For the complete list of commands supported by Edge Protect Tools package, see the Appendix A section.

4.2 Device communication

The Edge Protect Tools package supports communication with the device through programming tools such as OpenOCD via the SWD interface or the DFU Host Tool via serial interfaces like UART. The following table mentions the different interface support for various MCUs.

Table 4 Device communication interface support

Device	SWD	UART
PSC3M/P8, PSC3M/P7	Yes	Yes

Instead of directly invoking programming tool commands, the Edge Protect Tools package uses a wrapper mechanism to interface with these tools. This design is extendable, enabling potential integration with other programming tools.

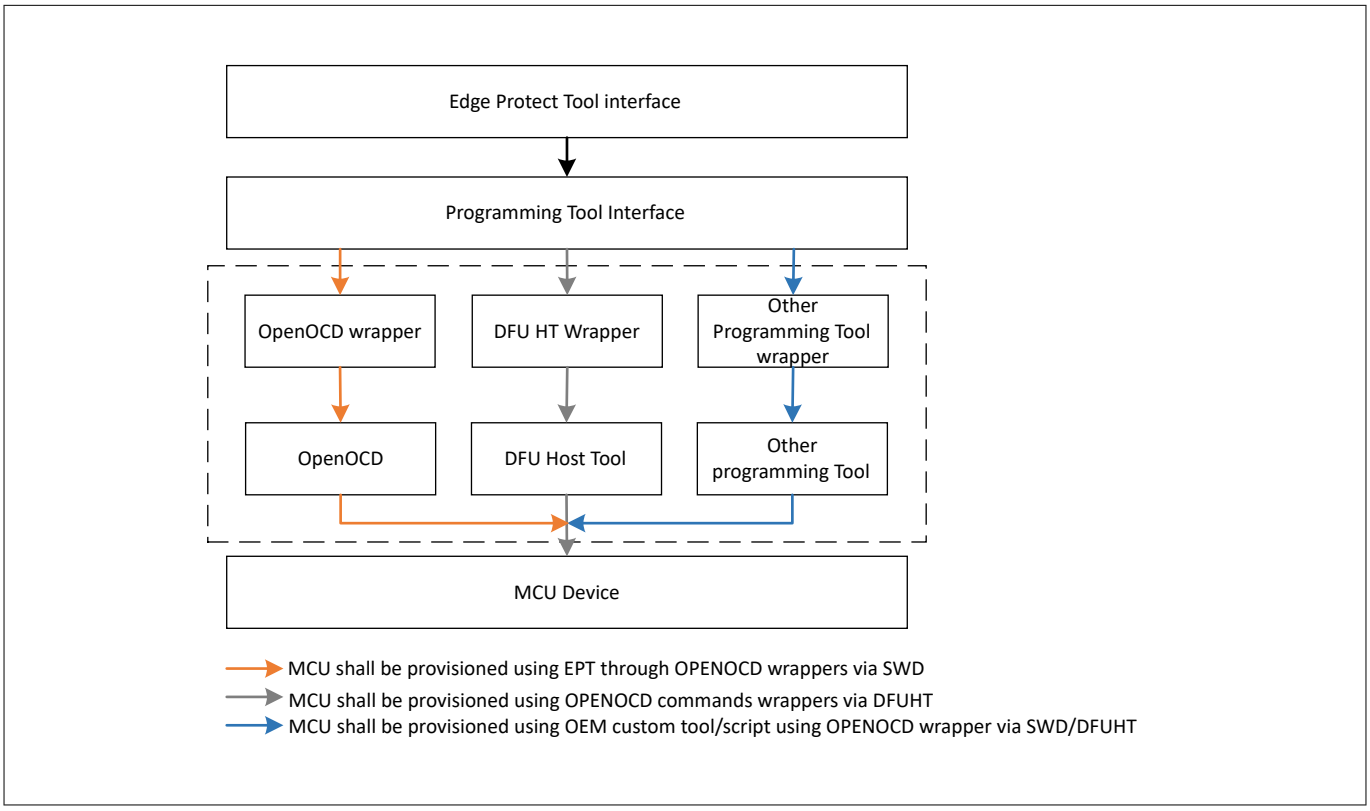


Figure 4 Edge Protect Tools device communication

4.3 Edge Protect Tools package for PSOC™ Control MCU

Edge Protect Tools is included as part of the Edge Protect suite for the PSC3M/P8, PSC3M/P7 devices.

4 Edge Protect Tools

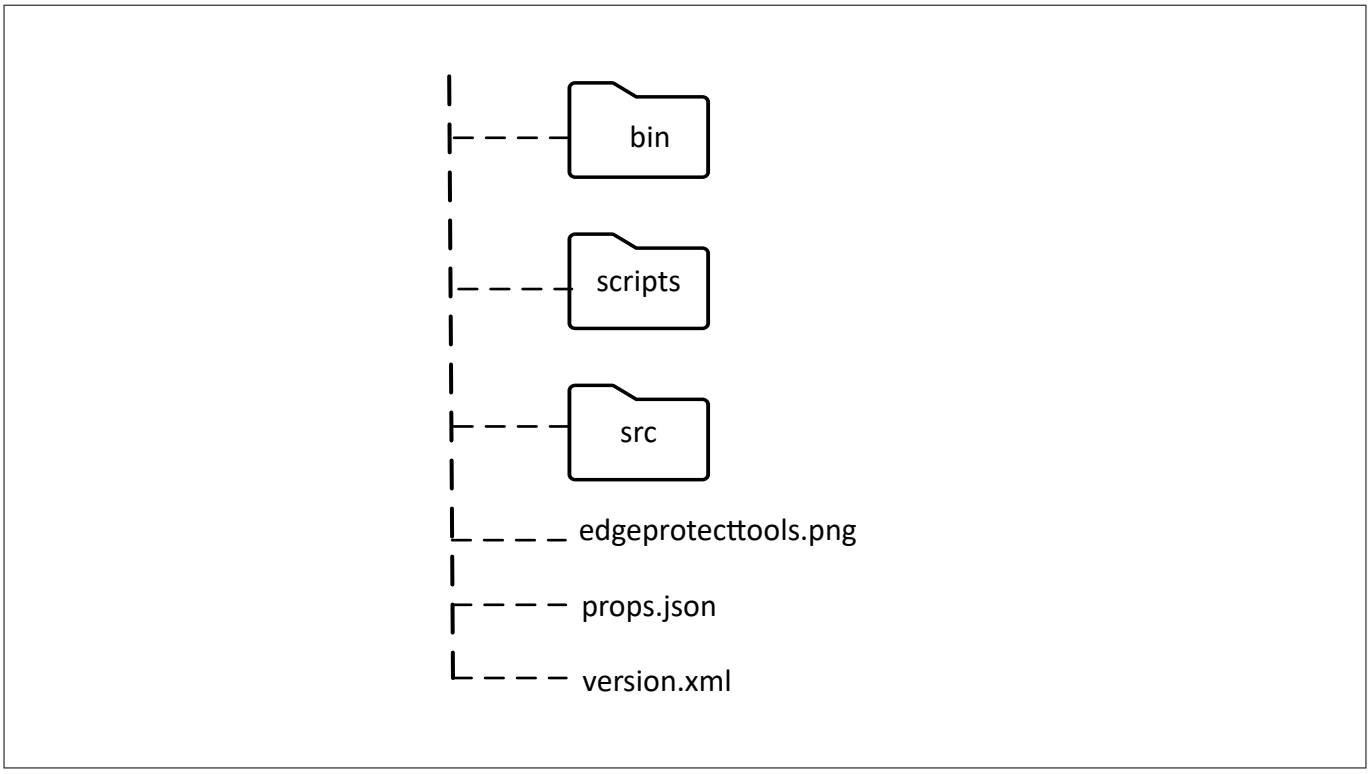


Figure 5 Edge Protect Tools in ModusToolbox™ Edge Protect Security Suite

The `./bin` directory contains a pre-compiled executable file for Windows, Linux, and macOS, which you can use directly. Add the executable path to your system's environment variable path:

```
<ModusToolbox-Edge-Protect-Security-Suite-install- path>/tools/edgeprotecttools/bin
```

Before starting, Edge Protect Tools must be initialized for a specific target, such as the 'psoc_c3x8' for the PSC3M/P8, PSC3M/P7 devices.

To initialize, navigate to your working directory and execute the following command in a shell CLI:

```
edgeprotecttools -t psoc_c3x8 init
```

After initialization, the following folder structure as shown in Figure 6 will be created in your working directory.

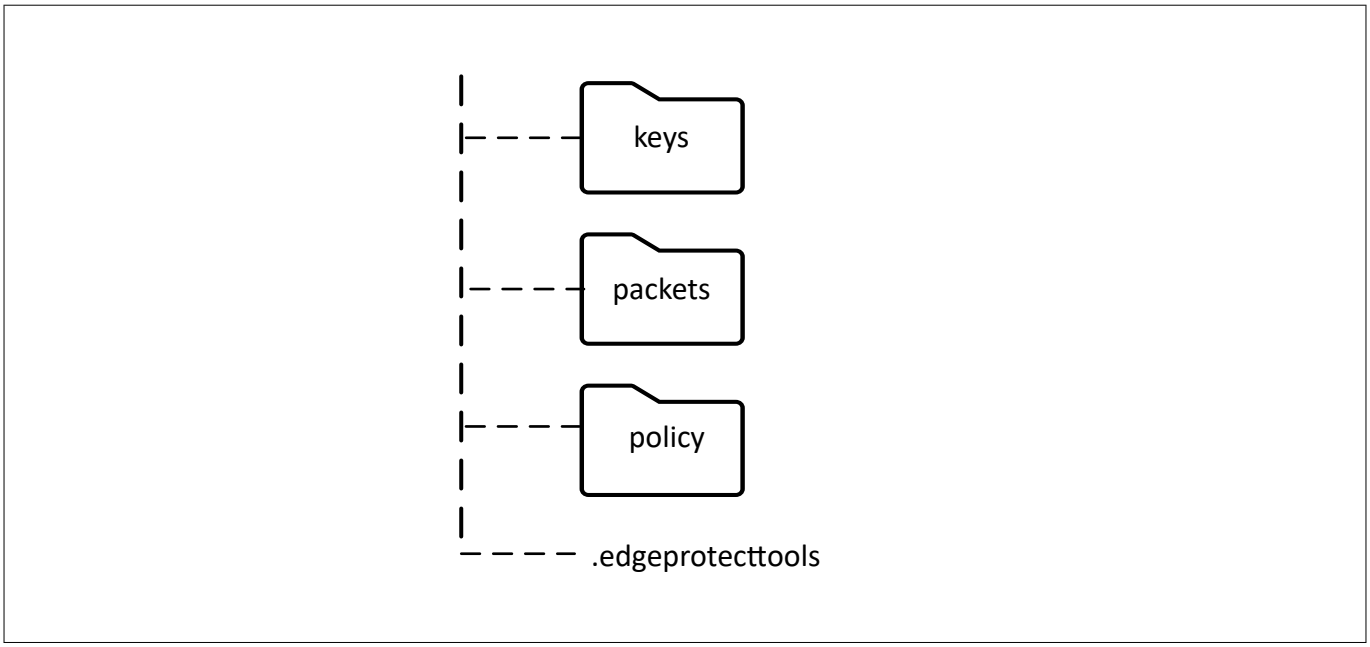


Figure 6 Edge Protect Tools project directory

Setting up the OpenOCD path:

To set up the OpenOCD path for use with the tools, execute the following command:

```
edgeprotecttools set-ocd --name openocd --path <openocd_path>
```

***Note**: Here, `<openocd_path>` is the path to the OpenOCD tool on your system. Typically, it will be located in the default installation directory: C:/Infineon/Tools/ModusToolboxProgtools-1.3/openocd.*

***Notes**:*

1. *On Windows, use a Windows-style path (that is, not a path like `/cygdrive/c/`). Additionally, use forward slashes (/). For example, `C:/Users/XYZ/ModusToolbox/`*
2. *After installing Edge Protect Tools on Linux or macOS, follow any additional instructions provided by `pip` during the installation process*

4.3.1 Packaged assets

The following assets are included in the Edge Protect Tools package and can be seen in the `<working-dir>` where the Edge Protect Tools is initialised

- **OEM policy**: The OEM policy file (`policy_oem_provisioning.json`) is used to configure device policy settings. This file is located in the `<working-dir>/policy` folder
- **RAM application**: Infineon-signed RAM application (DLM) binaries are available in the `<working-dir>/packets/apps` folder. RAM applications are owned by Infineon and cannot be customized by users
- **JSON template files**: JSON template files used by Edge Protect Tools are located in the `<working-dir>/packets` folder

5 DLM (Downloadable Module) package

A DLM package is a structured binary file that contains:

- Infineon signed RAM_APP
- Input parameters for device provisioning
- Signatures

5.1 RAM application (RAM_APP)

RAM applications are executable code loaded into the SRAM and executed in the PC0 (Protection Context 0) (AN242836 - Getting started with PSOC™ Control C3M/P8, C3M/P7 MCUs security). Certain device features, such as provisioning and re-provisioning, are implemented using RAM applications. These applications are developed by Infineon, signed with Infineon keys, and verified and launched by the device's BootROM. It is important to note that RAM applications cannot be developed by OEMs.

RAM applications are packaged in the MCUboot image format. This format adds application metadata to verify image integrity and authenticity, along with custom metadata in the tag-length-value (TLV) format.

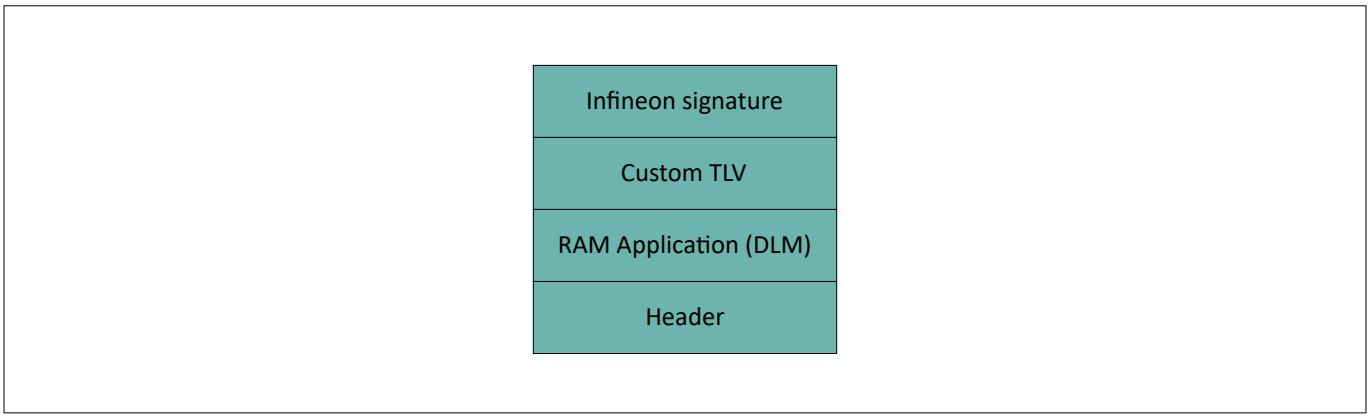


Figure 7 Signed RAM application

5.2 DLM (Downloadable Module) package

To provision a device, all the required input parameters required are set in the JSON files and this data is sent along with the RAM APP to the device to be provisioned. This whole package is called DLM package. During provisioning process, the edgeprotecttools transfer the package to the device SRAM, the device then launches the RAM_APP and the RAM_APP will provision the input parameters to the device SFlash. This package consists of the following components:

- Infineon signed RAM application
- DLM input parameters

The DLM packages are signed only for a few policies, such as reprovisioning, RMA, and DFU. For default device provisioning, the DLM signing is not required. The table below mentions which policies require signing.

Table 5 Policies requiring signatures

App name	Input parameters signed		DLM package signed	
	NORMAL_PROVISIONED LCS	SECURE LCS	NORMAL_PROVISIONED LCS	SECURE LCS
cyapp_prov_oem	NA	NA	NA	NA

(table continues...)

Table 5 (continued) Policies requiring signatures

App name	Input parameters signed		DLM package signed	
	NORMAL_PROVISIONED LCS	SECURE LCS	NORMAL_PROVISIONED LCS	SECURE LCS
cyapp_reprovisioning	Yes	Yes	Yes	Yes
cyapp_to_rma	NA	Yes	NA	Yes
cyapp_prot_fw_dfu	NA	NA	NA	Yes
cyapp_prot_fw_policy	NA	NA	NA	NA

***Note**: In the above table, the 'Yes' fields required OEM_ROT_KEYS for signing.*

To sign the DLM package, it is wrapped in the MCUboot format with a header and signature. The package is signed using the `OEM_ROT_PRIVATE_KEY`. The ROM boot always expects the DLM package to follow this format for processing; however, the package signature is verified only in the production lifecycle stage (SECURE LCS).

The OEM signature provides the OEM with control over which DLM is executed on the device in the production LCS, as well as its associated input parameters. To modify the input parameters within the package, the package must be re-signed using the `OEM_ROT_PRIVATE_KEY`.

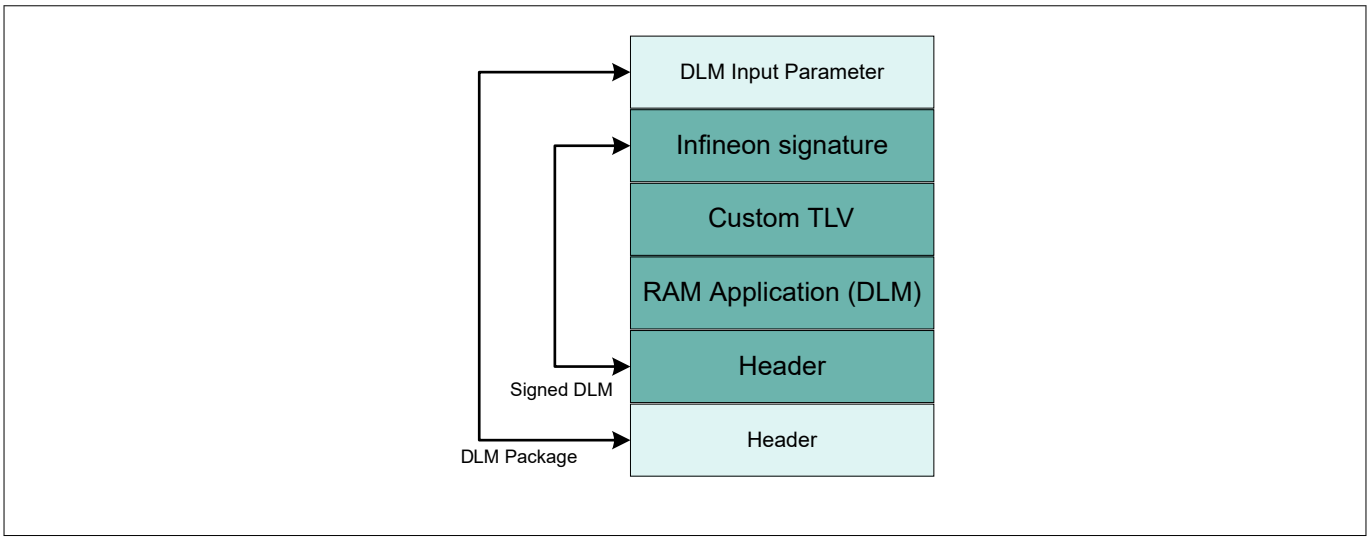


Figure 8 DLM package format

The placement of the DLM package in the device's SRAM memory is demonstrated in Figure 9. The DLM package is uploaded into the internal SRAM using either the debugger (SYS-AP) or an application firmware running on the device. Upon request, the BootROM validates the DLM package and post a successful validation, and it launches the RAM application in SRAM.

***Note**: In case of OEM provisioning, the DLM need not be signed and the DLM package will not contain the OEM signature.*

Once launched, the RAM application reads the input parameters from the device's SRAM and provisions them accordingly in the SFlash region.

The SRSS registers (BOOT_DLM_CTL) handle

- DLM package validation requests to BootROM
- RAM application launch control
- RAM application execution status reporting

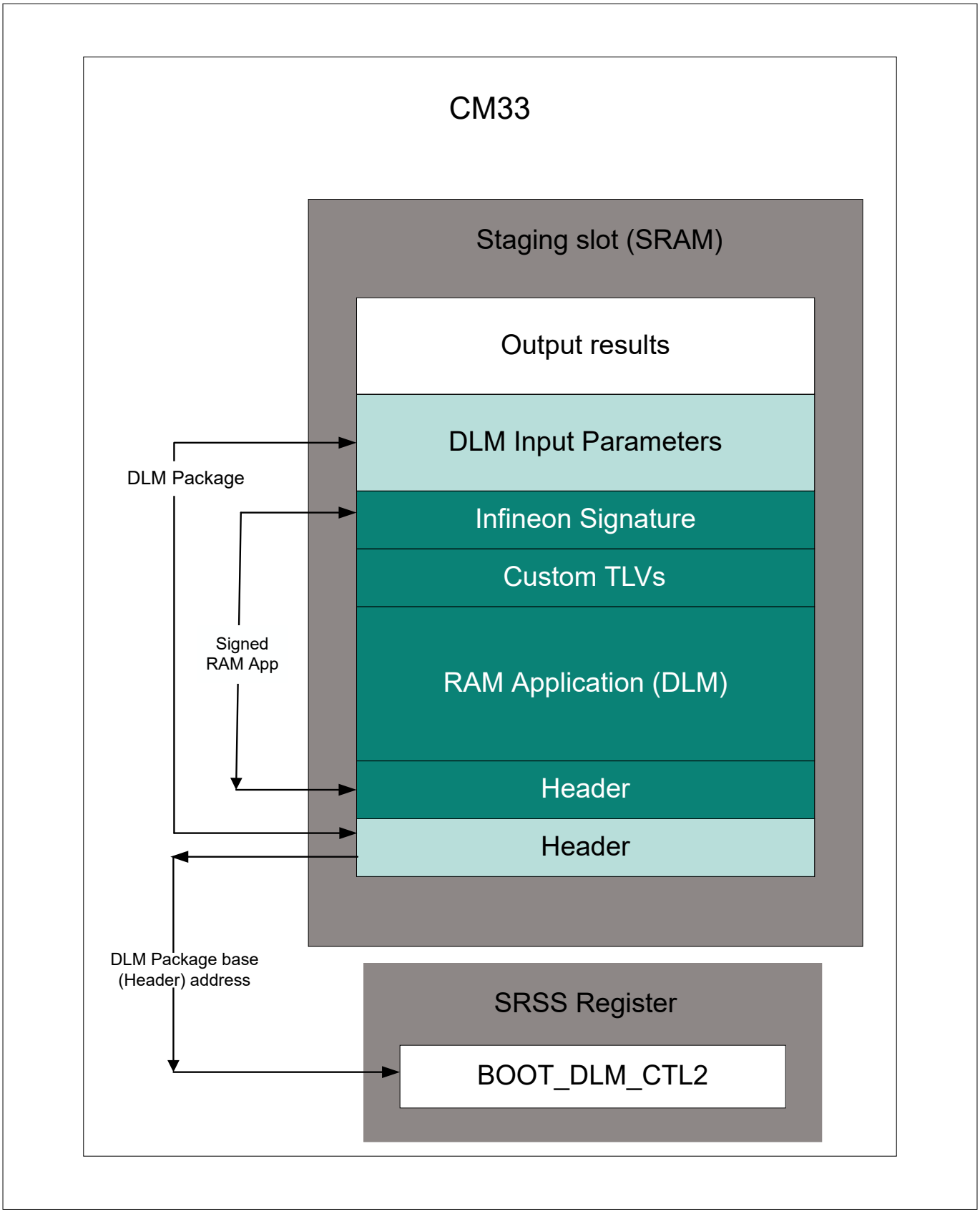


Figure 9 DLM package storage

Table 6 SRSS registers

SRSS register	Description
BOOT_DLM_CTL	Post requests for DLM package validation to the BootROM
BOOT_DLM_STATUS	Report the RAM application execution status in SRAM
BOOT_DLM_CTL_2	Store the DLM package base address in SRAM required by BootROM for validation
RES_SOFT_CTL	Trigger a device reset using the debugger (SYS-AP)

The input parameters, which are part of the DLM package placed in the device's SRAM, have their base address and maximum size stored by the ROM boot at a fixed SRAM location. This information is used by the RAM application to access the input parameters.

Similarly, the output results populated by the RAM application in the device's SRAM have their output result buffer base address and maximum size stored by the ROM boot at a fixed SRAM location. This information is used by the RAM application to write the output results.

See the Appendix A section for the list of RAM application status codes.

5.3 Creating a DLM package

Edge Protect Tools generates the DLM package from the pre-signed Infineon RAM application using the DLM parameters provided as input. The resulting DLM package includes the OEM signature, which is created using the OEM_ROT_PRIVATE_KEY. Additionally, the pre-signed Infineon RAM application binaries are included as part of the Edge Protect Tools package. Refer to the PSOC™ Control C3 MCU: OEM Provisioning code example for DLM package creation for a RAM application.

DLM package is created when *edgeprotecttools* provisioning command is run. It can be found in `<Workspace>/<CodeExampleName>/packets/apps/prov_oem/prov_oem_dlm.hex`

The DLM package can also be explicitly created with command below after running the edgeprotecttools init command.

```
# Create signed DLM package (the --key option may be skipped for BOOT_SIMPLE_APP)
$ edgeprotecttools -t psoc_c3x8 build-ramapp-package --app packets/apps/prov_oem/
cyapp_prov_oem_signed_ifx0.bin --inparams packets/apps/prov_oem/in_params.bin --key keys/
oem_rot_priv_key_0.pem --output packets/apps/prov_oem/prov_oem_dlm.hex
```

cyapp_prov_oem_signed_ifx0.bin - Infineon signed RAM_APP

in_params.bin - Input parameters to be provisioned to device

prov_oem_dlm.hex - Generated DLM hex file

5.4 Download and launch of the DLM package

This section provides details on downloading and launching the RAM application using different methods available for the device.

Method #0 uses edgeprotecttools to download the launch the DLM using standard provisioning commands.

Method #1 uses BootROM DFU to download the DLM package to device.

5.4.1 Launch flow: BootROM DFU

Configuring the DFU Interface for Provisioning

The Edge Protect Tools support two interfaces for device provisioning:

- OpenOCD (default)
- DFU

To use the DFU interface instead of OpenOCD:

- Configure the DFU Interface using the following command
- edgeprotecttools set-ocd -n serial -p `C:\Users\xxxx\ModusToolbox\packs\ModusToolbox-Industrial-MCU-Pack\tools\dfuh-tool`

After setting DFU interface, use standard provisioning commands. BootROM will handle communication through DFU protocol, no additional configuration is needed.

E.g.,: edgeprotecttools -t psoc_c3x8 provision-device `-p .\policy\policy_oem_provisioning.json`

The above command show provisioning the OEM policy using standard edgeprotecttools provisioning command over DFU.

A Appendix

A.1 RAM application (DLM) status codes

The RAM application status code is written to the BOOT_DLM_STATUS register. This code is read by the Edge Protect Tools package to provide command execution logs.

Table 7 RAM application status codes

Code	Status name	Description
0x45000002UL	CYAPP_BAD_PARAM	One or more invalid parameters
0x45000003UL	CYAPP_LOCKED	Resource lock failure
0x45000004UL	CYAPP_STARTED	Operation started, but not necessarily completed yet
0x45000005UL	CYAPP_FINISHED	Operation finished
0x45000006UL	CYAPP_CANCELED	Operation canceled
0x45000007UL	CYAPP_TIMEOUT	Operation timed out
0x45000010UL	CYAPP_APP_RUNNING	The provisioning application is in progress
0x45000020UL	CYAPP_OTP_INIT_FAILED	Failed to initialize OTP
0x45000021UL	CYAPP_OTP_BOOTROW_WRITE_FAILED	Failed to update the LCS
0x45000022UL	CYAPP_OTP_BOOTROW_READ_FAILED	Failed to read the LCS
0x45000023UL	CYAPP_OTP_WRITE_FAILED	Failed to program object into the OTP
0x45000024UL	CYAPP_OTP_READ_FAILED	Failed to read object from the OTP
0x45000025UL	CYAPP_OTP_BOOTROW_CORRUPTED	BOOTROW is corrupted
0x45000026UL	CYAPP_NUM_ZEROS_INVALID	-
0x45000030UL	CYAPP_LCS_INVALID	The current device LCS is invalid for the execution of the RAM application
0x45000031UL	CYAPP_OEM_KEY_ALREADY_REVOKED	The OEM key 0 was revoked. This operation can only be performed once
0x45000032UL	CYAPP_ICV_KEY_ALREADY_REVOKED	The ICV key 0 was revoked. This operation can only be performed once
0x45000033UL	CYAPP_SIGNATURE_VERIF_FAILED	Failed to verify the input parameter's digital signature
0x45000034UL	CYAPP_KEY_0_ALREADY_PROGRAMMED	The OEM key 0 was programmed. This operation can only be performed once
0x45000035UL	CYAPP_KEY_1_ALREADY_PROGRAMMED	The OEM key 1 was programmed. This operation can only be performed once
0x45000036UL	CYAPP_OEM_ASSETS_ALREADY_PROGRAMMED	The OEM assets were programmed. This operation can only be performed once
0x45000037UL	CYAPP_HASH_CALCULATION_FAILED	Hash calculation failed
0x45000038UL	CYAPP_INVALID_BOOT_CFG_ID	Invalid BOOT_CFG_ID configuration

(table continues...)

A Appendix

Table 7 (continued) RAM application status codes

Code	Status name	Description
0x45000039UL	CYAPP_HUK_GEN_FAILED	Failed to generate the Wrap Master Key
0X4500003AUL	CYAPP_DFU_FAILED	DFU flow failed
0X4500003BUL	CYAPP_PROT_FW_POLICY_DISABLED	Protected firmware policy is not configured
0X4500003CUL	CYAPP_MPC_POLICY_INVALID	MPC_POLICY is invalid
0X4500003DUL	CYAPP_OTP_NOT_DEFAULT	OTP assets are not zero
0X4500003EUL	CYAPP_BAD_APP_LAYOUT	Application start address and/or size is invalid
0X4500003FUL	CYAPP_PROT_FW_KEY_ALREADY_REVOKED	The PROT_FW key 0 was revoked. This operation can only be performed once
0x4500003EUL	CYAPP_HASH_CALCULATION_FAILED	Failed to calculate hash
0x4500003FUL	CYAPP_PROVISIONING_LIMITS_FAILED	Provisioning limits exceeded
0x45000100UL	CYAPP_PARAM_NV_CNT_INVALID	The input parameter NV counter is not valid
0x450000101UL	CYAPP_PARAM_OEM_KEY_0_HASH_INVALID	The input parameter OEM key 0 hash is not valid
0x450000102UL	CYAPP_PARAM_OEM_KEY_1_HASH_INVALID	The input parameter OEM key 1 hash is not valid
0x450000103UL	CYAPP_PARAM_PUBKEY_INVALID	The OEM public key used for digital signature verification does not match the OEM public key hash provisioned in the OTP
0x450000104UL	CYAPP_PARAM_CONTROL_WORD_INVALID	The input parameter control word is not valid
0x450000105UL	CYAPP_PARAM_TARGET_LCS_INVALID	The input parameter target LCS is not valid. Valid values are NORMAL, NORMAL_NO_SECURE, and SECURE
0x450000106UL	CYAPP_RMA_CERT_VERIF_FAILED	Failed to verify the certificate for RMA LCS transition
0x450000107UL	CYAPP_PARAM_ACCESS_RESTRICT_INVALID	The input parameter ACCESS_RESTRICT has an invalid configuration
0x450000108UL	CYAPP_RMA_ATTEMPT_COUNTER_INVALID	RMA_ATTEMPT_COUNTER is equal to zero
0x450000109UL	CYAPP_IAK_ALREADY_PROGRAMMED	IAK already programmed
0x45000010AUL	CYAPP_BOOTROW_CHECK_FAILED	BootROW check failed
0x45000010BUL	CYAPP_EFUSE_CHECK_FAILED	eFuse data check failed
0x45000010CUL	CYAPP_CBOR_INVALID	CBOR packet is invalid
0X4500010DUL	CYAPP_PROT_FW_POLICY_INVALID	PROT_FW_POLICY is invalid after decryption

(table continues...)

Table 7 (continued) RAM application status codes

Code	Status name	Description
0xF2A00001UL	CYAPP_SUCCESS	The RAM application is completed successfully
0xF2A00010UL	CYAPP_APP_RUNNING	The RAM application is in progress

A.2 Edge Protect Tools supported commands

Table 8 List of supported commands for PSC3 target device

Command	Description
bin2hex	Converts a binary image to a hex image
build-ramapp-package	Builds the DLM package with the RAM application and input parameters
cbor2json	Converts CBOR to JSON format
convert-key	Converts key to other formats
cose-sign	Creates a COSE_Sign message
cose-sign1	Creates a COSE_Sign1 message
cose-verify	Verifies the COSE_Sign message
create-key	Creates private and public key pairs
create-provisioning-packet	Creates a binary packet for device provisioning
debug-token	Creates debug token binary-based on template
device-info	Reads device information - Silicon ID, Family ID, Silicon revision
device-list	Lists the supported devices
encrypt-aes	Encrypts binary files using AES encryption
hash	Calculates hash of a file
hex-segment	Extracts a segment from the hex. file
hex2bin	Converts hex image to binary image
init	Initializes a new project
integrity-exam	Executes the integrity exam app
merge-bin	Merges different binary files into one
merge-hex	Merges different hex files into one
provision-device	Executes device provisioning
read-die-id	Reads die ID from the device
reprovision-device	Executes device re-provisioning
rma-token	Creates 'transit to RMA' or 'open RMA' token
serial-config	Configures serial interfaces

(table continues...)

Table 8 (continued) List of supported commands for PSC3 target device

Command	Description
serial-ports	Lists connected compatible hardware
set-ocd	Sets on-chip debugger
transit-to-rma	Transitions device to RMA lifecycle stage
verify-image	Verifies image with a key
verify-packet	Verifies provisioning packet with a key
version	Shows firmware version

To view list of commands supported by this specific device, execute the following in the shell CLI:

```
edgeprotecttools -t psoc_c3x8 --help
```

To view a list of options for a specific command, execute the following in the shell CLI:

```
edgeprotecttools -t psoc_c3x8 <COMMAND> --help
```

A.3 CBOR and COSE format

The Concise Binary Object Representation (CBOR) is a data format used in PSOC™ Control devices for concise data representation. The CBOR data format is defined as an Internet standard by RFC 8949, while the CBOR Object Signing and Encryption (COSE) format is defined in RFC 8152. COSE is used to ensure authentication, integrity, and confidentiality of exchanged data. Based on the JSON data model, the CBOR/COSE format is used to transport binary data such as certificates, firmware images, policy files, etc. by the Edge Protect Tools. Additionally, the CBOR web token (CWT), an open industry standard defined in RFC 8392 is used for wrapping a CBOR packet. For example, policy files are represented in JSON format. During the creation of a provisioning packet, the Edge Protect Tools converts the policy to CBOR, then wrap the payload to CWT for use as RAM application input parameters. When a COSE object is used, the optional parameter 'cose-type' can be used to identify the embedded object. The value to use with the parameter for each of the structures is defined in Table 9.

Table 9 CBOR tag and cose-type

CBOR tag	cose-type	Data item	Semantics
98	cose-sign	COSE-Sign	COSE Signed data object
18	cose-sign1	COSE_Sign1	COSE Single signer data object
96	cose-encrypt	COSE_Encrypt	COSE Encrypted data object
16	cose-encrypt0	COSE_Encrypt0	COSE Single recipient encrypted data object
97	cose-mac	COSE_Mac	COSE Mac data object
17	cose-mac0	COSE_Mac0	COSE Mac without recipient object

Edge Protect Tools can be used for conversion from CBOR to JSON format to create COSE_Sign and COSE_Sign1 messages, and to verify a COSE_Sign message. See Appendix A.2.

References

Application notes

[1] Infineon Technologies AG: *AN241270 – Getting started with PSOC™ Control C3M/P8, C3M/P7 MCUs on ModusToolbox™ software*; Available online

[2] Infineon Technologies AG: *AN242836 – Getting started with PSOC™ Control C3M/P8, C3M/P7 MCUs security*; Available online

Datasheet

[3] Infineon Technologies AG: *002-39793: PSOC™ Control C3 - PSC3P7xx, PSC3M7xx, PSC3P8xx, PSC3M8xx*; Available online

Reference manuals

[4] Infineon Technologies AG: *002-42141: PSOC™ Control C3 (PSC3P7xx, PSC3M7xx, PSC3P8xx, PSC3M8xx) architecture reference manual*; Available online

[5] Infineon Technologies AG: *002-41257: PSOC™ Control C3 (PSC3P7xx, PSC3M7xx, PSC3P8xx, PSC3M8xx) registers reference manual*

***Note**: For more documents related to PSC3M/P8, PSC3M/P7 devices, see PSOC™ Control C3 Performance Line webpage.*

Glossary

ASF
Alternate serial interface

CA
Certificate authority

CBOR
Concise Binary Object Representation. See the CBOR webpage.

CLI
Command-line interface

COSE
CBOR Object Signing and Encryption

CoT
Chain of Trust

CSR
Certificate signing request

CWT
CBOR web token

DFU
Device Firmware Update

DFU HT
Device Firmware Update Host Tool

DLM
Downloadable module. Alternatively referred to in this document as RAM application

DLM package
Package consisting of the RAM application (DLM), input parameters, and the OEM signature

EAP
Early access pack

EPC 2
Edge Protect Category 2. See Infineon Edge Protect.

FW
Firmware

GPIO
General-purpose input/output

I2C
Inter-integrated circuit

Glossary

LCS
Lifecycle stage

MCU
Microcontroller unit

MFG
Manufacturing

MPC
Memory protection controller

NVM
Non-volatile memory

OEM
Original Equipment Manufacturer. Infineon's customers.

OEM_ROT_DEV_PRIVATE_KEY
Private part of the OEM_ROT development key pair

OEM_ROT_DEV_PUBLIC_KEY
Public part of the OEM_ROT development key pair

OEM_ROT_KEY
OEM Root of Trust key pair

OEM_ROT_PRIVATE_KEY
Private part of the OEM_ROT key pair.

OEM_ROT_PUBLIC_KEY
Public part of the OEM_ROT key pair.

OTA
Over-the-air

OTP
One-time programmable

PC
Personal computer

PEM
Privacy Enhanced Mail

PSA
Platform Security Architecture

RAM application
RAM applications are developed, signed by Infineon, and loaded into SRAM for the purpose of provisioning and LCS transition.

Glossary

RMA
Return merchandise authorization

RoT
Root of Trust

RRAM
Resistive RAM

RSA
Rivest–Shamir–Adleman

SFDP
Serial Flash Discoverable Parameter

SPE
Secure processing environment

SPI
Serial Peripheral Interface

SRSS
System resources subsystem

SWD
Serial Wire Debug

TLV
Tag-length-value

UART
Universal Asynchronous Receiver Transmitter

USB
Universal Serial Bus

Revision history

Document revision	Date	Description of changes
**	2026-03-27	Initial release
*A	2026-09-18	Release to web

Trademarks

PSOC™, formerly known as PSoC™, is a trademark of Infineon Technologies. Any references to PSoC™ in this document or others shall be deemed to refer to PSOC™.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2026-09-18

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG

All Rights Reserved.

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference

IFX-yus1772016962406

Important notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability. Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury.

However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer. Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG). If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.

For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit www.infineon.com.