MCUXSPTUG_25.09_ZH

MCUXpresso 安全配置工具用户指南 v. 25.09

Rev. 18 — 10 October 2025 **User guide**

Document information

Information	Content
Keywords	MCUXpresso 安全配置工具
Abstract	MCUXpresso Secure Provisioning Tool (SEC，MCUXpresso 安全配置工具） 是一个旨在简化 NXP MCU 平台上可启动的可执行文件的生成和配置过程的 GUI 工具。它建立在由 NXP 提供的、经验证 的且成熟稳定的安全特性布署工 具集之上，并利用了 BootROM 提供的丰富的编程接口。

1 Introduction

可以从 Secure Provisioning Tool home page 下载 PDF 格式的中文版用户指南。

MCUXpresso Secure Provisioning Tool (SEC) 是一款图形界面（GUI）工具，旨在简化NXP MCU 可启动执行文件的生成与部署流程。该工具基于恩智浦成熟稳定的安全使能工具集构建，并充分利用BootROM提供的多样化编程接口。

对于新用户，SEC 提供了直观的操作体验，可便捷完成镜像（image）的准备、烧录与安全配置；而对于资深用户，可在图形界面下复用现有工具（如 sdphost、blhost、nxpimage 等），并通过修改工具生成的脚本，灵活定制安全配置流程，满足复杂场景需求。

对于高级用例，SEC 还通过可编辑脚本支持自定义，使经验丰富的用户能够定制配置流程以满足特定的安全和部署要求。

Figure 1. MCUXpresso 安全配置工具

2 Features

MCUXpresso 安全配置工具 有以下功能：

2.1 Security Enablement

- 镜像签名和加密：支持使用客户提供的密钥和证书生成签名和加密镜像。
- 安全启动配置：自动创建安全启动头（例如 HAB、TrustZone-M、BEE）。
- 密钥管理：集成 NXP 的密钥配置工作流程，包括 SRK、DEK 和 OTP 密钥编程。
- 可选签名提供程序：允许自定义集成 HSM 模块来签名镜像。
- 信任配置：设备 HSM 和 EdgeLock 2GO

2.2 Device communication and flashing

- BootROM 接口支持：通过 UART、USB、SPI 或 I2C 与 BootROM 通信。
- 闪存编程：支持写入内部闪存和外部存储器（例如 QSPI、FlexSPI NOR/NAND、eMMC）。
- 设备检测和连接管理：自动检测连接的设备并管理通信。

2.3 User experience

- 引导式工作流程：带有预定义配置文件的工作区向导，可轻松创建新配置。
- 可视化反馈：实时状态更新、日志和错误报告。
- 跨平台支持：适用于Windows、Linux和mac操作系统。

2.4 Useful extensions

- 调试认证支持
- SB 编辑器，允许创建安全二进制文件
- 合并工具，允许将多个镜像合并为一个
- MCUboot 签名器，允许通过 MCUboot 第三方工具签名自定义应用程序
- 制造工具用于 FAB 操作，允许并行配置多个设备
- 用于与设备进行低级交互的附加命令行实用程序

2.5 Device and platform support

- 广泛的 MCU 系列覆盖：兼容各种 NXP MCU（例如 i.MX RT、LPC、MCX）。
- 支持最新的硅片功能：定期更新以支持新的安全功能和硅片版本。

3 Terms and definitions

术语和定义

术语	定义
AAD	附加认证数据；通常与AES-GCM一起使用
AES	Advanced Encryption Standard - 标准对称加密算法
AES-128	Rijndael cipher (128-bit block/key)
AHAB	Advanced High Assurance Boot - 高可信启动协议
ATF	ARM Trusted Firmware
BCA	Bootloader Configuration Area
BEE	Bus Encryption Engine - 片上总线加密引擎
Block cipher	Block cipher: N={64,128,…}位分组加密算法
CA	Certificate Authority
CAAM	Cryptographic Acceleration and Assurance Module - 密码算法加速与保证模块
CBC	Cipher Block Chaining
CBC MAC	CBC-MAC: 分组密码消息认证码
CFPA	Customer In-field Programmable Area
Cipher block	Cipher block: 分组密码最小操作单元
Ciphertext	Ciphertext: 加密数据
CMAC	基于密文的消息认证码
CMPA	Customer Manufacturing/Factory Programmable Area

术语	定义
CMS	Cryptographic Message Syntax - 通用加密数据格式（支持数字签名/加密信封）HAB 将 CMS 作为包含 PKCS#1 签名的容器。
CSEc	加密服务引擎/EdgeLock加速器，包含各种加密、解密和身份验证算法的功能集，旨在满足SHE规范。
CSF	Command Sequence File - 二进制结构（由HAB执行身份验证操作）
DA	Debug Authentication - 调试认证
DAP	Debug Authentication Protocol - 调试认证协议
DCD	Device Configuration Data - 二进制表（ROM代码早期启动配置）
DCP	Data Co-Processor - AES/SHA加速器
DEK	Data Encryption Key - 一次性会话密钥（加密主要引导映像）
DUK	Device Unique Key - 设备唯一密钥（每个设备独有的加密密钥）
DUKB	Device Unique Key Certificate Block - 存储设备唯一密钥证书的数据块
ECB	Electronic Code Book - 电子密码本（无反馈的加密模式）
EKIB	Encrypted Key Information Block - 加密密钥信息块
ECU	电子控制单元
ELE	EdgeLock Secure Area - EdgeLock安全区域
EPRDB	Encrypted Protection Region Descriptor Block - 加密保护区域描述符块
FAC	Flash Access Control - Flash访问控制
FCB	Flash Configuration Block - Flash配置块（用于设备启动配置）
FCF	Flash Configuration Field - Flash配置字段
GCM	伽罗瓦/计数器模式；对称密钥密码块密码的操作模式，最常与AES一起使用
GUI	图形用户界面
HAB	High Assurance Boot - 高保证启动，在启动时在飞思卡尔处理器内部ROM中执行的软件库，除其他功能外，通过根据CSF验证数字签名来认证外部存储器中的软件。本文档严格限于运行HABv4的处理器
Hash	Hash - 摘要演算算法
HSM	Hardware Security Module - 硬件安全模块
IEE	Inline Encryption Engine - 在线加密引擎
IFR	Information Flash Region - 信息Flash区域
IMG	Image Signing Key - 图像签名密钥（与ISK可互换）

术语	定义
ISK	Image Signing Key - 图像签名密钥（与IMG可互换）
ISP	In-system Programming - 系统在线编程（直接编程写入产品代码的模式）
IVT	Image Vector Table - Image向量表
KEK	Key Encryption Key - 密钥加密密钥（用于加密会话密钥或DEK）
KeyBlob	KeyBlob - 密钥数据结构（使用AESCTR算法包装密钥和计数器）
KIB	Key Information Block - 密钥信息块（定义AES128-CBC的image向量和召回密钥）
MAC	Message Authentication Code - 消息验证码（提供完整性和身份验证检查）
Message digest	Message digest - 消息摘要（使用哈希算法计算的唯一值，仅提供完整性检查）
NBU	窄带单元
NDA	Non-disclosure Agreement - 非公开协定
OEI	Optional Executable Image - 可选的可执行image
OEM	Original Equipment Manufacturer - 原始设备制造商
OS	Operating System - 操作系统
OTFAD	On-The-Fly AES Decryption - 实时AES解密
OTP	One-Time Programmable - 一次性可编程（包含掩模ROM和eFuses）
OTPMK	One-Time Programmable Master Key - 一次性可编程主密钥
PFR	Protected Flash Region - 受保护的Flash区域
PKCS#1	Public Key Cryptography Standards #1 - RSA算法标准更多详情请参见 PKCS#1 Wikipedia article 和 RSA PKCS#1 安全存档 Security archive。
PKI	Public Key Infrastructure - 公钥基础设施（证书层次结构）
Plaintext	Plaintext - 明文文本（未加密的数据）
PRDB	Protection Region Descriptor Block - 保护区域描述符块（使用AES-CTR算法调用计数器）
PRINCE	为低延迟硬件实现设计的轻量级块密码，特别适用于嵌入式系统和物联网设备等环境
PUC	产品单元证书
PUF	Physical Unclonable Function - 物理不可克隆功能
pyOCD	基于 Python 的工具和 API，用于调试、编程和探索 Arm Cortex 微控制器；有关详情，请参阅 pyOCD official website
Rijndael	美国政府选择的取代 DES 的分组密码， *rain-dahl*
ROMCFG	ROM Boot Configuration - ROM启动程序配置

术语	定义
RoT	Root of Trust - 信任根
RSA	RSA Algorithm - 公钥密码算法（含哈希加速器）
RSA-PSS	RSA Probabilistic Signature Scheme - RSA概率签名方案
SDP	Serial Download Protocol - 串行下载协议（通过UART/USB下载代码）IT 允许在生产和开发阶段通过 UART 或 USB 提供代码。
SEC Tool	Secure Provisioning Tool - 安全配置工具
Session key	加密密钥是在加密时生成的。（一次性加密密钥）
SHA-1	Secure Hash Algorithm 1 - 生成160位消息摘要的哈希算法
SHE	Secure Hardware Extension - 安全硬件扩展
SI	Secure installer - 安全安装程序
SNVS	Secure Non-Volatile Storage - 安全非易失性存储
SPL	Secondary Program Loader - 辅助程序加载器
SPSDK	Secure Provisioning SDK - 安全配置SDK（开源工具）
SRK	Super Root Key - 超级根密钥（启动验证链起点，嵌入处理器OTP硬件）SRK 公钥的哈希嵌入在使用 OTP 硬件的处理器中。SRK 私钥由 CA 持有。除非明确说明，本文档中的 SRK 仅指公钥。
TEE	Trusted Execution Environment - 可信执行环境
UID	Unique Identifier - 唯一标识符（制造分配的序列号）
UUU	Universal Update Utility - 通用更新实用程序（用于下载镜像到MPU设备）
V2X	V2X - 车联网（i.MX 95独立加密加速器）
WPC	无线充电联盟
XIP	Execute-In-Place - 就地执行（直接从非易失性存储运行镜像）
XMCD	External Memory Configuration Data - 外部存储器配置数据类型

4 Installation

本章介绍在 Windows, Mac 和 Linux 操作系统上安装 SEC 所需的过程。有关支持的操作系统列表，请参阅 Minimum system requirements.

4.1 Minimum system requirements

该工具在 Microsoft(R) Windows(R)、Ubuntu、和 Mac 操作系统上运行。有关详细的系统要求，请参阅Secure Provisioning Tool Release Notes。该文档还描述了应该安装哪些调试探针驱动程序。这些驱动程序可能还需要在 ISP 模式下与处理器进行通信，其中探测器提供 USB 到 UART/SPI/I2C 转换器。

4.2 Windows

要将 SEC 桌面应用程序安装在 Windows 平台上，请执行以下步骤：

1. 访问Secure Provisioning Tool home page 下载适用于 Windows 操作系统的 SEC 安装程序。
2. 双击 `MCUXpresso_Secure_Provisioning_<version>.exe` 安装程序开始安装。
3. 在 wizard 的第一页上，点击 **Next**。

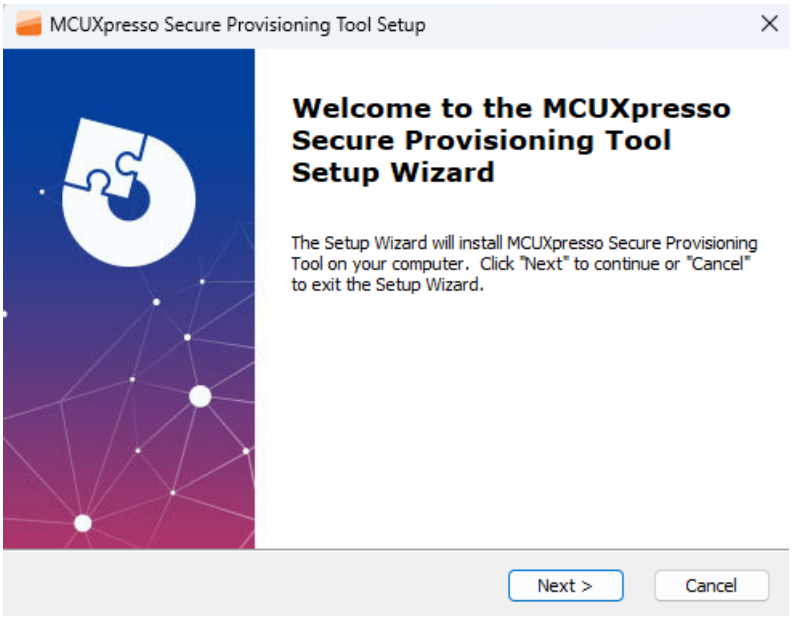

Figure 2. 安全配置工具设置

4. 在 Wizard 的 **End-User License Agreement** 页面，选择 **I accept the terms of the License Agreement** 并点击 **Next**。

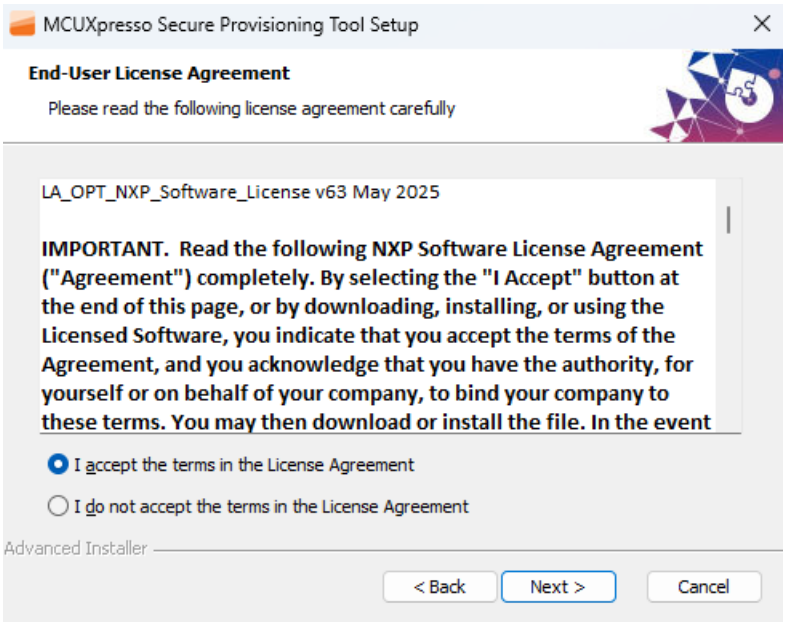

Figure 3. 接受最终用户许可协议

5. 在 Wizard 的 **Select Installation Folder** 页面，选择 **Browse** 并导航到要安装 SEC 的 目标文件夹，点击 **Next**。

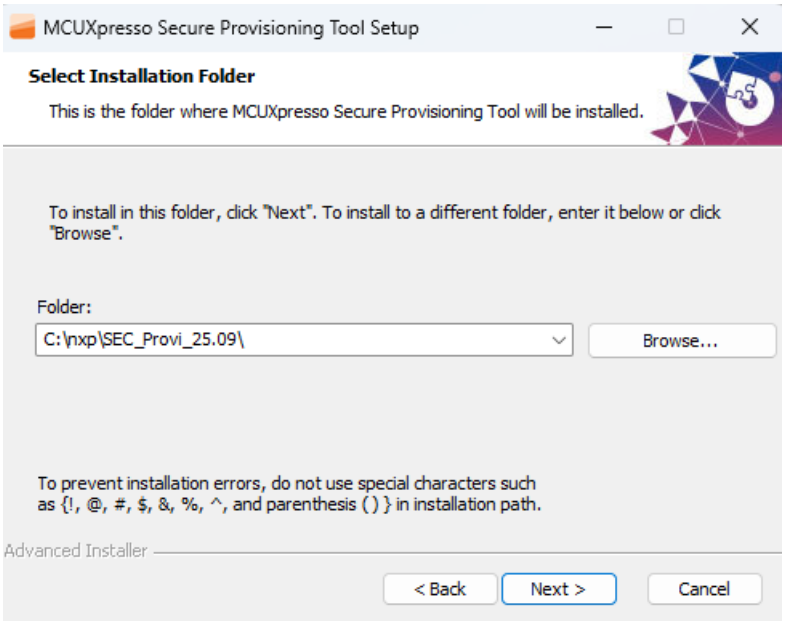

Figure 4. 选择安装文件夹

6. 在 Wizard 的 **Configure Shortcuts** 页面，选择要为 SEC 创建的快捷方式，并点击 **Next**。

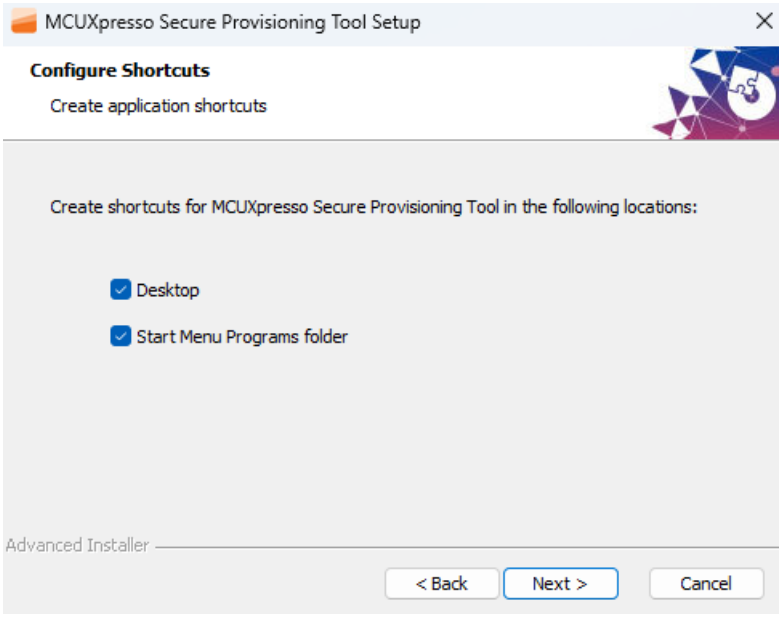

Figure 5. 配置快捷方式

7. 在安装向导的 **Ready to Install** 页面，选择 **Install**。
   安装程序开始安装。
   注意： 如果要查看或更改任何安装设置，请点击 **Back**。点击 **Cancel** 退出 Wizard。
   安装完成时，安装程序会给您提示。

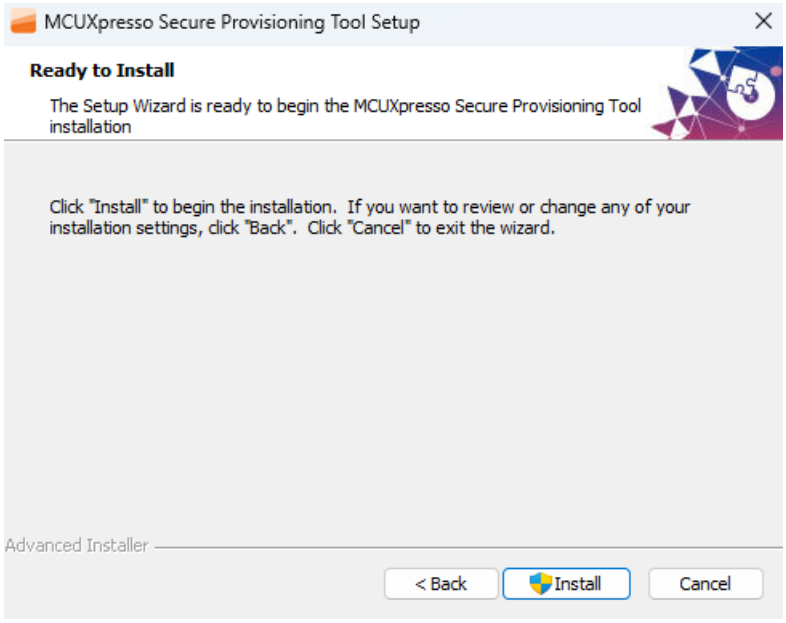

Figure 6. 准备安装

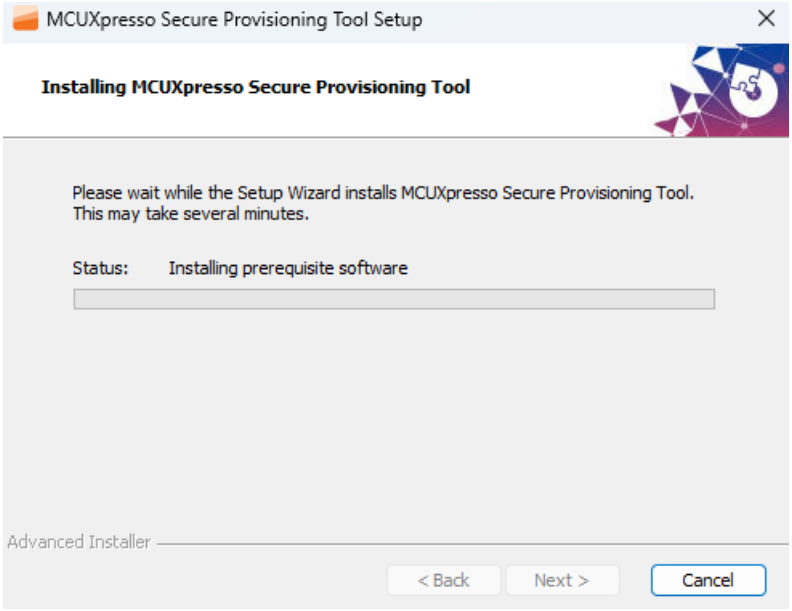

Figure 7. 安装 **MCUXpresso** 安全配置工具

8. 点击 **Finish** 关闭并退出安装向导。

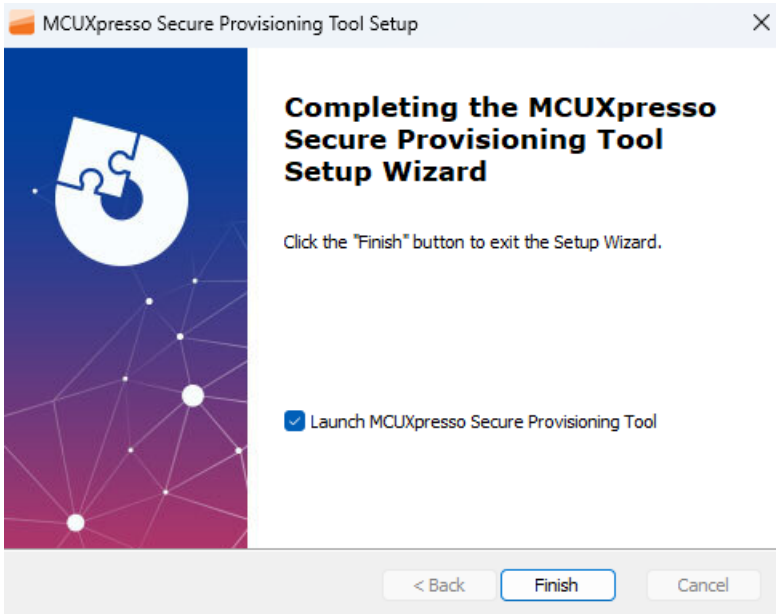

Figure 8. 完成安装

9. 要开始使用SEC，请从桌面的快捷方式或从 **Start** 菜单运行该工具。您还可以导航到 *<product installation folder>\bin* 文件夹，启动 `securep.exe` 或启动 *<product installation folder>* 中的快捷方式 。

4.2.1 Windows CLI

可以使用命令行安装SEC工具。在这种情况下，请使用带有以下参数的 **Run the installer**：start /wait

```
MCUXpresso_Secure_Provisioning_YY.MM.exe /exenoui /qn
```

4.3 MacOS

要将 SEC 桌面应用程序安装在 Windows 平台上，请执行以下步骤：

1. 请访问 NXP 官网 下载适用于 Mac 操作系统的 SEC 安装程序。根据您的计算机，选择适用于 Intel 或 Apple M 处理器的安装程序。
2. 双击 `MCUXpresso_Secure_Provisioning_YY.MM.pkg` 进入 **Install MCUXpresso Secure Provisioning Tool** 安装向导。

注意： 当您尝试打开 Mac 操作系统安装程序时，您可能收到 error 消息。为避免这种情况，请在 **Security & Privacy** 菜单中手动选择 **Mac App Store and identified developers** 选项。

1. 在 **Introduction** 页面，点击 **Continue**。

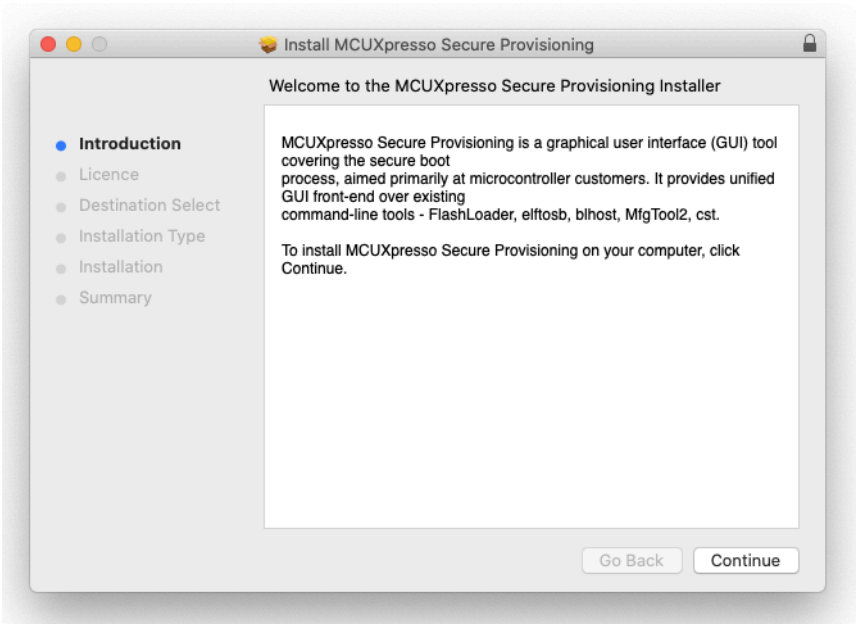

Figure 9. Introduction 页面

2. 在 **Software License Agreement** 页面，点击 **Continue**。

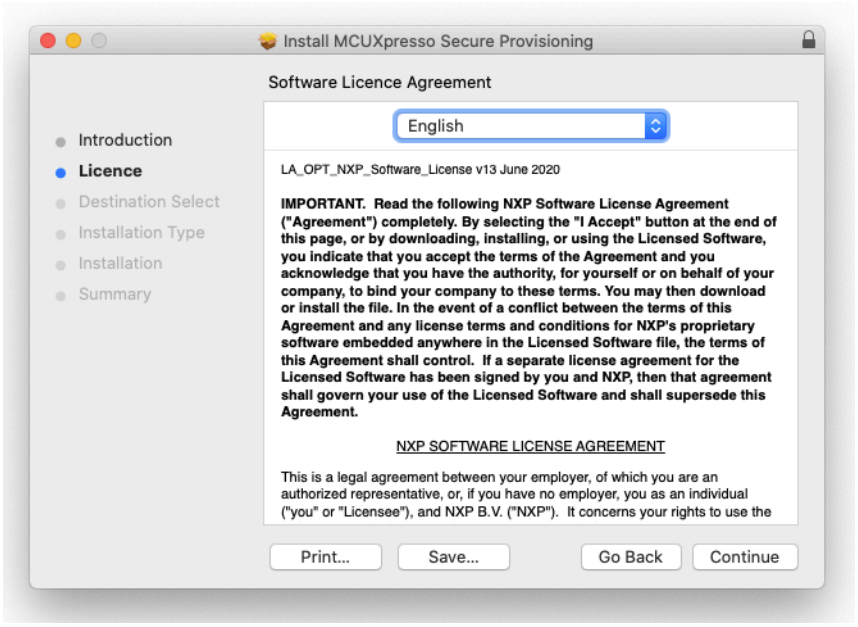

Figure 10. 软件许可协议

3. 请确认您已阅读并同意软件许可协议的条款，并点击 **Agree**。

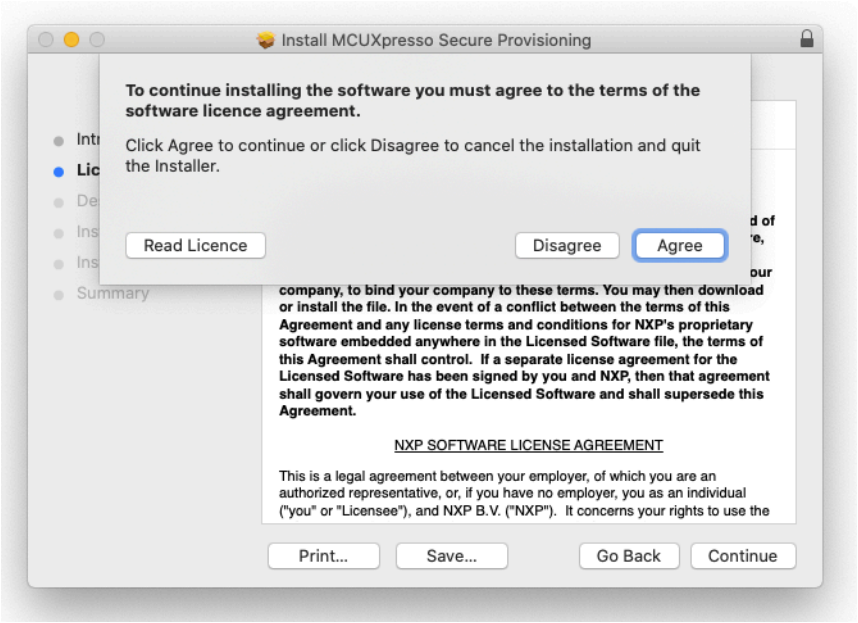

Figure 11. 接受软件许可协议

4. 在 **Destination Select** 页面，点击绿色箭头选择安装文件夹。完成后点击 **Continue**。

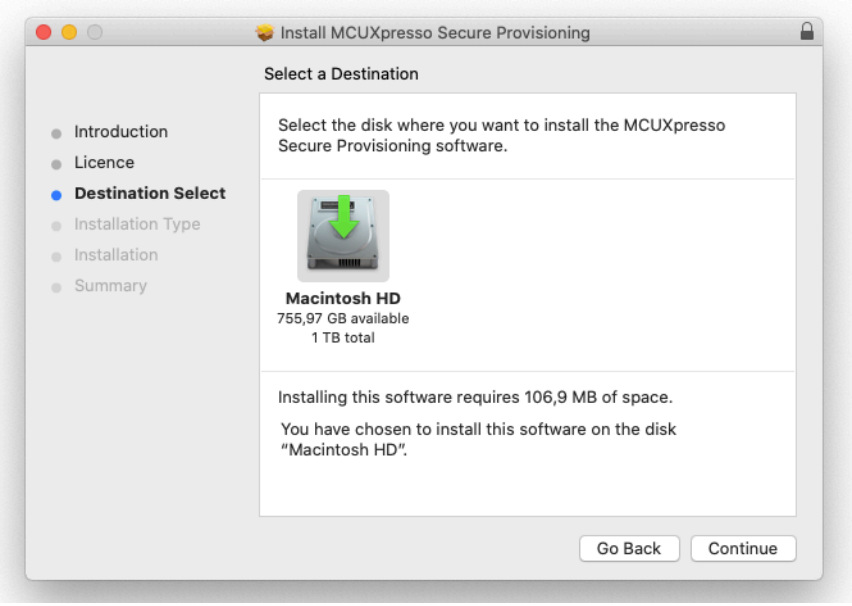

Figure 12. Destination Select 页面

5. 在 **Installation Type** 页，点击 **Install**。

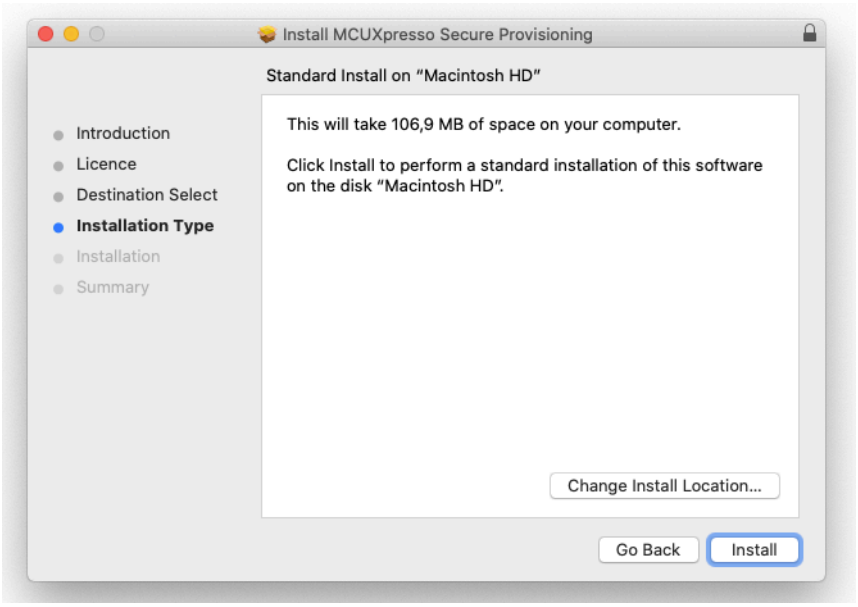

Figure 13. Installation Type 页面

6. 输入您的管理员用户名和密码以继续安装，然后点击 **Install Software**。

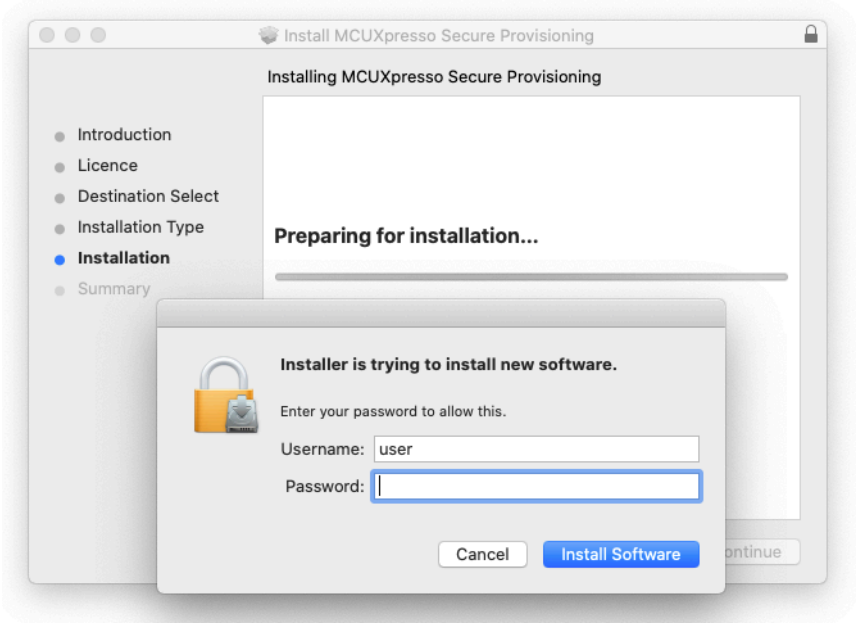

Figure 14. Installation 页面

7. 点击 **Continue**。
   除非报告错误，否则 **Summary** 页面会确认安装已成功完成。
8. 在 **Summary** 页面，点击 **Close**。

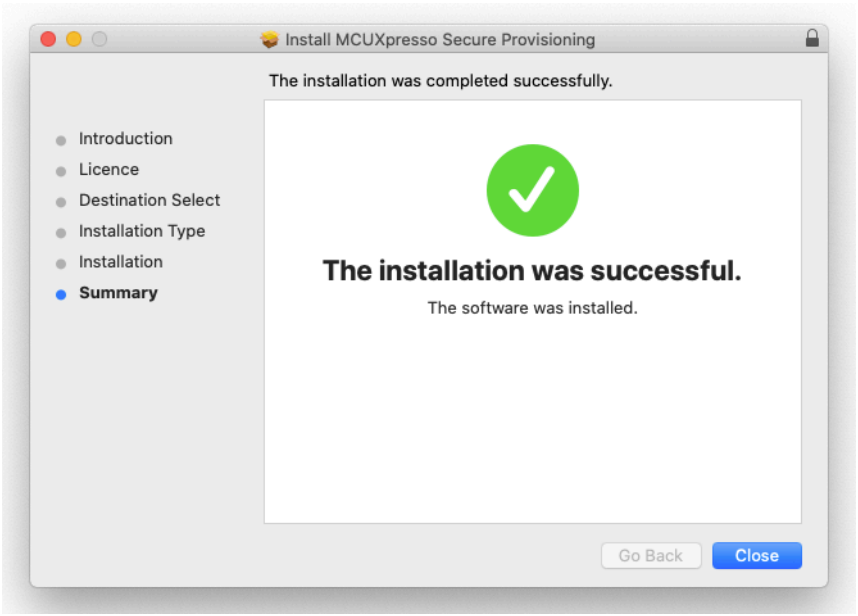

Figure 15. Summary 页面

4.3.1 Enabling USB connection on macOS

在第一次通过 USB 连接到目标时，MacOS X Catalina 将采取安全措施，阻止对 USB HID 设备的访问，操作将失败并出现 error。在 Mac 操作系统 13 (Ventura) 中，此操作的工作方式有所不同，并且不需要这些步骤。

请执行以下步骤以启用 USB 连接：

1. 在操作系统安全警报消息框中，选择 **Open System Preferences**。

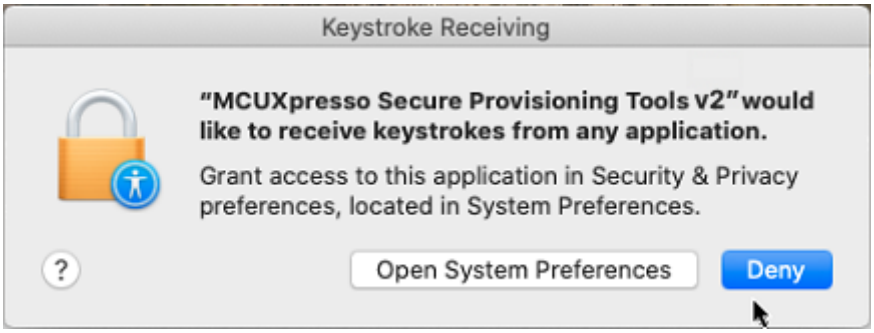

Figure 16. 打开系统偏好设置

2. 解锁 **Privacy preferences** 以启用更改。
3. 选择 **MCUXpresso Secure Provisioning Tool YY.MM**，确认，并退出应用程序。

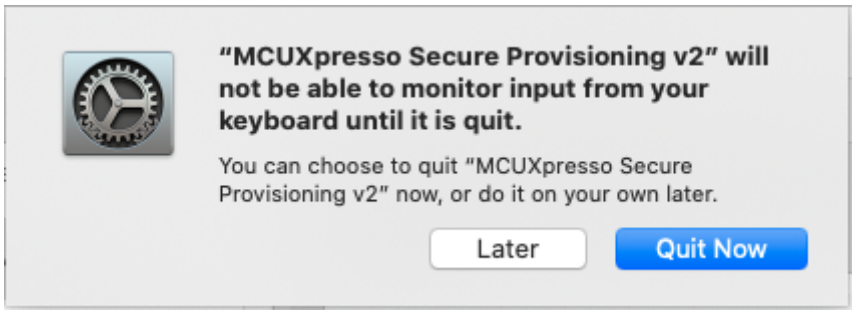

Figure 17. 确认更改

4. 锁定 **Privacy** 偏好设置。
5. 如果应用程序未关闭，请手动将其关闭。
6. 启动应用程序并继续操作。

4.4 Linux

在 Ubuntu 上对SEC的安装可以在终端完成。

1. 请访问 NXP 官网 下载适用于 Linux 操作系统的 SEC 安装程序。
2. 打开终端并更改下载安装程序的目录，使安装程序可执行。请使用 sudo 运行以下命令：

```
$ cd ~/Downloads
$ sudo dpkg -i mcuxpresso-secure-provisioning-<version>\_<architecture>\_<os-version>.deb
```

如果使用 sudo 执行的命令成功，安装程序会将SEC工具安装在专用文件夹 /opt/nxp/中。

注意： SEC 依赖于必须提前安装的其他软件包：

```
$ sudo apt install libhidapi-dev libsdl2-2.0-0
```

4.5 Uninstalling

4.5.1 Windows

Secure Provisioning Tool 可以通过以下方式卸载:

- 使用 Settings|Apps & features

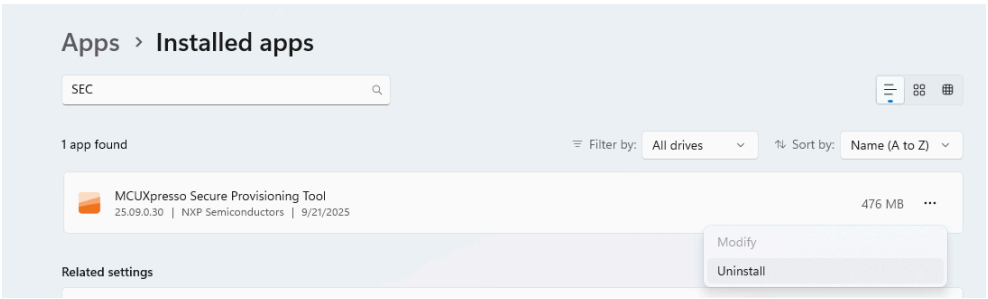

Figure 18. Settings 页面

- 进入 `%APPDATA%\NXP Semiconductors`，然后在 `Secure Provisioning Tool YY.MM\install\` 文件夹中找到适当的 MSI 安装程序，并在安装向导中选择 **Remove** 选项。

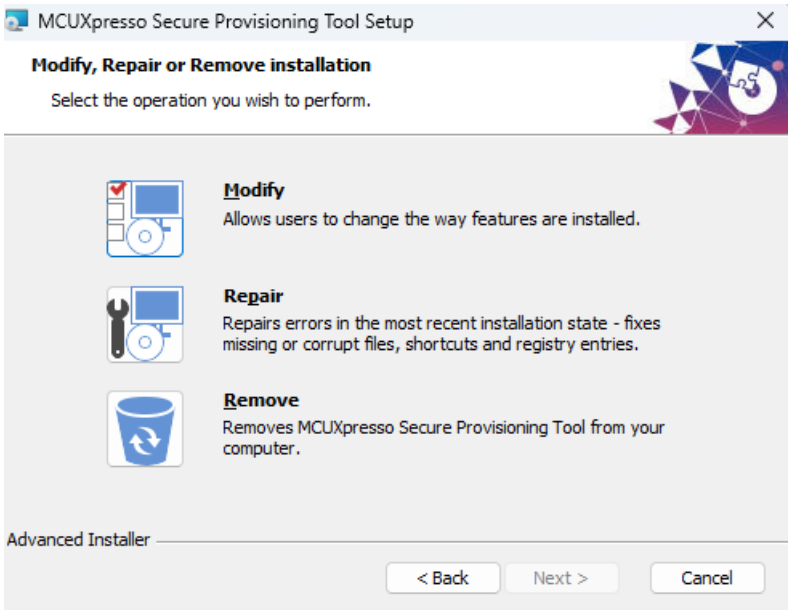

Figure 19. MCUXpresso 安全配置工具使用向导

4.5.2 MacOS

可以使用 Finder 来卸载安全配置工具, 导航到 Application, 将 `SEC_Provi_##.##` 移动到 Trash。

4.5.3 Linux

可以使用 Debian 软件包管理器卸载安全配置工具。

在终端（Terminal）中，您可以获得带有包名称（package name）的安全配置工具列表：

```
$ dpkg --list "\*-secure-provisioning\*"
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/halF-conf/Half-inst/trig-aWait/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                                    Version      Architecture
 Description
+++-=======================================-============-============-
===================================
ii  mcuxpresso-secure-provisioning-25.09    25.09        amd64        MCUXpresso
 Secure Provisioning Tool
ii  mcuxpresso-secure-provisioning-v3.1     3.1          amd64        MCUXpresso
 Secure Provisioning
ii  mcuxpresso-secure-provisioning-v4       4.0          amd64        MCUXpresso
 Secure Provisioning
```

可以卸载不再需要的版本：

```
$ sudo dpkg --remove mcuxpresso-secure-provisioning-25.09
```

4.5.4 Remove configuration files

用户首选项（user preferences）存储在 <user home>\.nxp\secure_provisioning_<version>\ 中，卸载安全配置工具后，它不会被移除。对受限数据的移除可以通过手动删除来实现。有关用户首选项的更多细节，请参见首选项。

4.5.5 Remove restricted data

受限数据安装在 <user home>\.nxp\secure_provisioning_restricted_data\ 文件夹中，卸载安全配置工具后，它不会被移除。对受限数据的移除可以通过手动删除来实现。

5 User interface

SEC 提供了一个简单友好的用户界面。它由 菜单栏、工具栏、和可通过页面访问的主视图组成：

- **Build image** 允许镜像。
- **Write image** 允许将可启动镜像写入处理器并选择性地保护它。
- **PKI management** 允许生成身份验证密钥或配置签名提供程序。
- 界面的底部被 **Log** 窗口和状态行所占据。

所选处理器不支持的项目不会显示在工具中。如果某个项目被所选处理器支持，但在当前设置中没有意义，则该项目是禁用状态（灰色）。界面里包含任何配置问题的项目将以红色（错误）或黄色/橙色（警告）突出显示。该问题会在工具提示框中描述。信息消息显示为蓝色。

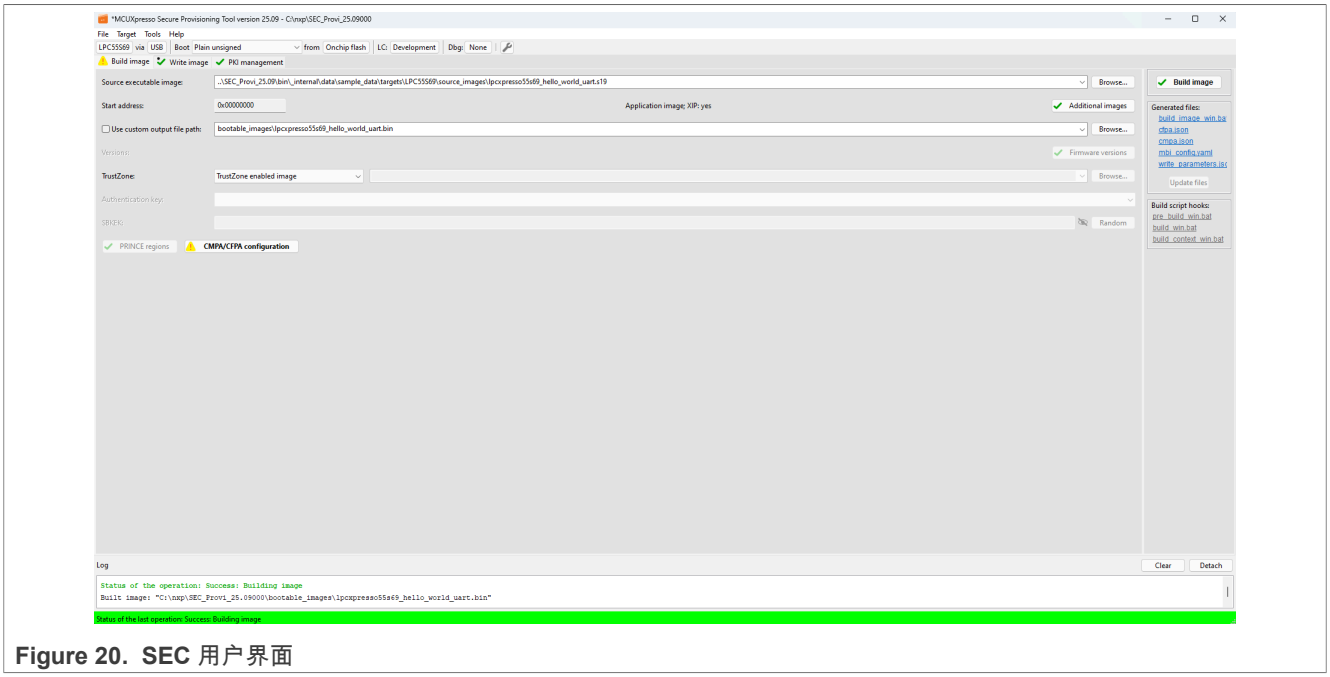

Figure 20. SEC 用户界面

5.1 Menu and settings

本章提供有关该工具的菜单和设置的详细信息。

5.1.1 Title of the Main Window

主窗口的标题包含：

- 星号 (*) 表示配置未保存到磁盘（即配置“已损坏”）。
- 工具的名称
- 当前 workspace 的路径

5.1.2 Menu bar

菜单栏（主菜单）包含多个下拉菜单，提供各种应用程序，配置和文件相关的功能。

File：一般 Workspace 和配置相关操作

- **New Workspace ...**：创建 workspace 。系统将提示您指定它的位置并从支持的处理器中进行选择。如果指定的位置已经包含一个workspace，将直接打开这个 workspace 而不创建新 workspace。有关更多详情，请参阅 Workspaces。
- **Import Manufacturing Package ...**：导入包含制造所需的所有数据的制造包 (*.zip) 并创建“量产 workspace ”（有关制造 workspace 的详情，请参阅Workspaces）。
- **Select Workspace ...**：切换到另一个 workspace。按系统提示指定将要打开的 workspace。更多详情，请参阅 Workspaces。
- **Recent Workspaces**：显示最近使用的 workspace 列表。更多详情，请参阅 Workspaces。显示的 workspace 的数量可以在 Preferences 中自定义。
- **Save Settings**：保存当前 workspace 的设置。
- **Export Workspace**：将当前 workspace 导出为包 (*.zip)，并可选择过滤掉不必要的文件（例如生成的脚本）。有关更多详情，请参阅 Sharing and copying workspaces。
- **Import Workspace**：将 workspace 包 (*.zip) 导入指定文件夹。
- **Explore Workspace...**：在当前 workspace 中打开文件资源管理器。
- **Open Terminal**：使用为 SPSDK 应用程序配置的路径打开当前 workspace 目录中的终端。
- **Preferences**：打开 **Preferences** 对话框。更多详情，请参阅 Preferences。
- **Exit**：退出 SEC。

Target： 此菜单重复了工具栏中可用的操作，更多详情，请参阅 Toolbar。

Tools：更多工具列表

- **Manufacturing Tool**：打开 **Manufacturing Tool**。更多详情，请参阅 Manufacturing Tool。
- **Flash Programmer**：打开 Flash 编程和编辑工具。更多详情，请参阅 Flash Programmer。
- **SB editor**：允许创建用于安全更新的自定义安全二进制文件。更多详情，请参阅 SB editor。
- **Merge Tool**：可将最多 8 个图像合并为一个二进制图像。更多信息，请参阅 Merge Tool。
- **MCUboot - Sign Image...**：允许使用 MCUboot 辅助引导加载程序的密钥对应用程序 镜像 进行签名。
- **MCUboot - Export Key...**：将给定私钥中的公钥导出为 MCUboot 辅助引导加载程序的 C 源文件。

Help：用户帮助和其他一般信息

打开本地文档的命令

- **User Guide**：打开用户指南。
- **Release Notes**： 打开发行说明 markdown 文件。
- **Quick Start Guide**：打开快速入门指南 PDF。

打开在线文档的命令

- **Online Documentation**：打开在线 SEC 工具文档（快速入门指南、用户指南和发行说明）。
- **Community**：打开一个带有博客的 NXP 网页，在那里您可以找到与 SEC 相关问题的讨论。
- **SPSDK Online Documentation**：显示一个带有 SPSDK 命令行工具文档的网页。

其它命令

- **Check for Updates**：检查该工具是否有可用的新版本。
- **About**：显示当前版本信息。

5.1.3 Preferences

用户首选项存储在 `<user home>\.nxp\secure_provisioning_<version>\` 文件夹下，并为所有 workspace 所共享。首选项是向后兼容的，例如 SEC v9 的首选项尚不存在，那么 SEC v9 可以加载使用 SEC v8 里的首选项。如果没有首选项可用，SEC 将以默认值开始。

用户首选项包含最近使用的文件和 workspace 的信息，以及在首选项对话框中可配置的选项：

Timeout for communication re-established after reset(flashloader to be initialized) [sec]： 表示在处理器重置后，ROM bootloader 程序或 flashloader 程序准备就绪所需要的延迟时间（以秒为单位）。实际值可能会受到主机配置的影响。选择的值可能会影响生成的写入脚本。

Maximal number of recent workspaces displayed in the File menu： 自定义在 **File > Recent Workspaces** 中显示的最近 workspace 的最大数量。支持范围 1 - 25。默认值： 9

Read current values after the OTP/PFR configuration is opened：选择打开 **OTP Configuration** 时读取处理器值的方式。

有以下选项：

- **Never**： 不自动读取值
- **Ask**：手动确认读取
- **Always**：自动读取设备值

Preferred language for the tool：选择工具要显示的语言。支持的语言为英文和中文。

有以下选项：

- **Default**：语言的选择基于操作系统中选择的语言。如果系统语言与支持的语言不同，则将使用英文。
- **EN**： 将工具设置为英语
- **ZH**：将工具设置为中文

Save tool settings：指定何时必须将工具设置保存到磁盘。

可以选择以下选项之一：

- **Automatically**：此为默认值。如果需要，设置会被保存。
- **On request only**：工具始终询问是否保存设置

Sound on error during configuration：如果在配置对话框中显示了新的错误，该工具会用声音信号通知用户。声音信号是特定于操作系统的。

Password/key visibility at startup：指定在 SEC 工具启动时显示或隐藏密码和密钥的时间。在用户界面中，可见性可以通过按钮控制，因此此选项仅配置初始值。

Check for the new version of the tool： 指定工具在启动时检查新版本可用性的频率。

可以选择以下选项之一：

- **Never**： 从不检查；该功能被禁用
- **Daily**： 每天执行一次检查
- **Weekly**： 每周执行一次检查
- **Monthly**： 每月执行一次检查

Help NXP improve the tool by sending diagnostical data (帮助 **NXP** 通过发送诊断数据改进工具)： 如果启用，此功能允许收集诊断数据以生成关于工具使用情况的统计洞察。这有助于我们增强功能并优先考虑改进。如果禁用，则不会收集或传输任何数据。

Statistical Data Collection Notice (统计数据收集通知)

：为了支持我们软件的持续改进，我们收集诊断使用数据，包括：

- 工具和用户界面功能使用频率
- 操作系统详细信息（类型、版本）
- 设备配置
  - 目标设备、启动类型、启动设备类型、TrustZone状态（启用/禁用）、连接类型（USB、UART等）
  - 信任配置方法（设备HSM、EL2GO）、调试认证（启用/禁用）、签名提供商（启用/禁用）、调试探针类型

尽管上面清楚列出了收集的数据类型，我们明确确认以下内容不会被收集：

- 个人身份信息（PII），如姓名、电子邮件地址或联系方式
- 用户生成的内容，包括项目文件、源代码或存储在用户设备上的任何数据
- 位置数据或行为追踪不在本工具的使用范围内。
- 敏感项目信息（例如，熔丝、加密密钥、文件路径或文件名）

所有收集的数据都是非个人的，不与第三方共享，仅用于改善用户体验和指导功能开发。

Restricted data ... : 受限数据可以从 Secure Provisioning Tool home page下载, 但是，获取此软件包的前提是签署 Non-Disclosure Agreement（NDA 保密协议）。其中，数据包含信任配置固件。这些数据是从 ZIP 文件中安装的。SEC 首先验证所选数据是否与当前工具兼容，如果兼容，则将数据复制到用户主目录中的子文件夹中。要开始使用数据，请重新启动 SEC。

Use restricted data from directory ： 允许控制是否使用受限数据。只有在为所选处理器安装了受限制的数据时，才启用复选框。

5.1.4 Workspaces

SEC 生成的所有文件都存储在一个名为 workspace 的专用文件夹结构中。

workspace 是一个实用的概念，用于操作使用不同密钥集签名的多个板卡、设备、或可执行文件。建议为每个项目创建一个 workspace。

workspace 总是为特定的设备家族（处理器系列）创建。一旦创建，它只能用于修改属于该系列设备的配置。

有两种类型的 workspace：

- 开发 workspace 在常规工具操作期间使用。
- 制造工作空间在制造过程中使用。更多详情，请参阅 Manufacturing workflow。

要创建一个 workspace，在 菜单栏 里选择 **File > New Workspace ...**，**File > Import Manufacturing Package ...**，或者 **File > Import Workspace ...**。更多详情，请参阅 New workspace。

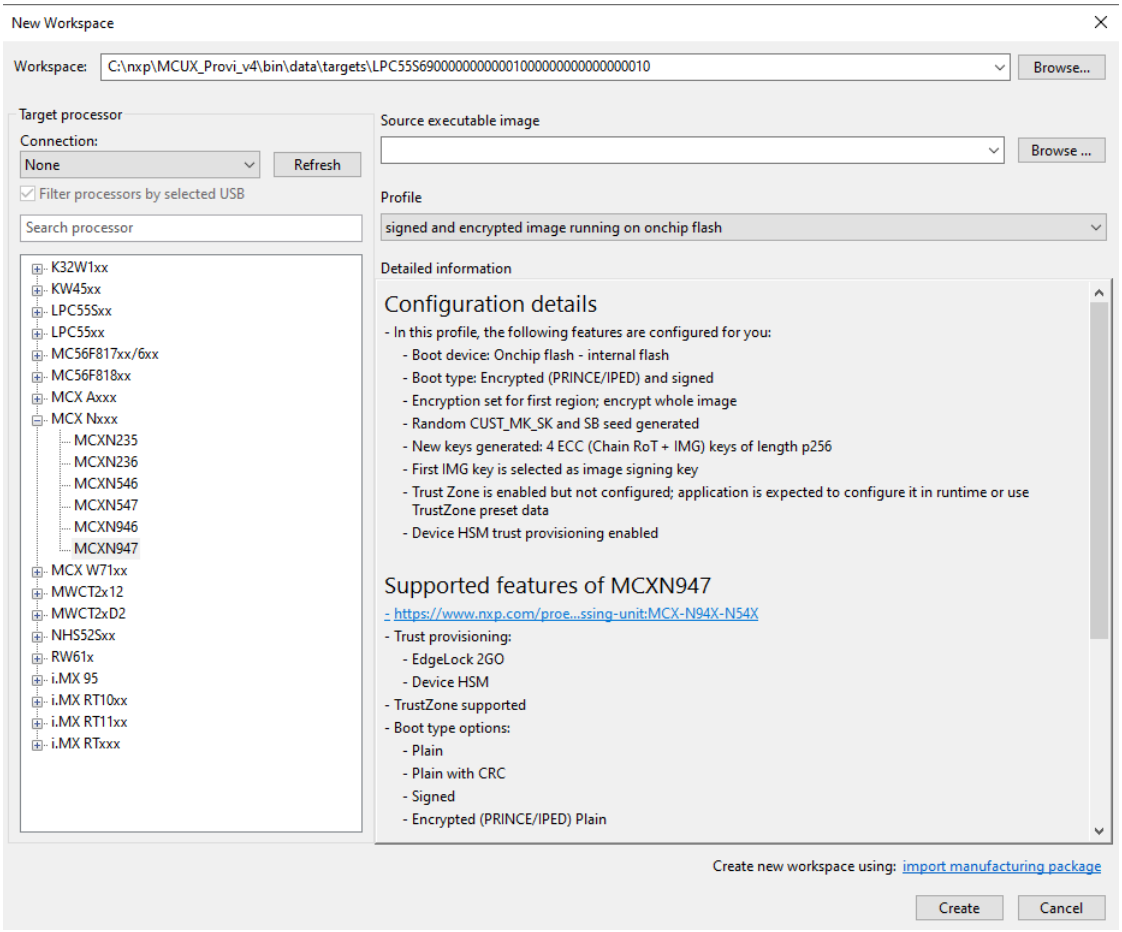

Figure 21. 创建一个 **workspace**

想要切换到不同的 workspace，可以从 菜单栏 中选择 **File > Select Workspace ...**，然后从 **Open Workspace** 对话框中选择。另一种打开现有 workspace 的方法是双击文件资源管理器中适当的 `settings.sptjson` settings.sptjson 。它打开具有给定 workspace 的工具。

想要切换到最近使用的 workspace，可以从 菜单栏 中选择 **File > Recent Workspaces**，并从列表中进行选择。

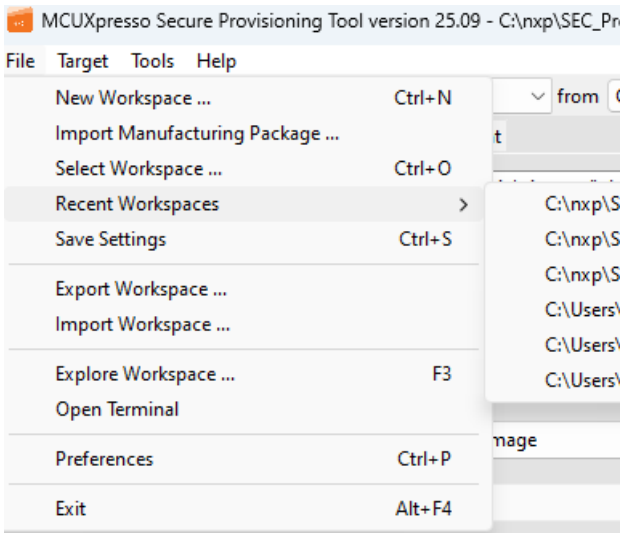

Figure 22. 选择一个最近使用的 **workspace**

每个创建的 workspace 都包含多个子文件夹。其中一些是特定于具体设备家族的。对于新的 workspace，大部分子文件夹都是空的；文件由用户稍后生成或添加。

- `backups`：在导入或生成新密钥后备份旧密钥/证书
- `bootable_images`：中间和最终 bootable image（nxpimage 输出）。nopadding 数据从地址 IVT 开始，而常规数据包含从启动设备起始地址开始的所有内容。
- `configs`：生成的配置文件，例如 OTFAD/IEE 配置文件（YAML）、BEE 用户密钥配置文件（YAML）以及相应的生成（BIN）文件。
- `crts`, `keys`：生成的证书及其对应的密钥。
- `dcd_files`： build image 步骤中包含的 DCD 文件。
- `debug_auth`： 调试由该工具生成的身份验证文件、用于生成证书的配置文件 (YAML)、证书请求 (ZIP)、证书（DC和ZIP）以及身份验证脚本。
- `el2go`： EdgeLock 2GO 信任配置文件
- `gen_bee_encrypt`： 在 build image 步骤中为 XIP 加密启动类型创建的 BEE 用户密钥文件。该密钥用于在写 image 文件步骤中烧写SW_GP2/GP4 fuses。
- `gen_hab_certs`： 输出超级根密钥和其哈希值（nxpcrypto 输出）。该表是与 bootable image 文件一起编程的。哈希值是烧写在 Fuse 中的。
- `gen_hab_encrypt`： 由 nxpimage 工具生成的 DEK 密钥文件。在写 image 文件期间使用 DEK 密钥文件为加密的 HAB 启动类型生成密钥 blob。
- `gen_scripts`： 用于工具操作的临时脚本。
- `hooks`： Hook 脚本，允许自定义构建、写入、或制造过程。带有前缀“pre”的 hook 在脚本执行之前执行。名称中带有“context”的 hook 在定义环境变量后在脚本的开头执行。它们可用于重新定义环境变量。一般 hook 文件在脚本执行过程中会被多次调用。更多相亲，请参阅 Build image, Write image, Manufacturing Tool。
- `logs`： 日志文件；log.txt 包含日志视图的内容/历史记录该文件夹还包含来自 SPSDK 命令行应用程序和制造过程的日志。
- ***root folder***： 包含以下内容：
  - `settings.sptjson`文件中该工具的上一次配置。
  - 构建和写入脚本。
  - 构建和写入 JSON 文件，包含用于生成构建和写入脚本的所有参数。
- `source_images`： 主要指向存储用户提供的镜像文件的文件夹。如果需要镜像文件格式转换，也用于指向镜像文件。

- `trust_provisioning`： 信任配置文件（仅当处理器支持信任配置时才支持）
- `trustzone_files`： elftosb 在 bootable image 文件生成步骤（nxpimage 输入）中使用的 TrustZone-M 配置文件（YAML、JSON 或 BIN）。

5.1.4.1 New workspace creation

要创建 workspace，请选择 **File>New Workspace**。可以在 **New Workspace** 对话框中选择以下选项：

- **Workspace path**： 创建 workspace 的文件夹；请使用空的或者不存在的文件夹。
- **Connection**： 应在新 workspace 中使用，用于与处理器通信的连接。如果未选择连接，则会使用处理器的默认连接创建一个新 workspace。
- **Refresh**： 更新检测到的连接列表。
- **Filter processors by selected USB**： 选中并选择 USB 连接后，设备树将按 USB VID/PID 标识的设备系列进行筛选
- **Search processor** : 从处理器树中筛选处理器。可以使用处理器、系列或主板名称的子字符串。
- **Processor tree**： 显示符合上述筛选条件的处理器。处理器按高级处理器类别和系列分类整理并放置在文件夹中。
- **Source executable image**： 选择输入的可执行应用程序镜像文件。关于输入 image 格式的更多详情，请参阅 Source image formats。
- **MCUboot bootloader image**： 选择输入的MCUboot引导加载程序镜像。更多关于MCUboot的详情，请参阅 MCUboot workflow。
- **Profile**： 将应用于新 workspace 的预定义设置。 建议使用默认配置文件开始开发并验证纯镜像是否有效。对于安全配置文件，“快速修复”将应用于配置，因此默认情况下不会出现错误。 可以创建自定义配置文件，请参阅 创建配置文件。
- **Detailed information**： 配置详情显示有关根据配置文件选择完成的 workspace 预配置的信息。“支持的功能”部分列出了该工具支持的处理器功能。

5.1.4.2 Sharing and copying workspaces

建议将所有使用的文件存储在 workspace 中。`settings.sptjson` 文件包含了相对于 workspace 根目录的所有路径，所以如果您在另一台计算机上打开该设置，仍然可以重新生成所有脚本。 workspace 可以是另一个项目的一部分。父文件夹的路径存储为相对路径，但其他文件夹的路径是绝对路径。建议使用 **Export Workspace** 对话框来共享 workspace。

如果必须在另一台计算机上直接执行脚本，它会使用环境变量来指定 SEC 安装目录和 workspace。这些环境变量可以在外部指定， 如果没有指定，则使用默认值。workspace 是自动检测的，只有在脚本复制到 workspace 之外时，环境变量才必须指定。

可以使用 **Export Workspace** 对话框来导出 workspace，从 菜单栏 中选取 **File > Export Workspace ...**。该对话框显示将当前 workspace 导出为带有可选 AES 加密的 ZIP 文件的选项。

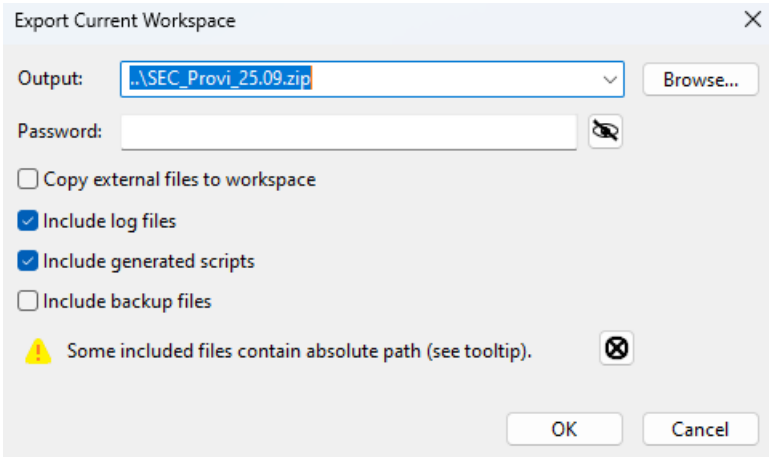

Figure 23. 导出一个 **Workspace**

Export Workspace 对话框允许选择以下选项：

- **Output：** 要创建的输出 ZIP 文件的路径。此操作将覆盖任何现有文件。
- **Password：** ZIP 文件加密的可选密码。如不需加密则留空。
- **Copy external files to workspace：** 此选项可将所有外部文件（位于 workspace 目录之外的文件）复制到导出的 workspace 中。工具提示中显示所有外部文件的列表。所有文件都将保存在 **source_images** 文件夹中。文件名冲突会自动解决，以确保没有文件被覆盖。当前workspace和设置不受影响。该操作仅影响 ZIP 压缩文件。如果有任何文件不存在，则该选项不可用。
- **Include log files：** 此选项可在导出的 workspace 中包含所有日志文件。
- **Include generated scripts：** 此选项可包括所有生成的脚本。脚本包含一个环境变量来指定 SEC 安装目录的 workspace，这是一个绝对路径。可能仍需要重新生成脚本。
- **Include backup files：** 包括来自“备份”子文件夹的文件。

当在包含的文件中检测到一些绝对路径时，会显示警告消息。这些文件的列表以工具提示的形式呈现。

5.1.4.3 Import workspace from the ZIP

要导入制造包，请从 菜单栏 中选择 **File > Import Manufacturing Package ...**。要导入 workspace 存档，请从菜单栏中选择 **File > Import Workspace ...**。

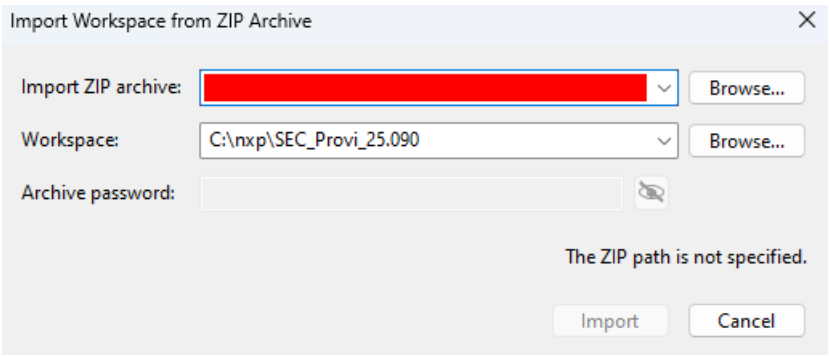

Figure 24. 导入制造包或 **workspace** 存档对话框

- **Import ZIP archive：** ZIP workspace 档案的路径。
- **Workspace：** 创建导入 workspace 的文件夹；建议使用空的或者不存在的文件夹。

- **Archive password**： 加密档案的 ZIP 密码。仅当检测到加密时才启用此字段。

该对话框允许从 ZIP 档案和制造包中导入 workspace。在这两种情况下，workspace 都会导入到新目录（或空目录）中。对于 workspace 导入，该工具支持从旧版本导入，但对于制造包，该工具和制造包版本必须相同。如果 ZIP 文件已加密，请输入正确的密码。更多有关制造流程的详情，请参阅 Manufacturing operations。

5.1.4.4 Profile creation

可以使用 `settings.sptjson`. 从任何 workspace 创建配置文件。配置文件是与配置文件相关的 workspace 设置的子集。配置文件必须放置在 `<install_folder>/bin/_internal/sample_data/targets/<processor name>/configuration_profiles` 文件夹中，以便在 **New Workspace** 对话框中作为配置文件选择的选项提供。要创建有效的配置文件，请手动指定以下 JSON 字段：

- **is_profile**： 标识配置文件设置的标志，必须将其设置为 **true**。
- **profile_name**： 配置文件的名称；该名称显示在工具中并且必须是处理器唯一的
- **profile_description_introduction**： 自然语言的配置文件描述。它显示在格式化的配置文件描述之前。该字段为可选。
- **profile_description**： 应包含配置文件设置的信息。必须将其指定为信息列表。

应从配置文件设置中删除的字段：

- **write_image_settings**、**keys_management_settings**、和 **connection** 对于新 workspace 没有任何意义。
- 从 **build_image_settings** 中，删除包含不应分发的机密的所有字段（例如 **dek_key**、**keyblob_key_id_int**、**sbkek**、**sb_seed**）或与新 workspace（**source_image_path**、**dcd_path**、**ele_firmware_path**、**xmcd_path**、**tz_path**、**script_updated**、**updated**）无关的字段

所有字段的描述可以在工具安装文件中分发的 JSON 模式 **settings_schema_##_##.json** 中找到。

5.1.5 Toolbar

通过 工具栏，您可以快速选择基本设置。相同的命令也可在 **main menu > Target** 中找到。

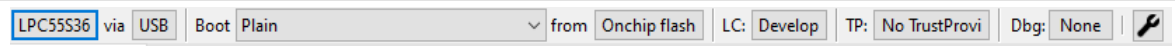

Figure 25. 工具栏

- **Processor**： 显示所选的处理器。单击按钮切换处理器。您可以切换到当前设置兼容的同一设备系列的处理器。要从不同的系列中选择处理器，请创建一个workspace。
- **Connection (via)**： 选择到目标的连接。该版本支持UART、USB-HID、SPI和 I2C连接。单击该按钮自定义连接详细信息。有关更多详情，请参阅 Connection。
- **Boot mode**： 选择启动类型。该列表取决于当前所选处理器的功能。
- **Boot memory (from)**： 单击按钮打开启动内存配置。更多详情，请参阅 Boot memory configuration。
- **Life cycle**： 允许选择处理器生命周期。点击按钮从特定于处理器的生命周期中进行选择；选择对话框显示每个选项的简短说明。
- **Trust provisioning type**： 允许选择信任配置类型，并启用它进行信任配置操作。更多详情，请参阅 Trust provisioning。
- **Debug probe**： 允许选择连接到计算机的调试探测器针；更多详情，请参阅 Debug Probe Selection dialog。
- **Quick fix**： 解决构建页面上显示的问题。并非所有问题都可以通过“快速修复”求解器来解决，因为并非所有问题都有单一的确定性解决方案。在解算器进行任何更改之前，系统会提示用户保存设置。快速修复完成后，将显示用于修复问题的操作列表；这些操作也会显示在日志中。建议仔细审查所有更改，以确保解决方案符合预期。

5.1.6 Connection

Connection 对话框允许您选择与目标处理器进行连接并测试它。

该对话框可以从 菜单栏 中的 **Target > Connection** 或 工具栏 访问。

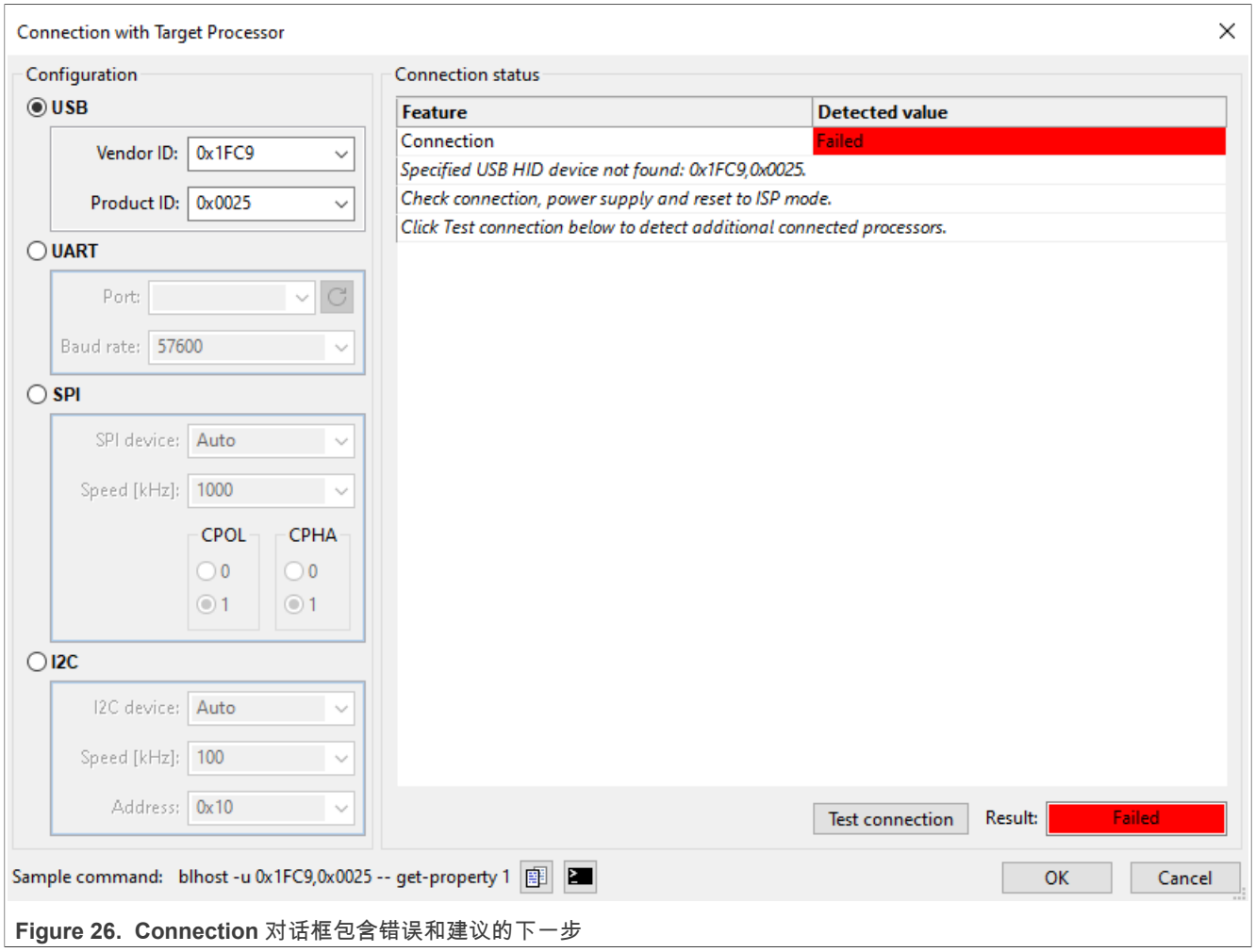

Figure 26. Connection 对话框包含错误和建议的下一步

它包含以下选项（如果处理器支持）：

- **USB**： 指定 USB 连接到指定的供应商ID/产品ID对。
- **UART**： 通过指定的端口和波特率指定 UART 连接。处理初始 ping 时，引导加载程序会自动检测波特率。这意味着在选择新的波特率后必须重置目标处理器。
- **SPI/I2C**： 可以使用基于 LIBUSB 接口的 SPI 或 I2C 方式与处理器连接：
  - MCU-Link Pro
  - LPC-Link2 在一些 EVK 板上有 LPC-link2，但是要通过 SPI 连接，必须使用跳线连接处理器。

在启动安全配置工具之前，有必要从上面列出的产品页面下载并安装 USB 驱动程序。可以配置以下连接参数：

- **SPI/I2C device**： 设备通过 USB 路径指定。连接对话框中的默认值是“自动”，这意味着如果只有一个设备连接到计算机，它将被自动选择。关于 USB 路径格式的详细信息可以在 SPSDK 文档中找到。
- **Speed [kHz]**： 通信时钟频率（单位：kHz）。
- **CPOL, CPHA**： 信号极性和相位；更多详情，请参阅 SPI 规范。
- **Address**： I2C 设备地址。

Crystal frequency [kHz]： 外部晶振的频率。用于配置与处理器的通信速度。目前它仅用于`lpcprog`。

使用 **Test Connection** 功能验证设备可以用给定的配置来正确访问。为了确保使用 **Test Connection** 成功检测处理器，请确保以下内容：

- 板卡已正确通电
- 板卡正确配置为 ISP (In-System Programming) 模式
- 板卡已通过 USB OTG 接口或 UART 连接

连接对话框检测以下参数：

- **Connection**： SPI/I2C 连接的通信设备（USB 或串行端口）或 USB 路径的选择状态
- **Mode**： Bootloader 或者 flashloader 通信模式。
- **Processor**： *Match*如果连接的处理器与所选处理器匹配则*No match*，否则No match。当 SEC 工具与错误的板通信时，此功能允许发现错误。此功能并非 100% 可靠，因为没有足够的信息来识别每个处理器。
- **Life cycle**： 在连接的处理器中检测到的生命周期。
- **Silicon version**： 仅当连接的处理器为旧版本时才显示。当检测到旧版本时，连接测试结果为“失败”状态。工具提示中提到了与此版本相关的所有已知问题。

可能会出现以下连接结果：

- **Not tested yet**： 使用 **Test connection** 按钮运行测试。
- **OK**： 已成功建立连接。
- **FAILED**： 连接测试失败。更多详情，请参阅 连接状态。有关故障的更多详情，可以使用详细模式的 SEC 并在控制台视图中查找。

连接对话框的底部有“Sample blhost/sdphost/lpcprog/nxpuuu command”，允许使用指定参数运行相应的 SPSDK 命令行工具。使用第一个按钮将带有参数的命令复制到剪贴板中。使用第二个按钮可以打开一个可以执行命令的终端。

5.1.7 Boot memory configuration

Boot Memory Configuration 对话框允许选择和配置引导设备。该对话框包含以下配置部分：

- **Boot memory type**： 这部分允许选择启动存储器类型，以及可选的实例。
- **Predefined template**： 该部分允许选择启动存储器配置模板。该列表特定于每种存储器类型，并包含 NXP 评估板上可用的存储器。打开启动内存配置对话框后，该选项包含与当前配置匹配的值（如果值未更改），或者为空，下拉菜单中的第一项是评估板上使用的内存（如果适用）。标记为已验证的内存配置模板已在硬件上进行了测试。
- **User configuration**： 此部分允许将配置加载或保存到所选文件。它对于将配置重用于另一个项目或与同事共享配置可能很有用。
- **Protected area**： 该部分允许指定 SEC 工具不得更改的存储区域。如果该工具尝试擦除或修改选定的存储区域，则会显示确认对话框。它对于保护启动存储器中的自定义数据可能很有用。指定评论/原因，因为它将显示为确认消息的一部分。
- **Boot memory configuration parameters**： 内存的配置，这些参数特定于每种内存类型。
- **Comment**： 启动存储器的描述，包含应用预定义模板的信息
- **Test the configuration**： 此按钮用于测试所连接的处理器/板的当前内存配置。该测试包括两个步骤：读取测试（是否可以读取内存）和引导加载程序测试，该测试使用 blhost list-memory 命令验证内存配置。
- **Convert to Complete FCB**： 此选项仅适用于 SPI NOR。该按钮允许将简化的 SPI NOR 配置转换为完整的“内存控制块”（更多详情，请参阅 SPI NOR）。

5.1.7.1 SPI NOR

FlexSPI NOR flash 有两种配置方式：

- 通过使用基于闪存加载器/ROM 的简化配置（两个 32 位字）
- 使用完整的闪存控制块（FCB二进制，512字节）

对于简单的配置，用户可以修改对话框中建议的值。要从 SEC 工具里创建自己的 FCB，先做 一个简单设置，并使用右下角的 **Convert to complete FCB** 按钮。该操作可打开 **Convert To Complete FCB**，允许使用连接的处理器将现有的简化配置转换为完整配置。简化的设备配置被写入 SPI NOR 闪存设备并读回，同时保留原始数据。当转换过程完成时， 它会在指定的 FCB 文件路径下创 建一个 `.bin` 文件。

Convert To Complete FCB 对话框提供了复选框，用于将创建的 FCB 二进制文件用作 SPI NOR/**user FCB file**。如果未选中，则只完成 FCB 转换。

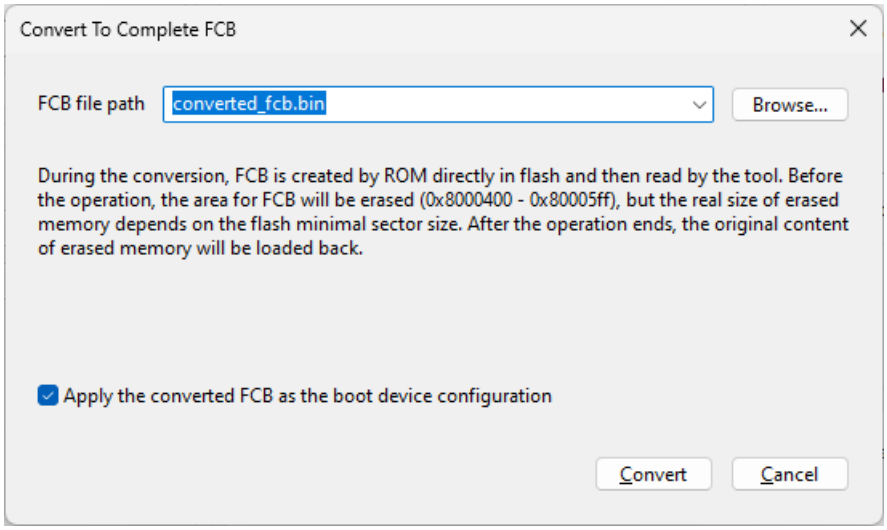

Figure 27. Convert to Complete FCB 对话框

还可以在 MCUXpresso IDE 中创建 FCB，方法是将 FCB 组件添加到外设工具的外设驱动程序中，在其中可以指定完整配置。

当在引导设备配置中指定完整的 FCB 时，用户可以指定两个单独的 FCB 文件。运行时 FCB 用于处理器启动期间的 flash 配置。FCB 文件的路径，该文件将用于 flash 配置以对应用程序进行编程。

5.1.7.2 OnChip RAM

SEC 工具允许创建在内部 RAM 中执行的 image，这可能对芯片 （重新） 配置有用。对于该启动存储器，写入脚本将应用程序写入处理器并中间启动它。如果芯片受到保护，则通过 SB 文件编写并启动应用程序。

注意： SB 文件还可用于恢复 flash。

5.1.7.3 Serial Downloader

串行启动提供了一种将启动镜像下载到芯片并执行该镜像的方法。在这种启动模式下，主机 PC/设备可以使用串行下载器协议与 ROM 引导加载程序通信。SEC 工具允许在内部或外部 RAM 中执行镜像。将该应用程序分发到无闪存设备上可能很有用。

写入脚本包含向处理器提供安全资产以及在 RAM 中加载和执行应用程序两部分。在生产过程中，应用程序不会被加载。

5.1.8 Trust provisioning

Trust Provisioning 对话框允许选择特定于处理器的信任配置类型，并将其启用为信任配置操作。

SEC 工具支持以下信任配置类型：

- **Device HSM** — 使用存储在处理器中的密钥加密。对于该系列的所有处理器都是相同的。
- **EdgeLock 2GO** — 密钥存储在 NXP 云中的 EdgeLock 2GO 服务器上。

5.1.8.1 Device HSM provisioning

设备 HSM 配置有两种类型：

- ROM 直接支持设备 HSM。在这种情况下，不需要额外的固件。
- 以下用例需要额外的固件：
  - i.MX RT5xx/6xx，其中设备 HSM 是可选的。有关如何启用的详情，请参阅 Device HSM provisioning。 该固件以受限数据包形式分发（请参见 **main menu > File > Preferences** 进行安装）。
  - RW61x 处理器，其中设备 HSM 对于安全启动类型是强制的，并且在 ROM 中得到部分支持。该固件分布在工具数据中。

要成功构建启用设备 HSM 的 image，必须在 Connection 对话框中连接并选择处理器。仅当已经存在使用设备 HSM 构建的 image 并且设置相同时才不需要。

5.1.8.2 EdgeLock 2GO trust provisioning

EdgeLock 2GO 参数允许配置对 EdgeLock2GO 服务器上资产的访问。

EdgeLock 2GO 支持两种流程：

- **EdgeLock 2GO with device ID**：安全对象在配置过程中从 EdgeLock 2GO 云服务器检索
- **EdgeLock 2GO per product type**：安全对象从服务器批量数据库中检索，无需连接到云服务器即可执行配置（离线）

其他配置控制：

- **API key** - 用户特定的十六进制密钥，授予对服务器的访问权限。更多详情，请参阅 API key to access EdgeLock 2GO server 。
- **Device group ID**- EdgeLock 2GO 服务器上包含所有安全资产的设备组的标识号。在 EdgeLock 2GO 服务器上，可以在 **Devices > MCU & MPU** 下找到。
- **12NC**- 硬件产品唯一标识。在 EdgeLock 2GO 服务器上，可以在有关设备组的信息中找到该信息。
- **Secure objects address**- 配置期间存储安全对象的闪存中的地址。
- **Server URL**- EdgeLock 2GO REST API 服务器的 URL。使用空字符串应用默认值。
- 使用 **Test** 按钮可以验证与服务器的连接。
- **Product Batch Database** 面板允许从 EdgeLock 2GO 服务器生成和下载安全对象数据库，并将其用于离线按产品 配置流程。更多详情，请参阅 Create a database containing security objects for each product type of EdgeLock 2GO。
  - **#devices** - 批处理数据库中要包含的处理器数量
  - **#batches** - 要生成的批次数据库数量

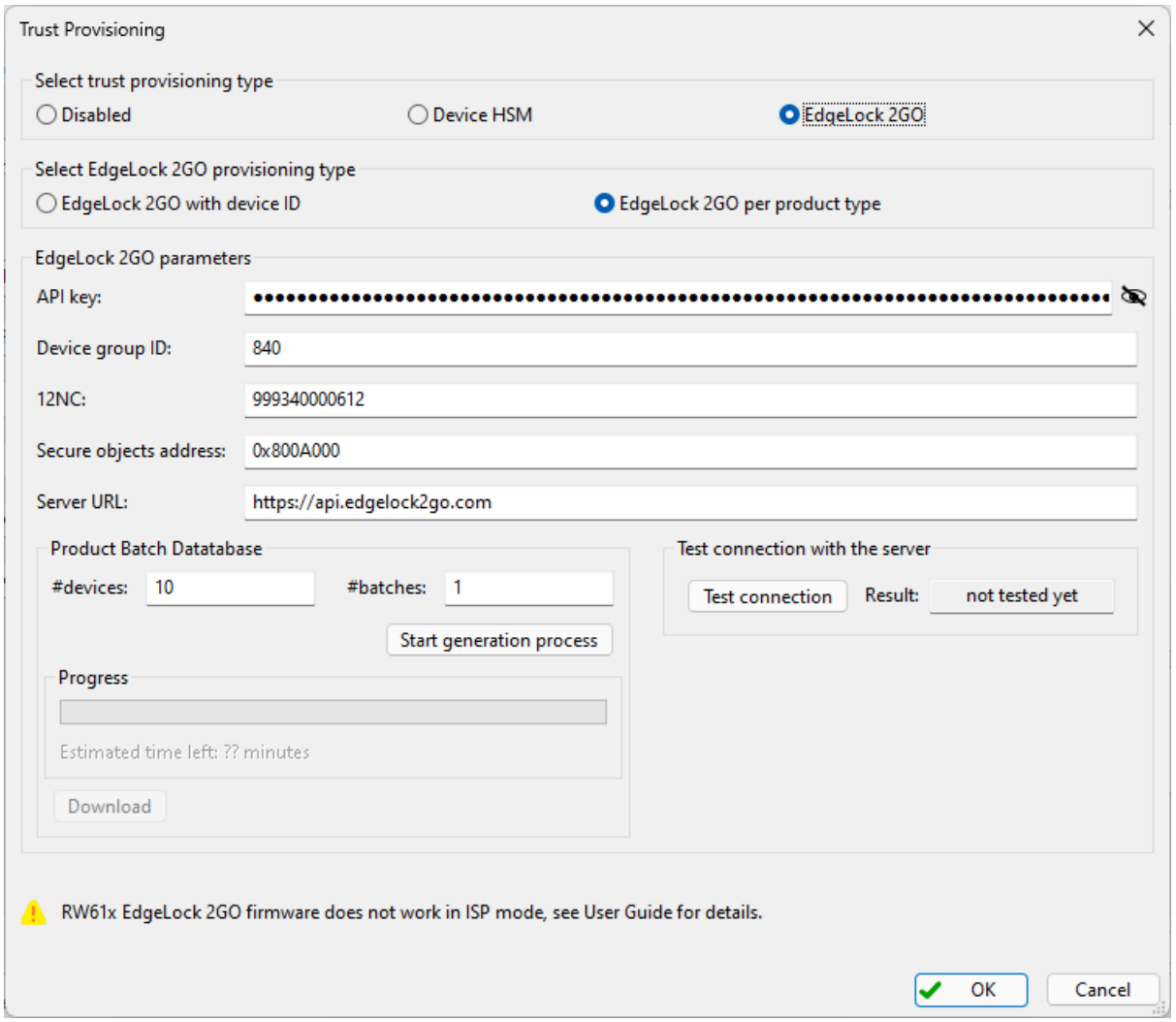

Figure 28. EdgeLock 2GO 信任配置

5.1.9 Debug Probe Selection dialog

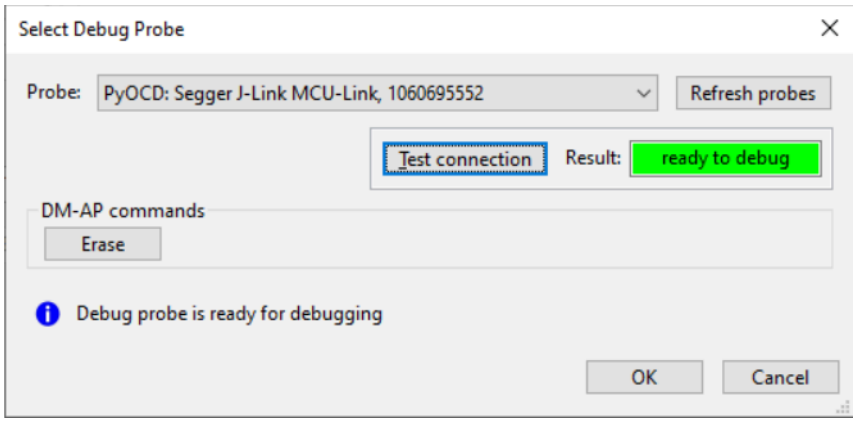

Figure 29. 调试探测器选择对话框

调试探测器选择对话框允许：

- 选择影子寄存器的探测器（仅适用于通过调试探测器初始化 fuse 影子寄存器的处理器）
- 使用带有给定探测器的测试连接
- 擦除处理器 flash（如果处理器通过调试探测器 API 支持）

打开对话框后，将检测到连接到计算机的探测器。稍后可以使用 **Refresh probes** 按钮再次重新扫描探测器。

选择探测器后，选择将存储在 workspace 设置中，包括硬件 ID（序列号）。如果连接了不同的板，请更新选择。在打开对话框期间，如果未找到所选探测器，该工具会自动更新硬件 ID。

Test connection 按钮提供有关是否可以为所选调试探测器启动调试的信息。可能的结果是：

- **ready to debug** - 如果调试探测器和处理器已准备好进行调试。
- **no debug** - 如果已建立与调试探测器的连接，但无法调试处理器；在这种情况下，请确保调试器已正确连接并且处理器正在运行（不是 ISP 模式）。
- **FAILED** - 如果与调试探测器的连接失败。

Erase 按钮允许擦除内部 flash（批量擦除）。更多详情，请参阅处理器参考手册。

5.2 Build image

在 **Build image** 视图中，您可以将原始 image 程序文件转换为与所选设备兼容的可启动image 格式。

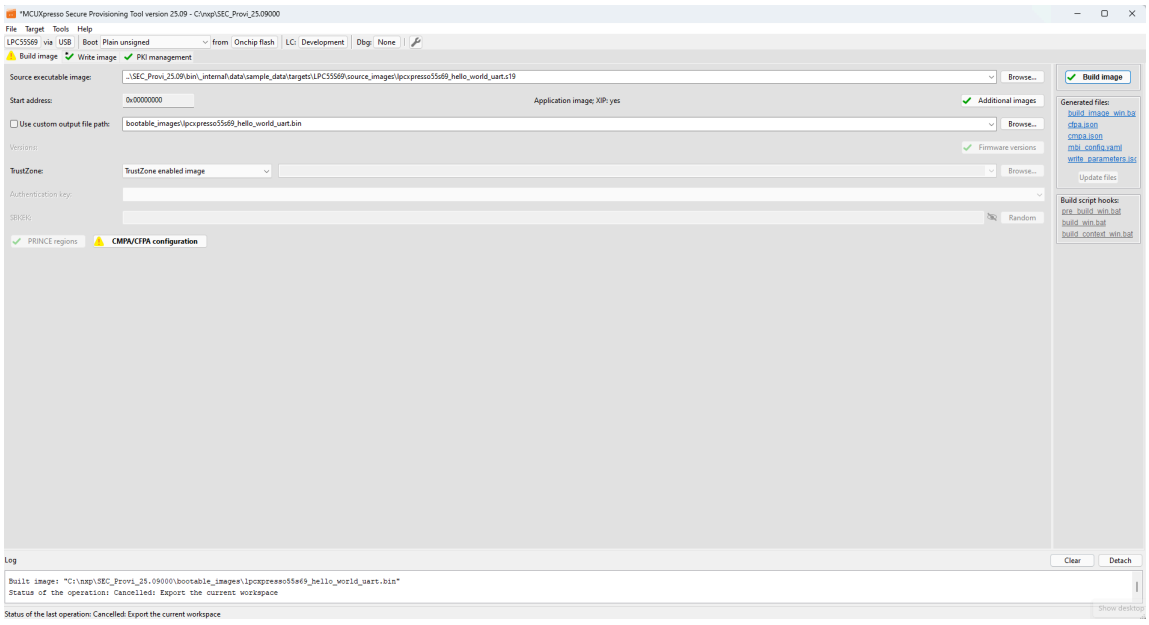

Figure 30. 生成 **image (LPC55Sxx)**

5.2.1 Controls on the build image view:

Source executable image： 选择输入可执行文件。关于输入 image 格式的更多详情，请参阅 Source image formats**Start address**： image 的基址。它只能编辑二进制 image。对于 ELF 和 S-Record 和 HEX 文件，它会自动检测。

Application/Bootable image and XIP： 提供了关于所选源 image 的以下信息:

1. 是应用程序还是可启动的 image。

2. 所选源 image 是否是 XIP，并将从存储它的启动设备执行。如果不是，则必须在执行之前将 image 复制到 RAM 中。该信息来自 image 的起始地址，并与所选处理器的内存地址进行比较，因此，如果所选 image 与所选处理器不匹配，则结果可能不正确。

Additional images： 打开附加用户/OEM image 的配置对话框。 请参阅 Additional images 。

Use custom output file path： 生成的可启动 image 的名称及其位置。如果没有指定，工具根据输入来命名 image。文件扩展名特定于处理器和启动类型，它是用于可启动 image 的 BIN 或用于安全二进制胶囊的 SB。

Image version： 可引导 image 的版本。它用于双程序启动。首先启动高版本的程序。

Dual image boot： 打开双程序乒乓启动配置窗口。image 可以写入 Flash 的基空间（image0）和/或重映射空间（image1），它们各自都有自己的版本。然后 ROM 使用程序版本来选择最新的程序来启动。如果最新的程序启动失败，则使用旧程序重新启动。

Firmware versions： 打开固件版本 （安全和非安全） 的配置对话框。固件版本允许设置防回滚保护。更多详情，请参阅 Firmware versions。

XMCD： 允许启用外部存储器配置数据功能。对于需要大容量 RAM 的综合性或功能丰富的应用程序，需要 XMCD （片上 RAM 是不够的）。 可以提供 YAML 或 BIN 配置文件，也可以在 XMCD configuration dialog 中准备 XMCD 简化配置 \（有关详细信息，请参阅 **-xmcd-cfg** CLI 参数的描述\）。

DCD (Binary)： 选择哪些 DCD 数据应该包含在可启动 image 中。只有当源 image 包含 DCD 时，才可以使用选项 **From source image**。DCD支持包括 SDRAM 在内的初始化配置。 MCUXpresso 配置工具可以生成兼容格式的 DCD。如果目标处理器不支持 DCD 文件，则禁用复选框。更多详情，请参与 Creating/Customizing DCD files。

TrustZone： 允许您启用 TrustZone 功能。可以进行以下选择：

- **TrustZone disabled image** — 禁用 TrustZone。某些处理器可能不支持此选项。
- **TrustZone enabled image** - 使用处理器中预设的默认配置启用 TrustZone。
- **TrustZone enabled image with preset data** — 启用从指定位置加载自定义 TrustZoneM 数据的 TrustZone。支持 JSON 和 BIN 文件。JSON 数据可以在配置工具的 TEE 工具中生成和导出。BIN 文件是 由 elftosb 程序创建的。更多详情，请参阅 TrustZone configuration file。

Authentication key： 使用指定的密钥对 image 进行签名。该密钥还可以用于 SB 文件的身份验证。此选项仅适用于已验证和加密的启动模式，并提供在 **PKI management** 视图中生成的密钥选择 。

Key id： keyblob 加密密钥标识符用于加密 (AHAB) 启动类型

AHAB/HAB encryption algorithm： 选择加密 (HAB) 或加密 (AHAB) 启动类型中使用的 AHAB/HAB 加密算法。

Key source： 为程序 image 签名的密钥源。

User key： 对于 OTP 密钥源，主密钥用于派生其他密钥。对于 PUF KeyStore，用户密钥用于签名程序 image。仅适用于普通签名的启动类型。

SBKEK, SB3KDK, or CUST_MK_SK： 密钥用作密钥加密密钥来处理 SB 文件。仅适用于安全启动类型。对于 RT5xx/6xx，仅当密钥源为 KeyStore 时才启用它。对于 LPC55Sxx 设备，密钥存储在处理器生命周期中只初始化一次，在此之后，SBKEK 中的任何更改都将导致 SB 文件加载到处理器中失败。更多详情，请参阅 PFR and PUF KeyStore。OEM 寻求用于随机创建SB文件的十六进制密钥 \（使用 CUST_MK_SK 密钥\）。

Configuration dialogs： 构建镜像视图上有以下配置对话框：

- **XIP encryption (BEE user keys)** 打 开 XIP 加密用户密钥配置对话框。选项仅对 *XIP ####*（*BEE ####*）### # 和 *XIP #####*（*BEE ####*） 启动类型有效。

- **XIP encryption (BEE OTPMK)**： 打开带有 OTP 主密钥的 BEE 配置窗口。该选项仅对 XIP 签名加密（BEE OTPMK）的启动类型启用。
- **IEE encryption**： 打开 IEE 加密配置对话框。该选项仅对 IEE 加密启动类型启用。
- **OTFAD encryption**： 打开 OTFAD 加密配置对话框。选项仅为 OTFAD 加密启动类型启用。对于 RT10xx 处理器，按钮名称为 XIP 加密 （OTFAD用户密钥）。
- **XIP encryption (OTFAD OTPMK)**： 打开带有 OTP 主密钥的 OTFAD 配置窗口。该选项仅对 XIP 加密（OTFAD OTPMK）身份验证启动类型启用。
- **PRINCE/IPED regions**： 打开加密的 PRINCE 区域配置窗口，该窗口允许指定将加密的 flash 区域。
- **OTP/PFR/IFR/BCA/FCF configuration**： 打开 OTP/PFR/IFR/BCA/FCF configuration。

5.2.2 Generated files

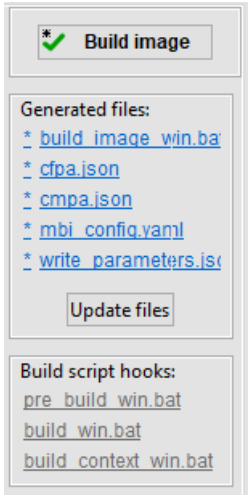

Figure 31. Build image 面板

生成面板中有一个 **Build image** 按钮，生成页面的右侧列出了生成的文件。使用 **Build image** 按钮来更新所有文件并执行生成脚本。如果生成脚本未更新或未成功执行，按钮上的图标将显示一个名为“dirty flag”的星号(*)。**Build image** 该页面中也显示相同的指示。

以下生成的文件将显示为可单击的链接，如果您单击这些文件，则会显示文件内容。如果磁盘上的文件未更新，则文件名以星号(*)开头。这可能是由于配置更改引起的。将鼠标光标移到 * 上可查看有关更改/差异的详细信息。

注意： 二进制文件或较大更改不支持差异。

此信息在后台以一定的延迟更新，因此可能需要几秒钟才能显示实际状态。在有关文件的信息更新之前，文件图标为 **?**（问号）。

Update files 按钮允许更新所有文件而不执行脚本。

对于 EdgeLock 2GO 使用案例，需要上传到 EdgeLock 2GO 服务器的每个文件都有一个图标。

5.2.3 Build script hooks

Build script hooks 显示在生成的文件下方的构建图像面板中。Hook 脚本在构建脚本执行之前或执行期间被调用。灰色标签表示该文件在 workspace 中尚不存在。单击后，将使用示例中的内容创建一个新文件并打开进行修改。如果文件存在，标签为蓝色，单击后将打开文件。更多详情，请参阅 Script hooks。

5.2.4 Source image formats

SEC 支持如下几种格式的源 image 程序格式：ELF，HEX，BIN 或 SREC/S19（S-record）。这些 image 程序格式会被统一为构建脚本所需的格式，这种转换是在 SEC 内部完成的（之前的构建脚本会被调用）。建议避免格式转换并使用构建所需的格式。

SEC 工具部分解析 image 以检索入口点，检测 image 是否为 XIP 并验证目标地址。在S19文件格式中，可以显式列出入口点，而对于其他格式，入口点是从中断向量表中检索的（因此中断向量表必须位于应用程序 image 的开头）。

对于某些处理器，MCUXpresso SDK 示例包含包含 FCB 配置的可启动 image。SEC 工具支持将可引导 image 拆分为多个部分（FCB、DCD、XMCD 部分），并重用这些部分来构建新的可引导 image。一旦可启动 image 被选中，解析器接受该 image，SEC 就会提供重用特定部件确认对话框，如果确认，则更新配置。

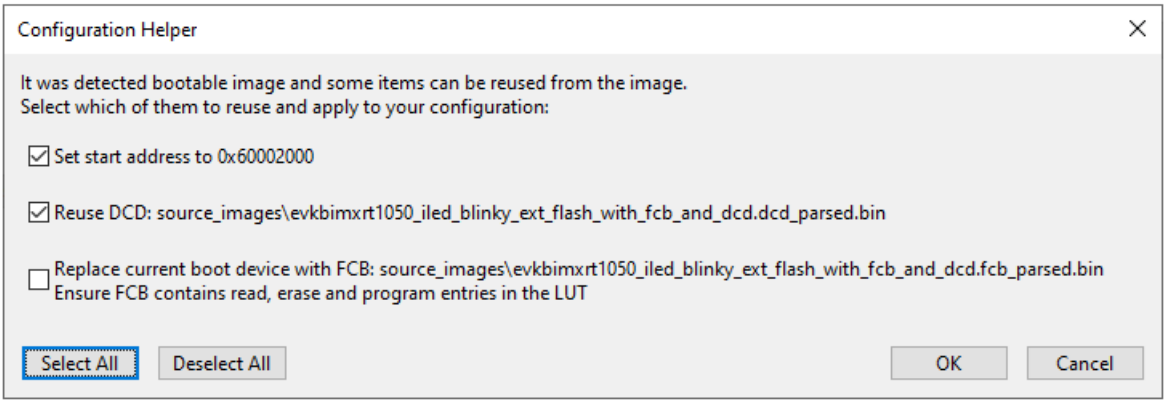

Figure 32. 配置助手

5.2.5 Additional images

该对话框允许用户指定最多 8 个用户 image，这些 image 将与应用程序一起写入启动存储器。对于 i.MX 9x 处理器，该对话框允许指定最多 16 个用户 image，因为处理器支持主要和辅助容器集。可配置选项（列）取决于 image 格式。它们对于 AHAB image 和其他处理器是不同的。

配置以表格形式出现，每一行代表一个用户 image。列代表每个 image 的可配置功能。每个功能的描述可以在工具提示中找到。除 i.MX 9x 系列之外的所有处理器都为应用程序可执行 image 保留了最后一个 image，该 image 会根据构建页面自动更新。除引导 image 外，任何 image 的顺序都可以通过右上角的向上/向下箭头更改。

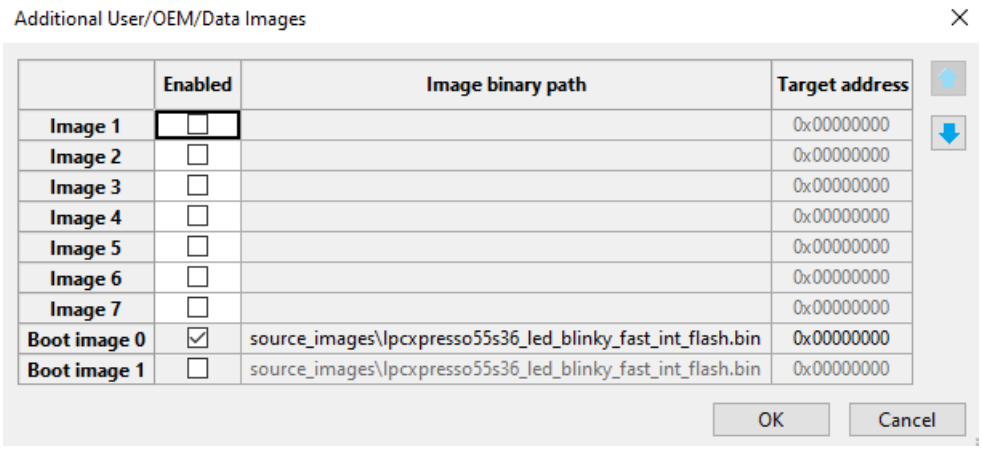

Figure 33. 额外的用户/**OEM image** 配置

附加 image 的典型用法示例：

- 为第二个 core 添加 image
- 为辅助 bootloader 添加应用程序 image
- 从外部文件添加二进制数据

5.2.6 Firmware versions

Firmware Versions 对话框允许配置防回滚保护。在 **Firmware Versions** 对话框中，指定 image 的安全和非安全固件版本以及用于防回滚保护的 CFPA 或 fuse 的值。

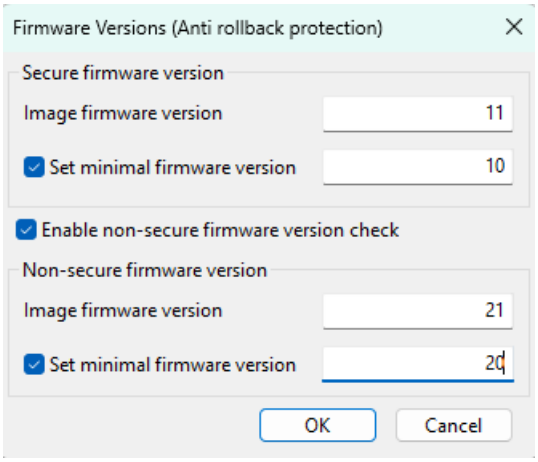

Figure 34. 固件版本对话框 **(LPC55S36)**

每个面板（安全固件版本和非安全固件版本）用于特定的固件版本。在 SB 文件或 AHAB image 上传期间，将根据 CFPA 中的值（上传的版本必须等于或大于 CFPA 中的适当值）处理器或 fuse 值处理器检查这些版本。安全固件版本指定由 ROM 引导期间检查的安全 image 版本。

指定非安全固件版本的值是可选的，可以启用或禁用。非安全固件版本面板仅对支持非安全固件版本的处理器可见。

Image 固件版本。此版本包含在可启动 image 中和/或在 SB 文件或 AHAB image 中进行检查。该值必须大于或等于最低固件版本。

设置最低固件版本。指定 CFPA 或 fuse 中存储的最低固件版本以启用防回滚保护。该值必须大于或等于最低固件版本。如果未指定该值，可以在 **OTP/PFR configuration** 对话框中指定，也可以使用默认值。

5.2.7 XMCD configuration dialog

编辑器允许为简化的 XMCD 自定义参数。这些参数特定于选定的 XMCD 内存接口，可以是：

- 用于 HyperRAM/APMemory 的 FlexSPI/XSPI 接口或
- SDRAM 的 SEMC 接口

当选择 FlexSPI/XSPI RAM 或 SEMC SDRAM 时，可以通过构建 image 视图上的 XMCD **Edit** 按钮访问 XMCD 编辑器对话框。

Import 按钮允许从 YAML 或二进制文件导入简化配置。**Reset to defaults** 按钮允许返回到默认设置。

5.2.8 TrustZone configuration file

MCU 中 TrustZone 及相关功能可以通过启动时应用程序头里的数据进行预配置，而不是 在应用程序代码里设置寄存器。MCUXpresso Config Tools 中的 TEE（Trusted Execution Environment）工具允许您导出 TZ-M 预设数据以供 SEC 使用。按照以下步骤修改现有的示例应用程序，导出 TZ-M 文件并将其添加到应用程序 image 中。

要创建、导出和导入 TrustZone 文件，请执行以下操作：

1. 打开一个 SDK 示例：
   a. 来自 MCUXpresso IDE：
      i. 在 **Quickstart** 面板中，选择 **Import SDK example(s)…**。
      ii. 选择要导入的示例。
      iii. 在 **Project Explorer** 中，打开导入的安全工程的上下文菜单。
   b. 来自 MCUXpresso 配置工具：
      i. 在开始时，选择 **Create a new configuration and project based on an SDK example or hello world project**。
      ii. 克隆一个启用 TrustZone（安全）的项目。
2. 打开 TEE 工具：
   a. 在 MCUXpresso IDE 中：
      i. 在 菜单栏 中，选择 **MCUXpresso Config Tools > Open TEE**。
   b. 在 MCUXpresso 配置工具中：
      i. 在 **Config Tools Overview** 中选择 **TEE** 工具。
3. 选择 **Security Access Configuration> Miscellaneous** ，在**Output type** 下拉框列表中选择 *ROM preset*。
4. 根据需要配置内存区域的安全策略（更多详情，请参阅 MCUXpresso 配置工具用户手册（桌面版）document GSMCUXCTUG
5. 在 菜单栏 中 ，选择 **File > Export > TEE Tool > Export Source Files**。
6. 在 **Export** 指定 JSON 文件下载文件夹，点击 **Finish**。
7. 从 *SystemInitHook(void)* 函数中删除 *BOARD_InitTrustZone()*。调用和位于主应用程序文件（例如，*hello_world_s.c*）中的 *tzm_config .h* 头文件包含语句。

或者，包括在 SEC 布局内的基本 TZ-M-preset JSON 数据也可以用作 TrustZone 预配置 进一步修改的起点模板。设备特定的模板文件在 `sample_data\targets\<processor>\trust_zone_template.json`

注意： TrustZone 模板包含所有具有默认预设值的寄存器/选项。因为模板中 SAU 和 AHB 被禁用，所以模板在使用前需要被自定义设置。

下载完 JSON 文件后，可以在 SEC 内导入它：

1. 在 SEC 的 菜单栏 中，选择 **File > Select Workspace ...**，并选择一个 workspace。或者，通过选择 **File > New Workspace ...** 创建一个 Workspace。
2. 在 **Build image** 视图中，将 **Boot type** 切换为*Signed*或 *Unsigned with CRC*。
3. 使用 **TrustZone pre-configuration** 下拉列表中选择 **Enabled (custom data)**。
4. 点击 **Browse** 导航到存储的 JSON 文件所在位置，选择 **Open** 以导入。

可以使用 YAML 格式而不是 JSON 将 TZ-M 预设数据导入 SEC。

5.2.9 OTP/PFR/IFR/BCA/FCF configuration

配置对话框允许配置：

- 一次性可编程 Fuse
- PFR 区域的 CFPA 和 CMPA 页面。
- IFR 区域的 ROMCFG/IFR 页
- Bootloader Configuration Area（BCA Bootloader 配置区域）
- Flash Configuration Field（FCF Flash 配置字段）

使用配置对话框来：

- 检查 SEC 准备的配置
- 从连接的处理器读取当前配置
- 自定义配置
- 锁定配置（此功能仅适用于一些 fuse 区域）

在 OTP（One Time Programmable）配置中，可配置项称为 **fuse**。在 PFR（Protected Flash Region）配置中，可配置项称为 **field**。对话框的内容取决于所选择的处理器。

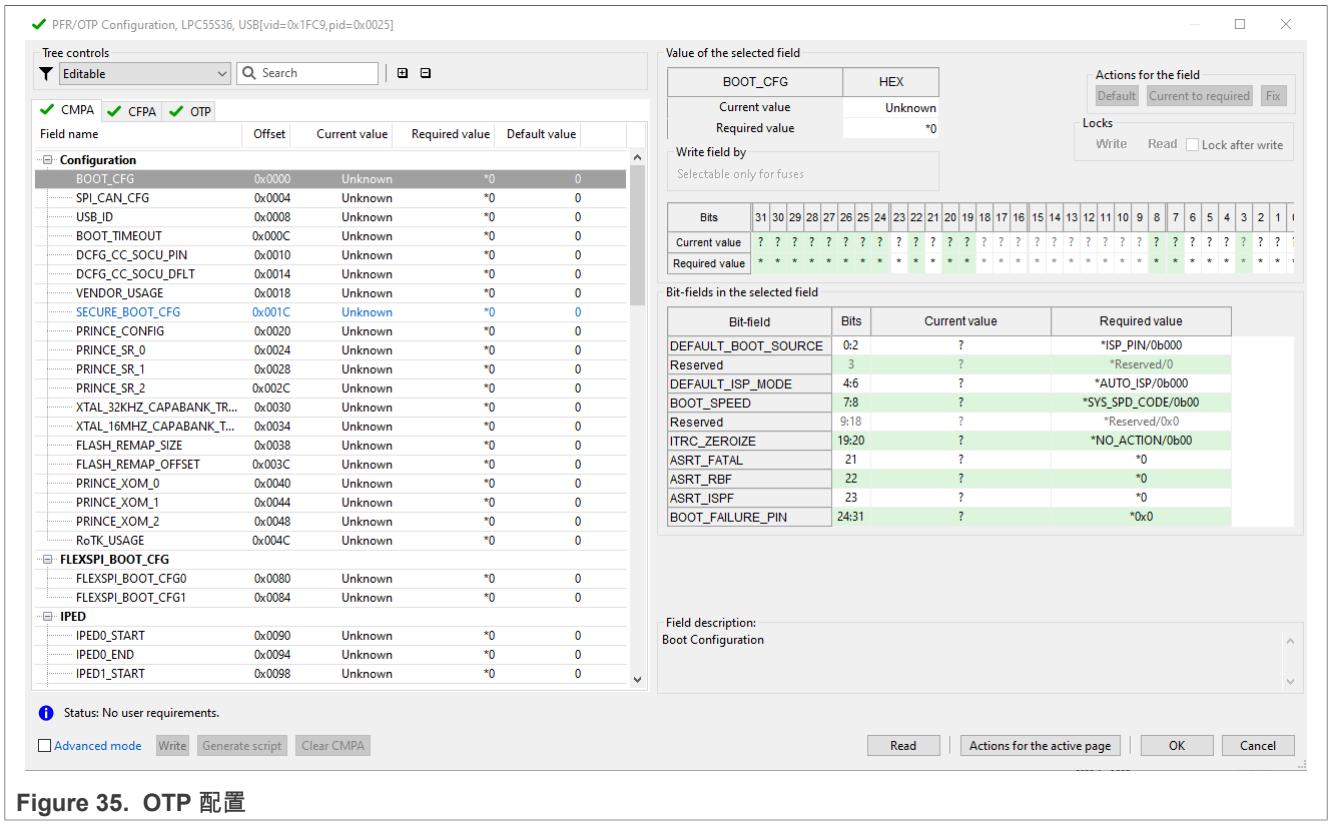

Figure 35. OTP 配置

配置对话框中的所有更改都将作为写入脚本的一部分，Advanced mode 除外。

配置窗口包含三个主要区域：

- 左侧所有 fuses/fields 的表格
- 所选 fuse/field 的详细信息在右侧
- 按钮栏在视图底部

两个主要区域的宽度可以使用分配器进行调整。

5.2.9.1 Table of all items

PFR 配置支持两个页面：CFPA（Customer In-field Programmable Area）和 CMPA（Customer Manufacturing/Factory Programmable Area）。IFR 配置支持称为 ROMCFG （ROM Bootloader configurations） 或 IFR 的单页。每个页面都有属于自己的 一个字段列表，这些字段以树的形式组织。在 OTP 配置中，所有 fuse 都显示在单个树中。树中的项目被组织成逻辑组。所有项目的树以包含以下列的表的形式显示（如果处理器无法使用该功能，则一列可 能无法显示）：

Name： 该项目的人类可读名称。有些名称可能不公开，可以作为受限制的数据包提供。有关受限制数据的部分，请参见 Preferences。

Offset or shadow： 字段地址的偏移量或影子寄存器地址的偏移量

#： 项目的索引（blhost 访问 Fuse 的参数）

R/W/O： 从处理器检索的读/写/操作锁的状态。更多详情，请参阅 Locks。

Current value： 从处理器读取的项的当前值（十六进制）

Required value： SEC 或用户所需的十六进制值。SEC 预设的值以蓝色突出显示，只能在 Advanced mode 下修改。

Default value： 项目的默认值（十六进制）- 重置后的值。

注意： 在某些设备（KW45xx 和 K32W1xx）上，有宽度大于 32 位的 fuse，例如 256 位密钥和 512 位版本计数器。这些长 fuse 显示在表中的几行上，并具有公共#索引（fuse索引）。

5.2.9.2 Tree-filtering toolbar

可以筛选显示在树中的项目。在下拉列表中有预定义的过滤器类型，在工具提示中有描 述。也可以使用文本框搜索元素的名字。

工具栏还包含两个按钮，可以展开或折叠所有组。

5.2.9.3 Item editor

在对话框的右侧，将显示所选项目的以下详细信息：

- 以十六进制数显示当前和所需项目值的表格
- 读写锁的当前状态
- 选择，将写入 fuse 的位置（参见 Burn fuse/write field）。
- 以二进制数显示当前和所需项目值的表格
- 以位域显示当前和所需项目值的表格（仅当项目拆分为位域时）
- 所选对象的描述（项目组，fuse，field，bit-field）

5.2.9.4 Buttons

以下按钮可用于对所选项目进行操作：

Lock after write checkbox： 写入后锁定项目 (请参阅 Buttons)。

Default： 删除该项目的用户设置值并应用默认值。

Current to required： 复制当前项值作为新设置值。

Fix： 自动修复显示的问题。建议使用按钮来解决只能选择一个值的问题。如果有多个选项可用作修复，该功能将设置其中之一。

按钮栏中有以下按钮：

Advanced mode： 参阅 Advanced mode。

Burn/Write： 将所有设置值写入连接的处理器。只在 Advanced mode 下启用。 注意： 请谨慎使用。更改是不可逆的。

Generate script： 生成脚本来烧写设置值。仅供高级用户使用。默认情况下，所有项都由写入脚本写入。只在 Advanced mode 下启用。

Read： 从连接的目标处理器中读取所有项的锁定状态和当前值。可以从源镜像或连接的设备中读取 BCA 和 FCF 页面的值。

Actions for the active page： 显示上下文菜单，其中包含将应用于当前活动页面的操作。

OK： 接受更改并返回到 **Write image**。

Cancel： 关闭对话框而不接受更改。

活动页面的操作中提供以下项目：

All current to required： 复制当前页面的所有当前项值作为新需求。仅当当前值已知时才适用。

Default all： 删除所有用户要求并对所有项目应用默认值。

Import ...： 以 JSON 文件格式导入以前导出的配置。

Export ...： 将当前配置导出为 JSON 文件。

5.2.9.5 Read from connected device

在启动 OTP/PFR/IFR/BCA/FCF 配置工具之前，建议在 **Connection** 对话框中检查板卡是否与主机连接。如果目标设备需要 flashloader（RT10xx、RT116x/7x、和 RT118x），建议在 **Write image** 视图中点击 **Start flashloader**，确保可以与设备建立通信。

配置工具打开后，它会提供从处理器加载 fuse 值。这个功能是可选的，也可以在任何时候使用 **Read** 按钮完成。Preferences 对话框包含一个选项来配置初始读取操作。

读取操作包含以下步骤：

1. 读取锁来找出哪些 fuse 是可读的
2. 读取当前 fuse 值
3. 检测单个的写锁（如果适用于处理器，更多详情，请参阅 Locks）

5.2.9.6 Required value

所选配置必须写入的项目（例如包含预设值）会以蓝色突出显示。其余的项目，默认情况下，要么没有任何设定值 - 显示为 * （星号）；如果默认值适用，则显示为 displayed as *<value> (for example *0)。可以通过如下格式指定这些项目的设定值：

- 按 32 位十六进制数
- 按比特。点击 *Bits table* 第二行的值来切换位，双击可将设定值从该位中移除。
- 按每位字段（仅当寄存器包含位字段时）。
  对于某些位域，值可以从下拉列表中选择，否则就是指定为十进制或十六进制（带 **0x** 前缀）。
- 对于宽度超过 32 位的项，其值只能以十六进制字符串的形式指定。十六进制字符串表示字节序列，其顺序与它们在内存中的存储顺序一致。

5.2.9.7 Burn fuse/write field

在 OTP/IFR 配置中，可以选择烧写 Fuse 或写入字段的源：

- 写入脚本
- 写入应用程序 image 的 SB 文件
- 配置 FW 可以是设备 HSM SB 文件或配置期间使用的 SB/BIN 文件（取决于处理器）。

可用选项取决于配置，例如，如果使用了影子寄存器，则不启用其他选项。

5.2.9.8 Locks

只有在 OTP 配置中才能使用锁来锁住 Fuse。锁指定 Fuse 的访问限制 - 通常是读或写限制。锁必须在开发周期结束时被写入，此时其余的配置已经稳定并经过测试，不会被更改。锁也用于IFR ROMCFG配置。此处，锁定块指定对ROMCFG块的写入访问限制，因为每个16字节块只能写入一次。同样，该锁也用于 IFR 配置。此处，锁指定了对 IFR 字段的写访问限制，因为每个 4 字节字段只能写入一次。

SEC 支持两种类型的锁：

Global： 配置在单独的保险丝中，通常称为 *LOCK*。该配置应用于几个其他 Fuse 或影子寄存器。读、写或操作锁均存在，每种类型限制了对 Fuse 的相应访问。

Individual write locks： 对于某些处理器，可以为单个 Fuse 设置写锁。而且，有些 Fuse 只能写一次且必须设置写锁。这两个功能都显示为 **Lock after write** 复选框，见下面的描述。 此 **Lock after write** 复选框也用于指示对整个ROMCFG页的写入限制。

所有锁的状态在读取操作期间被更新。状态显示在 fuse 表中，具体来说，在这些列中：

R： 显示 Fuse 读锁的状态

W： 显 示 Fuse 写锁的状态（全局写锁和单个写锁的组合状态）

O： 显示 Fuse 操作锁的状态

以下图标表示锁的状态:

锁状态

图标	描述
无图标	Fuse 不支持对应的锁。
	锁状态未知

图标	描述
	Fuse 访问未锁
	Fuse 访问已锁

Lock after write checkbox复选框根据所选 fuse 动态启用或禁用。

- 所选 fuse 不支持单独的写锁 -- 复选框被禁用并且没有选中
- fuse 在第一次写入后必须锁定 --复选框是禁用并且选中
- 单个写锁是可选的 -- 复选框是启用的

对于配置为开启单独写锁的 Fuse，如下图标会被显示在 Fuse 表中的 **Required value** 列

单独写锁

图标	说明
	配置 fuse 以打开单独的写锁

在 RT116x, RT117x, and RT118x 处理器上无法检测到 lock after write 状态，因此即使 fuse 已经锁定，也可能显示为未锁定。

锁验证

在 Fuse 的写锁打开并且 Fuse 值没有完全指定的情况下，OTP 配置会报告一个警告。

5.2.9.9 Calculated fields

有些寄存器或位字段的值通过另一个位字段的值计算得来。例如：

- 一个项的值由另一个项的值按位取反得到
- 一个位域包含该项目的其他位域的 CRC 校验

此功能 在配置中支持验证，在计算值不匹配或未指定计算项目的源值的情况下会显示错误。有关验证和问题解决方案的详细信息，请参见下面的章节。

5.2.9.10 Validation and problem resolution

配置功能可验证所需值，问题会通过窗口标题、树状结构中的图标以及（如果选中该项目）详细信息部分中的所有编辑器中的图标来指示。

BITS 编辑器中，只有受问题影响的位才会显示问题。它允许通过点击影响位值（例 如，反转所需的位值）来轻松修复问题。

对于所有的问题，你都可以使用 **Quick Fix** 按钮。这个按钮修复所选项目内的所有问题。 确保审查 quick fix 所做的更改，以确保新设定的值符合您的期望。

当处理器读取的当前值无效时，系统可能会显示警告信息以通知用户。

注意： 用于设备在其生命周期中的过渡，授予条件或锁定访问设备中的各种调试资源的特定的 OTP/PFR 字段一旦被编程。作为最佳实践，建议一旦安全启动已被验证，就对这些寄存器进行编程。这些字段中的不正确值可能会使平台失去功能或无法进行恢复。在一些平台上，这些寄存器包括特殊的“有效”语义 - 例如，在 LPC55Sxx 处理器上， `DCFG_CC_SOCU_PIN` 和 `DCFG_CC_SOCU_NS` 配置字段包括必须被编程为寄存器中所有其他字段的取反值，以便配置是有效的。只有当寄存器包含非零值时，Fix Problems 工具才会强制执行此约束 - 也就是说，寄存器的显式用户配置已经被检测到。

5.2.9.11 Advanced mode

默认情况下，建议将修改后的配置应用到 workspace 设置文件和脚本中，因此它与可启动 image 一起烧写。然而，有时可能需要烧一个单独的 fuse 值，在这种情况下你可以使用高级模式。高级模式不依赖配置工具单独使用，它允许您：

- 将设定值直接写入连接的处理器（更多详情，请参阅 Write/Burn）
- 生成烧写设定值的脚本。
- 修改所有必需的值，甚至是 SEC-PFR 配置预设的值
- 清除 CMPA 页面；将默认值应用于整个 CMPA 页面

此外，一些保留项的值会在这种模式下从连接的处理器中读取（这些值可能只对 NXP 有用）。

高级配置不期望应用于脚本，因此 **OK** 按钮是禁用的。可以使用 **Export** 按钮将创建的配置存储到外部文件中。

注意： SEC 工具支持的正常工作流不应该需要高级模式，它仅用于该工具不支持的用例。

5.2.9.12 Write/Burn

写入/烧写操作将所有设定值烧写到连接的处理器中，包括所有锁。烧写操作由以下步骤组成：

1. 从处理器读取当前值
2. 更新验证问题
3. 生成脚本
4. 执行脚本

记住，烧录操作是不可逆的。建议：

- 仔细检查所有被烧写的值
- 仔细检查所有被锁定的项目
- 仔细检查配置报告的所有问题
- 生成写入/烧写脚本并检查内容

烧写和生成脚本（Burn and Generate Script）是有区别的：

- 烧写操作是为选定的处理器优化的。如果 Fuse 值匹配或 Fuse 被锁定，则 Fuse 不会被烧写。对于 CFPA 和 CMPA，整个页总是能被写入的。
  警告： 生成脚本默认是在新出厂的处理器上使用。
- 它包含了所有 Fuse 的配置。如果任何 Fuse 已经被烧写，它可能会失败。

5.2.9.13 PFR/IFR and OTP differences

- OTP 配置中的项目被称为 Fuse，而其它配置中的项目被称为 Field。
- OTP 配置中的 Fuse 是逐项烧写的，所以可以指定单个设定值。IFR ROMCFG 配置中的字段由 16 字节块写入，因此必须指定已完成的 16 字节要求。PFR 总是更新整个页 面，所以如果没有设定值，将使用默认值。
- 仅对 Fuse 支持锁和 burn fuse by 选项。
- Fuses、IFR 字段和 IFR ROMCFG 模块支持锁选择。
- CFPA，CMPA，BCA，和 FCF 页面可以多次写入，而 ROMCFG 块或 IFR 字段只能写入一次。

5.3 Write image

使用 **Write image** 视图将 image 写入启动设备并烧写 Fuse 以实现安全启动。**write image** 页面的图标上可能有一个脏标志，这意味着有一些选项改动没有在上一次构建操作中生效。

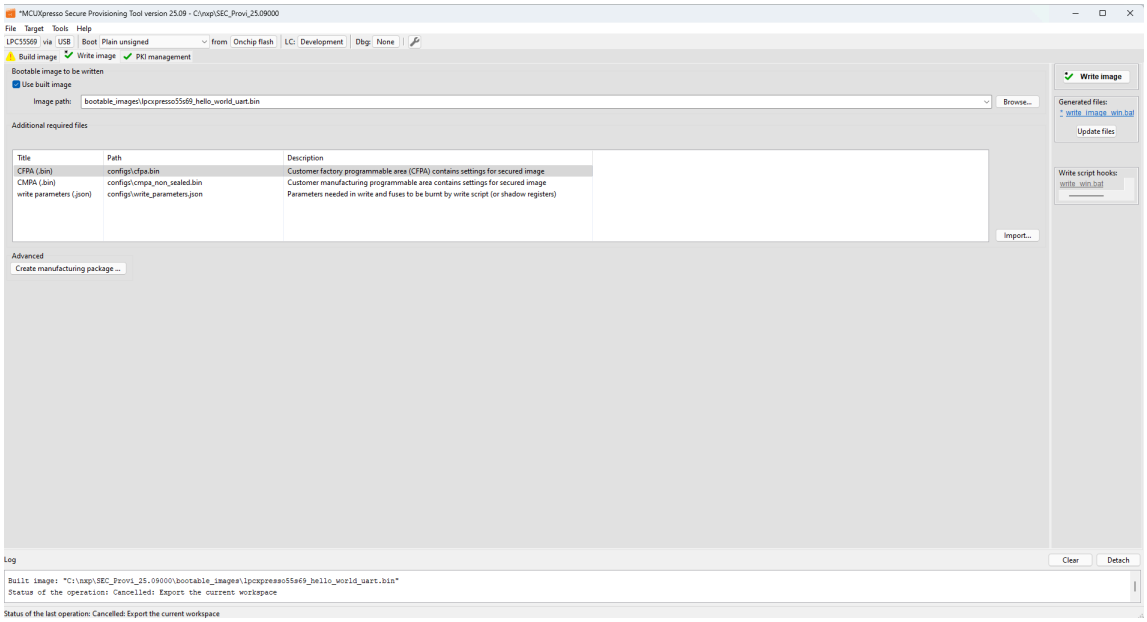

Figure 36. 写入 **image (LPC55Sxx)**

5.3.1 Controls on the write image view

Use built image： 如果选中，将烧写 Build image 操作的输出文件。

Bootable image： 要写入目标设备的 image 的路径。文件扩展名特定于处理器和启动类型，它是用于可启动 image 的 BIN 或用于安全二进制胶囊的 SB。二进制 image 必须是不带 FCB 块的“nopadding”形式，因为 FCB 块是在单独的步骤中写入的。

Additional input files： 显示 **Write image** 操作的输入文件。内容取决于处理器、 启动类型和其他构建选项。默认情况下，内容是 **Build image** 操作的输出文件。您可以使用 **Import** 按钮手动将每个文件替换为自定义文件。

Start flashloader： 允许您在连接的处理器上初始化和启动 flashloader。如果开启了芯片安全特性，则会自动创建签名 flashloader。如果您想从命令行中使用 blhost，这将非常有用。

Test life cycle： 打开可设置临时高级生命周期状态的对话框。 这允许它在选定的生命周期状态下测试处理器行为，而无需烧毁 fuses。必须通过调试器探测器连接开发板。

Disable flash security： 打开一个对话框，用户可以在其中指定后门密钥来禁用闪存安全。后门密钥可以从 FCF 配置、可启动 image 中获取，也可以手动输入。作为闪存禁用过程的最后一步，可以执行批量擦除来清除整个闪存。

Write image panel： 在窗口的右侧，有 Write image 面板，其中包含 **Write image** 按钮和写入脚本的文件名。

Generated files： **Generated files** 面板具有与 Generated files.相同的功能。

Write script hooks： 包含写入脚本执行期间调用的 hook 脚本的面板。灰色标签表示该文件在 workspace 中尚不存在。单击后，将使用示例中的内容创建一个新文件并打开进行修改。如果文件存在，标签为蓝色，单击后将打开文件。更多详情，请参阅 Script hooks workflow。

点击 **Write image** 按钮之前，确保主板已连接并配置为ISP模式。如果写入脚本执行了任何不可逆操作，则会出现一个包含详细信息的确认对话框。

5.3.2 Manufacturing package

量产包是一个ZIP文件，包含写入脚本和写入操作所需的所有其他文件，它被设计为将文件发送到工厂进行生产。点击 **Write image** 页面上的按钮 **Create manufacturing package …** 可以创建量产包。该对话框在制造工作流程中进行了描述。

在生产中，可以使用 **main menu > File > Import Manufacturing Package….** 导入 Manufacturing package。更多详情，请参阅 Manufacturing operations。

5.4 PKI management

PKI management 视图显示用于验证 image 文件真实性的证书列表。生成的密钥可以被导出以供以后使用。PKI 管理允许生成：

- 用于 image 文件认证的密钥
- 用于信任配置的密钥
- 用于调试身份验证（Debug Authentication）的密钥对

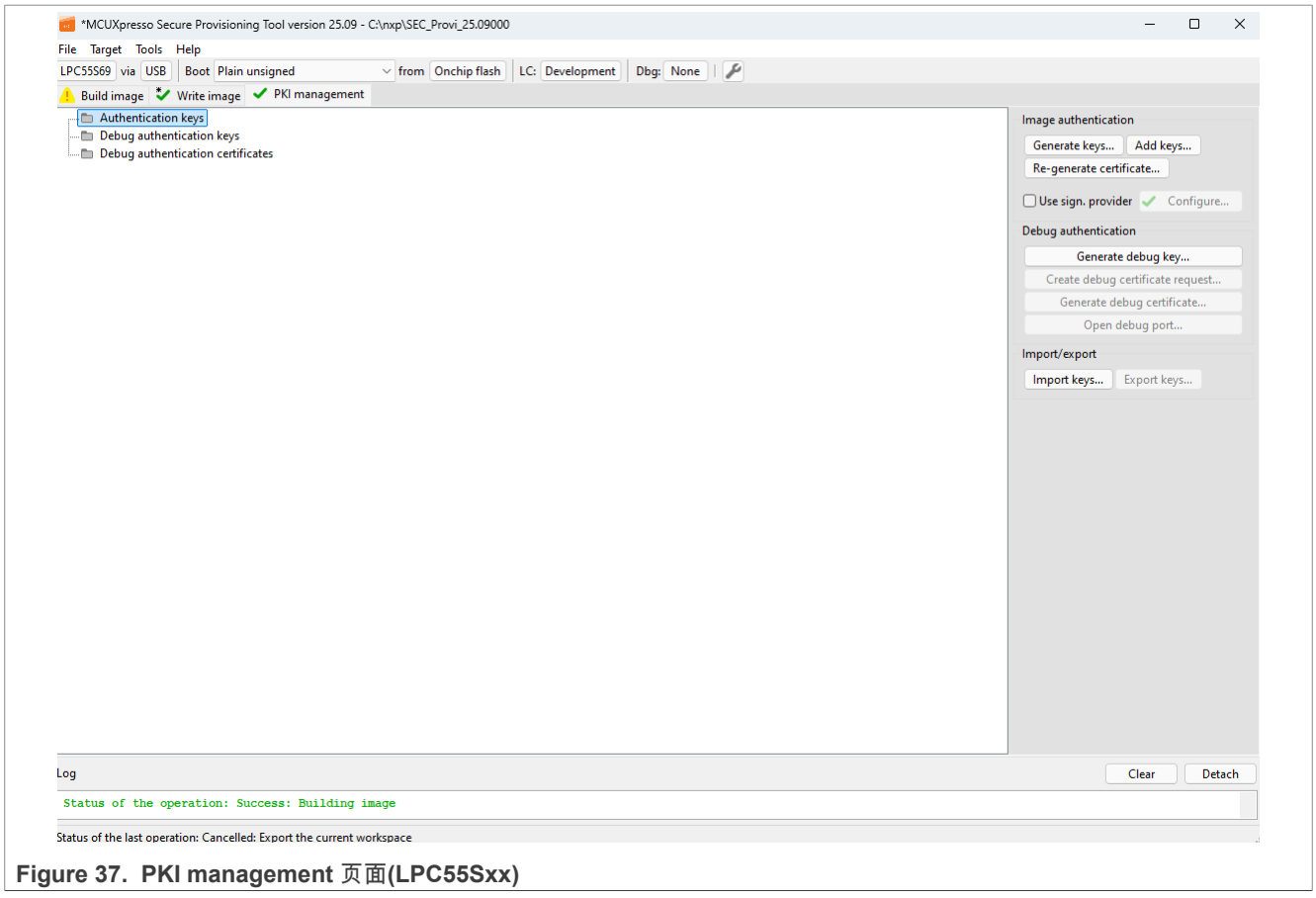

Figure 37. PKI management 页面(LPC55Sxx)

5.4.1 Generate keys

要验签的 image 文件依赖于一组 Public Key Infrastructure（PKI）证书。SEC 工具通过图形界面简化了 PKI 兼容证书的生成过程。

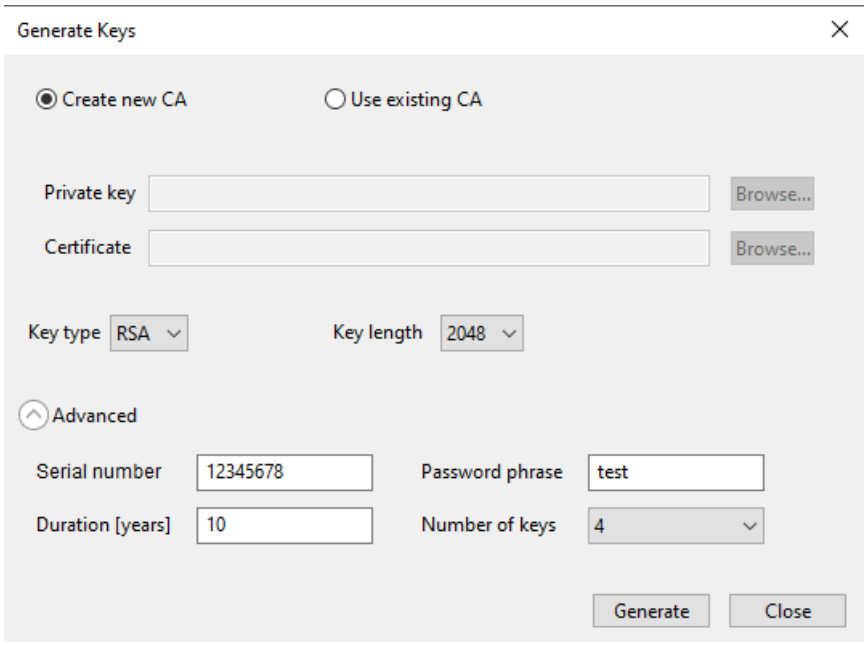

Figure 38. 生成密钥 **(RT1xxx)**

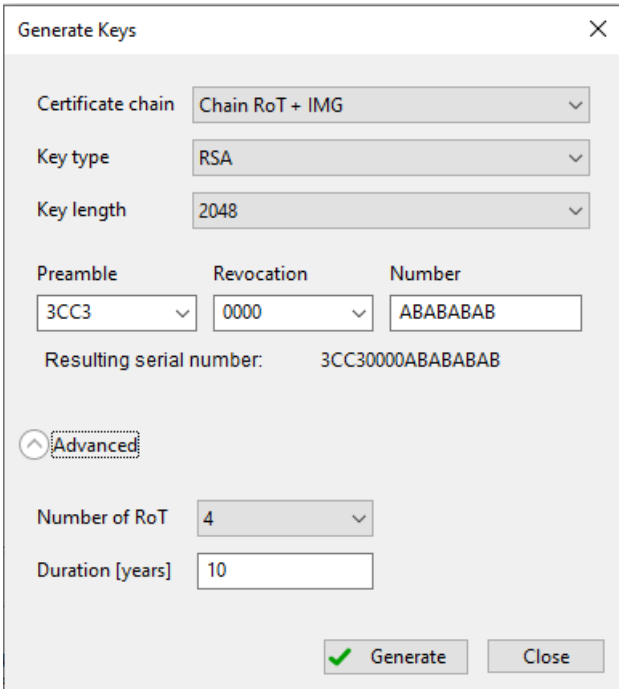

Figure 39. 生成 **RSA** 密钥

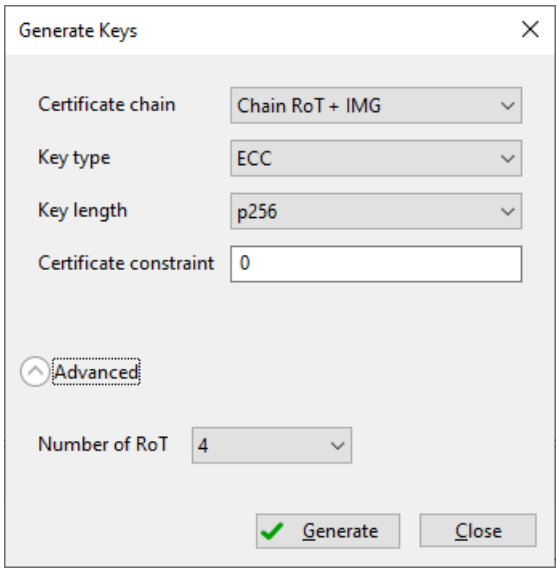

Figure 40. 生成 ECC 密钥

Create new CA： 使用此选项生成具有证书颁发机构（Certificate Authority）功能的密钥。

Use existing CA： 该选项卡可启用使用用户指定的CA。**Certificate** 和 **Private Key** 必须为 PEM 格式。

Private Key： 私钥文件路径

Certificate： 证书文件路径

Key type： 选择生成的密钥类型。

Key length： 选择生成的密钥长度（以 bit 为单位）。

Serial number： 用于密钥撤销的值。LPC55Sxx 和 RT5xx/6xx 上的 RSA 密钥序列号由三部分组成：前导码、撤销和编号。撤销部分根据 OTP/CFPA 中的撤销字段进行验证。

Password phrase： 使用指定参数保护生成的 CA。

Duration [years]： 设置证书的有效期（年）。对于支持的处理器，它仅用于签名目的，因为持续时间在硬件中不能直接验证。

Number of keys： 设置需要生成的 RoT（信任根）密钥个数。支持的最大密钥数以及推荐的默认值是4。

Certificate chain： 密钥链的深度。

Preamble： 序列号的前缀，强制固定值

Revocation： 序列号的中间部分，16位撤销ID — 此值应与设备上 OTP/PFR (CFPA) 中的 IMG_REVOKE 字段匹配。

Number： 用于唯一标识证书/密钥的序列号字节的后缀。

Certificate constraint： 用于撤销 image 签名密钥，根据 OTP/PFR 中的撤销字段进行验证。

有关 Password，Serial 和 Duration 选项的更多详细信息，请参阅 OpenSSL 文档。

指定所有参数后，单击 **Generate** 按钮。密钥生成脚本的输出将显示在进度窗口中。

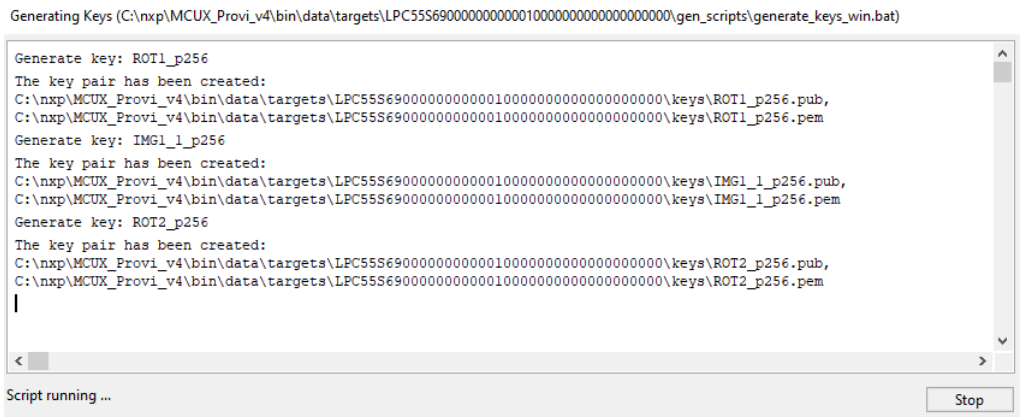

Figure 41. 生成密钥 - 进度

5.4.2 Add keys

一旦在 **Generate keys** 对话框中生成密钥，可以用 **Add keys** 对话框添加它们。

IMG key path： 要生成和添加的 IMG (ISK) 密钥的路径。

CSF key path： 要生成和添加的 CSF 密钥的路径。

Key： 要生成的密钥类型：用于 image 签名的 **Image**，或用于创建密钥链的 **Intermediate**

有关其他项目的说明，请参阅 Generate keys。

指定首选项后，单击 **OK** 以添加密钥。输出将显示在进度窗口中。

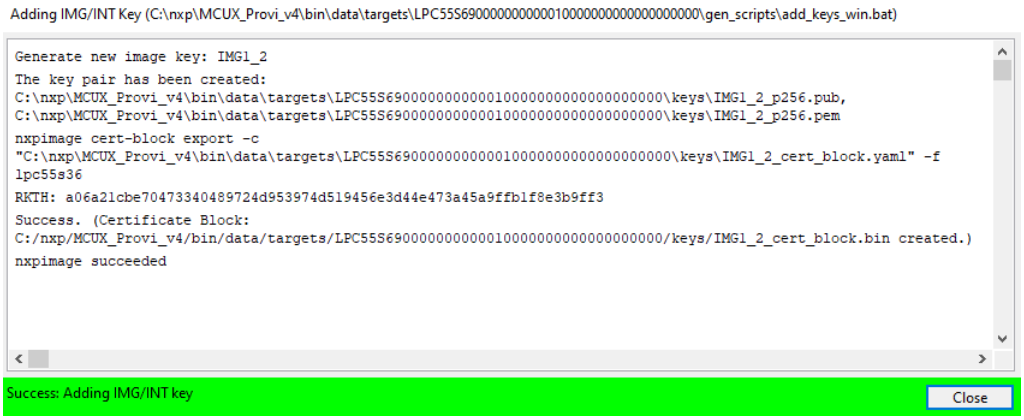

Figure 42. 添加密钥进度

5.4.3 Re-generate certificate

SEC 工具允许重新生成 ROT 证书，用于证书吊销（仅适用于 RSA ROT 证书）。选择证书并单击 **Re-generate certificate...** 按钮。在对话框中，查看参数，更新序列号并确认重新生成。当前证书将备份到备份文件夹中，然后重新写入。

再生参数与生成参数相同，详细说明，请参见前几章。

5.4.4 Import/Export keys

您可以使用 Export 功能导出生成的密钥以供以后使用。导出密钥步骤：

1. 在 **PKI management** 中，选择 **Export keys**。
2. 在对话框中，导航到你想要导出密钥的位置，并选择 **Select folder**。

注意： Export keys 也导出工作空间配置，包括对称密钥，例如，用于 LPC 的 SBKEK 或 RT10xx 处理器的 BEE 用户密钥。

您可以稍后使用 Import 功能将密钥导入到新的工作空间中。该操作在当前 Workspace 中 备份当前密钥和设置。从导入的文件夹中恢复对称密钥，例如 LPC55Sxx/RT5xx/RT6xx 的 SBKEK、RT10xx 的 BEE 用户密钥、RT116x/RT117x/RT118x/RT5xx/RT6xx 设备的 OTFAD 密钥数据以及 RT116x/RT117x/RT118x 的 IEE 密钥数据。

导入密钥步骤：

1. 在 **PKI management** 视图中，选择 **Import keys**。
2. 在对话框中，导航到包含 **keys** 和包含要导入的 key 信息的 **crts** 子目录的文件夹，然后选择 **Select folder**。

要导入未从 SEC 工具导出的外部密钥：

1. 生成新的虚拟密钥并将其导出到新文件夹中；确保密钥的数量与要导入的密钥相匹配。
2. 用导入的文件覆盖导出文件夹中的密钥（重命名导入的密钥，使其与 SEC 工具中使用的命名约定相匹配）。
3. 从文件夹中导入密钥。

5.4.5 Debug authentication

只有支持 DA (Debug Authentication）功能的处理器，在 PKI management 中才会显示 DA 按钮。**Generate debug key** 按钮总是启用，**Create debug certificate request** 和 **Open debug port** 只有在 DA 密钥存在时才启用。**Generate debug certificate**在具有身份验证密钥的 workspace 中启用。**Generate debug key** 按钮打开一个对话框，用于创建调试身份验证密钥对。**Generate debug certificate request** 帮助您创建一个请求，该请求将被发送到 OEM，从中可以生成调试证书。**Generate debug certificate** 按钮显示一个证书，OEM 在其中选择密钥并设置将授予证书持有者的权限。**Open Debug Port** 对话框通过使用 DAC 准备连接的设备进行调试。

注意： 更多详情，请参阅 Debug authentication workflow。

5.4.6 Signature provider

签名提供程序允许使用自定义提供程序进行身份验证，而不是存储在本地计算机上的密钥。签名提供程序需要自定义实现 HTTP 服务器，并使用提供身份验证的简单 API。

一旦选中 **Use sign. provider** 复选框，就可以打开配置对话框。启用后，workspace 中的当前密钥将被移动到备份文件夹。并且签名提供程序服务器将用作公钥的来源并提供签名。**Signature Provider Configuration** 对话框中设置的配置将在用于构建的配置文件中使用 (`cert_block`, `mbi_config`, `sb_config`, `debug_credentials_config`)。

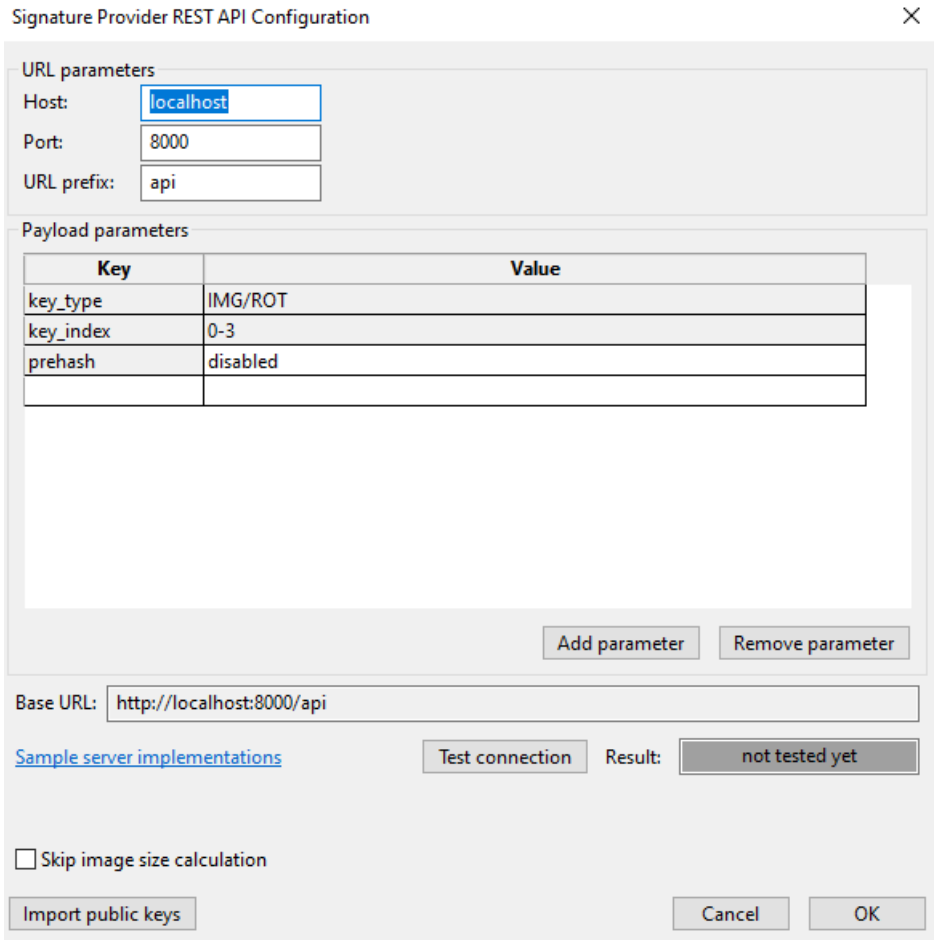

Figure 43. 签名提供程序配置

5.4.6.1 URL parameters

服务器 URL 中所需的参数：

- **Host** 服务器的主机名或 IP 地址，接受 RFC 3986 中指定的符号。建议使用 localhost，因为 HTTP 通信不安全。对于与另一台计算机的通信，建议实现代理服务器，通过安全通道 (HTTPS) 转发通信。
- **Port** 服务的端口号，为整数。
- **URL prefix** REST API 前缀，如果没有则使用空字符串。

5.4.6.2 Payload parameters

负载参数作为 JSON 负载随每个发送到服务器的请求一起传递。每个参数都由唯一的键标识并包含一个文本值。以下按键是为该工具保留的：

- **type** - SPSDK 用于识别签名提供者类型。
- **host, port, url_prefix**- 用于指定服务器连接。
- **data** - 用作请求数据负载的密钥。
- **key_type** - 由 SEC 工具用来指定是否应使用 image 或根密钥；支持的值为 **IMG** 或 **ROT**。
- **key_index** - 由 SEC 工具用来识别应使用哪个私钥，值是 0-3 范围内的十进制数，通过从构建选项卡 ROT1 -> 0、ROT2 -> 1、IMG1_1 -> 0、IMG2_1 -> 1 中选择密钥来设置等等。

- **prehash** - 允许选择是发送完整数据进行签名还是仅发送 hash。可以从下拉菜单中选择 hash 算法；但是，建议使用默认值。

5.4.6.3 Buttons and Base URL

- **Remove parameter** 用以从表中删除一个参数，删除选定的参数。如果没有选择参数，则删除最后一个参数。
- **Add parameter** 用以在所选参数下方或表格末尾添加一个空行。
- **Base URL** 用以显示根据所需参数创建的 URL。
- **Test connection** 用以发送签名长度请求以测试服务器是否已启动并运行。
- **Import public keys** 用以从服务器导入公钥，请求发送到端点 'public_keys_certs'，服务器响应格式在 `signature\_provider\_key\_tree\_schema_##\_##.json`. 描述。
- **Sample server implementations** 用以打开包含签名提供程序服务器实现示例的文件夹。

5.4.6.4 Signature provider server API

签名提供程序 API 必须实现以下三个端点，以及一个可选的附加端点：

`sign`

sign 处理给定数据块的签名。该请求发送要使用可选参数指定的私钥进行签名的数据。选择 ROT1 或 ROT1-IMG1_1 作为签名密钥的 MCX947 签名请求示例：“http://localhost:8000/server/sign”json 有效负载：

使用 ROT1 密钥签署证书：

```
{
  "data": "hex string of certificate block to be signed",
  "key_type": "ROT"
  "key_index": 0
  "prehashed": "none if not prehashed, hash algorithm name if data is prehashed"
}
```

使用 IMG1_1 密钥进行 image 签名：

```
{
  "data": " hex string of image block to be signed",
  "key_type": "IMG"
  "key_index": 0
  "prehashed": "none if not prehashed, hash algorithm name if data is prehashed"
}
```

`signature_length`

signature_length 返回签名长度（以字节为单位）。

`verify_public_key`

verify_public_key 验证所使用的公钥与服务器上的私钥是否形成有效的密钥对。公钥作为数据有效负载发送，RSA 密钥以 NXP 专有格式发送，ECC 密钥以 DER 格式发送。要标识应与此密钥匹配的私钥，请使用可选参数。如果键匹配，则返回“true”。

该 API 旨在检测服务器使用的密钥与客户端不同的密钥是否存在问题。对于不需要此类验证的环境，实现可以简单地返回“true”。

`public_keys_certs`

public_keys_certs 是一个可选的端点。如果实施，它允许将公钥从签名提供程序对话框导入 SEC 工具。SEC 工具期望来自该端点的响应是身份验证密钥的树，ROT 密钥位于顶层，IMG 密钥作为叶子。响应的结构如

<install_folder>/bin/_internal/schema/signature_provider_key_tree_schema_##_##.json 所述。通常，可以导入可从“生成密钥”对话框生成的任何树，但根密钥仅限一个子密钥。RSA 树不需要每个子密钥具有相同的密钥长度。

5.4.6.5 Server examples

服务器示例是 SEC 工具分发的一部分。有服务器实现的 Python 脚本示例。这些事例在 Signature provider workflow 中进行了描述。

5.5 Manufacturing Tool

使用 **Manufacturing Tool** 可以在同时连接到主机的多个设备上配置资源。**Manufacturing Tool** 可以从 菜单栏 中的 **Tools** 中访问。处理器可以通过 USB 或 UART 或 SPI 或 I2C 连接到主机。用户界面由这些区域组成：**Operation**，**Command**，**Connected Devices**，**Communication parameters** 和按钮栏。如果处理器支持信任配置，界面也有 **Trust provisioning** 区域。

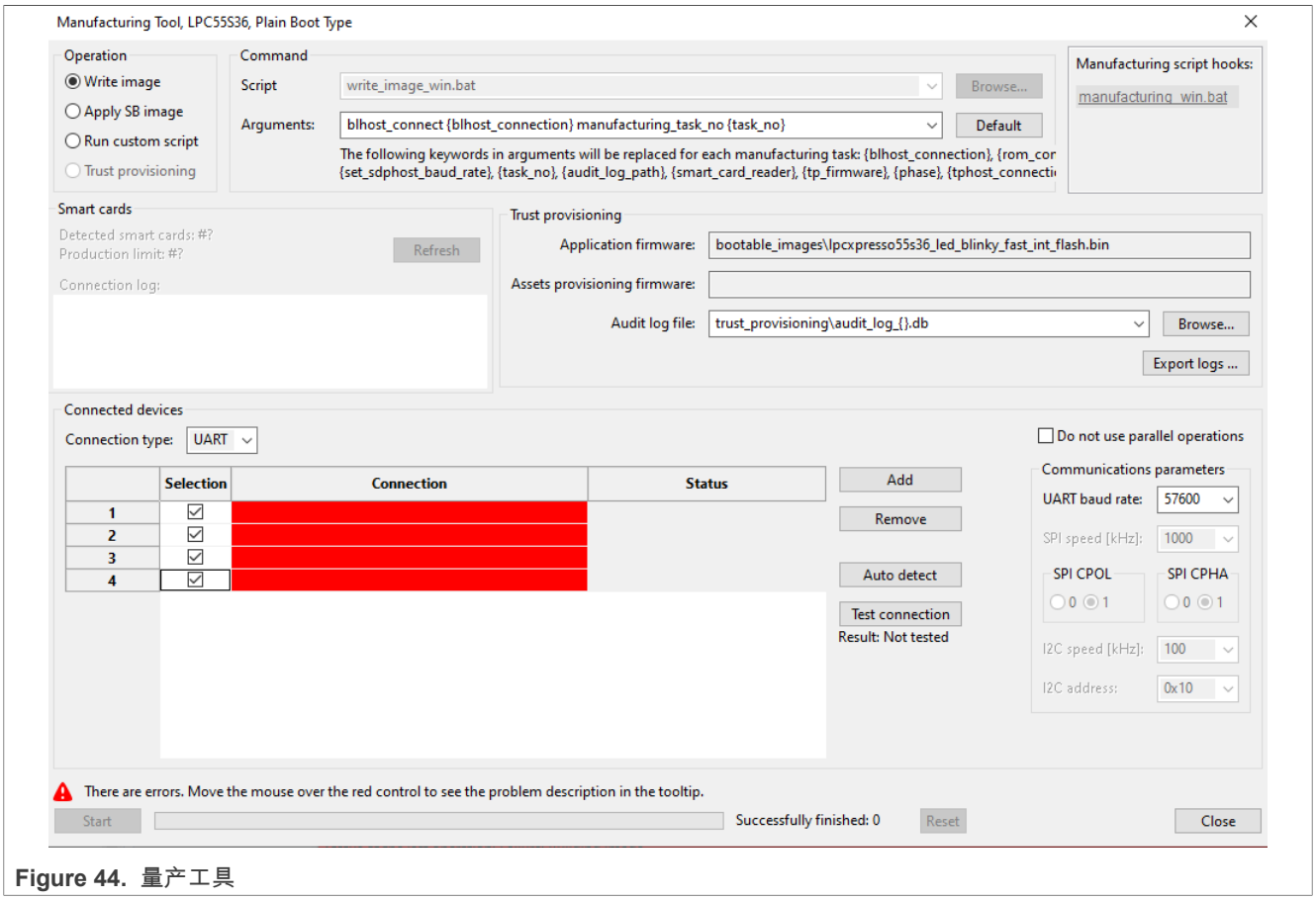

Figure 44. 量产工具

Operation： Operation 区域包含代表不同操作的单选按钮。

- **Write image** : 执行与 **Write image** 相同的操作。在使用之前，必须在 **Build image** 视图中构建 image，并且 **Write image** 视图不能显示任何错误。
- **Apply SB image** : 使用 SB（Secure Binary）格式更新处理器的现有 image。对于大多数处理器，SB 文件是在构建安全 image 操作期间创建的， 默认情况下位于 Workspace 的 *bootable_images* 子文件夹中。对 于 RT10xx/RT116x/RT117x/RT118x 处理器，必须手动创建 SB 文件，因为 SEC 目前不支持它。

- **Run custom script** : 使用自定义脚本配置。假定该脚本是一个经过修改的 SEC 脚本，接受 SEC 参数（特别是连接参数）。退出状态 0 被视为成功。

Command： Operation 区域框中选中的操作其参数在 Command 区域框里。根据所选操作的不同，参数会有所不同。

- **Script (Write image, Run custom script, Trust provisioning)** : 写入或自定义脚本的路径。请使用 **Browse** 按钮指向自定义脚本。
- **SB file (Apply SB image)：** SB 文件的路径。请使用 **Browse** 按钮指定文件。
- **Arguments：** 操作期间使用的参数的交互式列表。使用 **Default** 按钮可以恢复任何 更改。注意，参数中花括号括起来的关键字将为每个量产任务替换。 您可以在工具提示中找到带有描述的完整关键字列表。

Manufacturing script hooks： 带有在制造脚本执行开始和结束时调用的挂钩脚本的面板。灰色标签表示该文件在 workspace 中尚不存在。单击后，将使用示例中的内容创建一个新文件并打开进行修改。如果文件存在，标签为蓝色，单击后将打开文件。更多详情，请参阅 Script hooks workflow。

Trust provisioning： 信任设置操作的配置。

- **Application and Assets provisioning firmwares：** 列出的这些文件路径仅供参考。
- **EdgeLock 2GO API key：** 用于访问 EdgeLock 2GO 服务器的密钥在这里指定，也可以在环境变量 `SEC_EL2GO_API_KEY`。
- **EdgeLock 2GO server: Test connection：** 允许测试与 EdgeLock 2GO 服务器的连接。
- **EdgeLock 2GO product batch DB：** 用于每个产品（离线）流程的 EdgeLock 2GO 安全对象数据库路径

Connection type： 该按钮允许选择与处理器的通信接口。

#phases： 针对同一设备执行多个阶段。传递给写入脚本的 USB 设备标识可能会在重置后发生变化（请参阅 USB path）。如果写入镜像脚本支持 USB 连接，则脚本将分 **phases** 执行，每个阶段都通过重置完成，并在执行下一个阶段之前更新 USB 标识。此参数允许覆盖正在执行的阶段数和强制自定义数量（以防写入脚本被手动修改）。取值`disabled`表示未使用相位。建议指定 `default` 值。如果选择 `default` 值，则会在工具提示中显示当前阶段数。

Do not use parallel operations： 选中此复选框可禁用制造过程中的并行操作。如果检测到并行操作出现任何问题，可以使用此功能。

Connected devices： Connected Devices 区域包含一个显示所有已连接处理器的交互式表。该工具可以检测和编程使用 USB 连接的处理器（只有当处理器是 ISP 模式）。

列：

- **Selection：** 允许用户启用/设备；每个设备都在表格中配置。如果该处理器被取消选中，下次自动检测将保持该处理器被取消选中。
- **Connection：** 允许用户手动输入设备名称或路径。
- **Status：** 显示已连接设备的状态。操作执行完成后，可以通过单击 **Status** 列中的条目来打开日志文件。

用于修改已连接设备的按钮：

- **Select all：** 该按钮用于选择下表中所有设备。
- **Deselect all：** 该按钮用于取消选择下表中所有设备。
- **Autodetect：** 自动检测连接到所选连接类型的主机的所有设备。这是推荐用法。有关 USB 路径的详细信息，请参阅 USB path。
- **Add：** 手动添加设备的按钮（在表格中添加一个空行）。
- **Remove：** 此按钮用于从表格中移除选定的设备（移除选定的行）。

- **Test connection**： 用于测试（ping）所选设备的按钮。该特性可能对 UART、SPI 和 I2C 设备有用，因为在这些设备中，检测到通信设备（COM 端口或 USB 路径）并不意味着与处理器建立了连接。因此，建议在开始生产操作之前进行测试连接。
- **Export logs**：允许将制造日志导出为zip文件。除了导出所有日志外，还可以仅导出特定日期的日志。

Load KeyStore (Apply SB image for RT5xx/6xx)： 在上传 SB 文件之前从外部 Flash 加载 KeyStore。

Communication parameters： 与设备的连接配置。*Baud rate* 选项允许从下拉列表中选择一个值，或者手动指定一个值。更多详情，请参阅 Connection。

Crystal freq. [kHz]： 用作处理器时钟源的外部晶体的频率。这仅用于 `lpcprog` 工具。

Button bar： 按钮栏包含操作按钮，并显示任何警告和警报。

- **Start**： 启动所选并配置的操作。您可以在相邻的进度条中观察操作的进度。
- **Successfully finished**： 在标签上注明成功完成的操作数量。这个数字会自动递增，并存储在 Workspace 中的设置文件中。
- **Reset**： 该按钮允许重置 Successfully finished 计数器。
- **Close**： 该按钮不运行操作直接关闭量产工具。

5.5.1 USB path

量产工具支持使用串口、USB、SPI 或 I2C 连接。自动检测连接功能足以应付大多数情况。USB 路径用于识别 USB、SPI 或 I2C 通信的 USB 设备，因此可以唯一识别具有相同 USB VID+PID 的设备。它根据操作系统的不同而不同。USB 路径格式的描述可以在 SPSDK 文档中找到。

量产工具支持自动USB路径更新，在量产操作期间可能会发生这种更新，原因如下：

1. 在某些操作系统上，每个设备重置后，USB路径都会改变。
2. ISP 模式下的 RT10xx 处理器 VID/PID 相较于 flash loader 应用程序的 VID/PID 是不同的。

更新后，量产工具将自动执行完成所需操作的脚本的第二部分。

注意： ISP模式下的RT10xx设备与flash loader应用程序的VID和PID相比，具有不同的VID和PID，上传到目标端，对flash进行编程。在 ISP 模式中重置处理器时，自动检测会搜索带有 PID&VID 的设备。如果 flashloader 在处理器上是激活的，则将其重置为 ROM bootloader 模式。量产过程略有不同：作为第一步， flashloader 被上传到所有板卡上，并检索其 USB 路径。在这一步之后，剩下的过程是并行执行的。量产完成后，设备 flashloader 仍然处于激活状态，所以在复位处理器之前，设备不会被检测到。

5.6 Flash Programmer

Flash 编程器旨在从当前选择的 flash 读取/写入，并支持所有 flash 类型，包括内部 flash， 外部 NOR 和 NAND flash，SDcard 和 eMMC。Flash 编程器可以从菜单栏中的 **Tools** 中访问。处理器可以通过 USB，UART，SPI 或 I2C 连接到主机，并且必须处于 ISP 模式（因为工具内部使用 blhost 或 lpcprog 协议）。Flash 编程器可以用来准备写入数据，在没有设备连接时只是显示或修改保存的数据块。左边包含操作工具栏，右边包含十六进制和 ASCII 格式的内存内容的显示区。工具有额外的功能：“自动擦除”和“自动验证”。要读取一段地址范围内的数 据，请在 Address 框中填写起始地址，并在 Size 框中填写所需大小（以字节为单位）。点击 Read 按钮将从内存中读取值并显示出来。操作结果显示在左下角。如果操作以失败结束，工具将显示遇到的错误的详细信息。Flash 编程器窗口的设置不会保存到 workspace 中。如果工具关闭，它们将被丢弃。

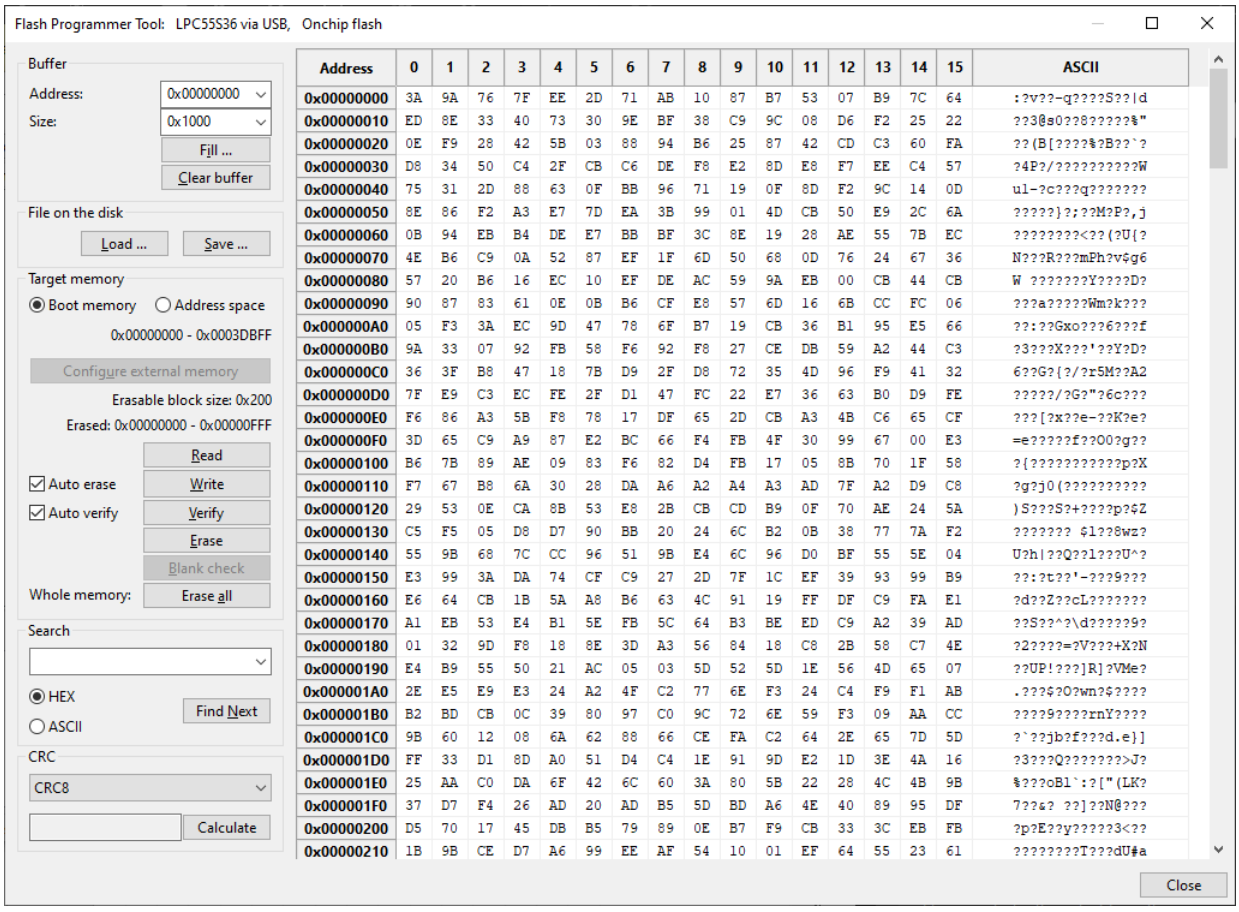

Figure 45. Flash 编程器工具

Buffer： 缓冲区代表闪存的一部分（副本），该部分将由处理器写入或读取。缓冲区由地址、大小和内容定义。

- **Address**： 指定缓冲空间的起始地址。该地址不能低于或高目标内存的开始或 结束地址。
- **Size**： 缓冲区的大小指定应读取/写入多少字节。文件大小限制为 16 MB。
- **Fill ...**： 打开可以用如下值（字节、单字、双字或随机）填充显示区的对话框。 单字和双字可以是大端或小端。
- **Clear buffer**： 清除显示的值并将 size 设置为 0。

File on the disk：对磁盘上的文件进行加载/保存操作。

- **Load ...**： 将文件加载到缓冲区； `.srec` 和 `.hex` 文件被格式化（起始地址）加载，其他没有起始地址信息的文件类型被作为二进制文件处理。address 是应该加载进显示区中的开始地址，size 指示应该从数据的哪一部分开始加载。加载的数据与现有的显示区上下文合并 （例如，缓冲区可以扩展），重叠的区域将被覆盖。加载操作也支持拖放功能，允许直接将文件添加到网格中。
- **Save ...**： 将显示区或显示区的一部分保存到文件中。支持的文件格式是 `.srec`, `.hex`和 `.bin`

Target memory： 该块提供对目标内存的基本操作。顶部的十六进制范围是所选内存的有效存储器空间范围。

- **Selection of the memory type**： 选择要定位的内存：`Boot memory` 指与工具面板中选择的内存相同的内存。`Address space`指处理器的整个地址空间。可以通过下面的按钮直接从工具重新配置 `Boot memory`，而对于`Address space`，该工具则期望内存已经可用。
- **Configure**： 尝试配置已连接的存储器。它执行与 启动内存配置 中相同的存储器配置。如果是带有 FCB 的存储器，那么配置的最后一步将尝试读取最小可擦块的大小。该命令只支持外部 Flash，不支持内部 Flash。

- **Erasable block size**： 定义最小可擦的块大小。此值可从FCB读取（如果可用），或者从不使用FCB的内存的启动设备配置中读取。如果找不到可擦除块大小，则无法计算已擦除的大小。
- **Erased**： 该地址范围将通过擦除操作被擦除。地址范围由缓冲器地址和大小确定，并且与可擦除块边界对齐。如果缓冲区未与可擦除块大小对齐，则会显示警告。
- **Read**： 从存储器中读取给定范围的数据以填充缓冲区。对于带有 ECC 的存储器类型，如果存储器被擦除，则不能读取。
- **Write**： 将显示区中的值写入存储器，参见选项 Auto erase 和 Auto verify。
- **Verify**： 检查显示区中的值是否与存储器中的值匹配，不匹配的值将用红色高亮显示，工具提示将显示存储器中的值。
- **Erase**： 擦除存储器，总是擦除向上对齐的最小可擦块大小。
- **Blank check**： 检查存储器是否为空，不支持内部 Flash。
- **Erase all**： 擦除全部存储器。
- **Auto erase**： 写操作时，在写操作前将存储器擦除。
- **Auto verify**： 写操作完成后，验证缓存区是否正确写入。

Search： 在缓存区中搜索一个值；该值可以以十六进制数（”FF AA”, “12 A3 00”）或 ASCII（”abc”, “hello world”）的形式提供。找到的值以蓝色突出显示，开始地址显示在左上角。 **CRC**： 计算缓冲数据的所选类型的 CRC。

5.7 SB editor

SB 编辑器工具旨在创建用于更新软件、数据和/或处理器配置的自定义安全二进制文件。SB 编辑器支持 SB 格式：`SB2.x`、`SB3.x`、和 `SBx`，并且还支持常规 `SB` 文件和 `device HSM SB` 文件。可以使用命令 **main menu > Tools > SB Editor** 启动 SB 编辑器。当第一次打开 SB 编辑器时，并且工作空间已经包含带有 SB 配置的 YAML 文件（推荐使用），SB编辑器工具提供导入文件的功能。

SB 文件是使用 nxpimage 或 nxpdevhsm 命令行实用程序生成的，因此 SB 编辑器生成配置 YAML 文件，该文件用作这些工具的输入。SB 编辑器是 YAML 配置文件的智能 GUI 编辑器。在图形用户界面 (GUI) 中，还支持“$”扩展名（参见高级命令和 $ 变量），这些扩展名会被转换为相应的值。

5.7.1 SB editor contains the following main UI parts:

Top buttons bar： 该栏允许选择 SB 文件类型、导入、导出或清除当前配置。

Properties： 该页面允许指定 SB 文件生成属性。

Commands： 该页面允许指定 SB 文件中应执行的命令序列。

Output： 该面板可以指定输出文件（配置 YAML 文件和生成的 SB 文件）。**View** 按钮允许更新 YAML 配置文件并在外部编辑器中打开。**Generate** 按钮允许（重新）生成结果 SB 文件。

Status and bottom buttons： 允许创建制造包和启动制造工具的按钮。**OK/Cancel** 按钮允许无论是否将配置保存到 workspace 都可以关闭 SB 编辑器。

5.7.2 File type

SB 编辑器支持创建：

- **regular SB file** 由构建视图上指定的 OEM 密钥保护；
- **device-specific SB file** 用于由设备特定密钥保护的设备 HSM。创建此 SB 文件需要将处理器连接到计算机。更多详情，请参阅 Connection。

每种文件类型的属性和命令可能不同，因此切换文件类型可能会影响当前配置。

5.7.3 Properties view

Properties 视图允许编辑常规属性和属性列表。如果处理器包含任何列表类型属性，则可以在左侧选择它，右侧显示属性编辑器表。如果处理器不支持列表属性，则 GUI 仅显示属性编辑器表。

属性编辑器表包含以下列：

Group： 特性按逻辑分组（仅供参考）。

Property： 属性名称（仅供参考）

Value： 属性值。如果未指定值，则存在一个空字符串。该值可以是以下类型：整数；一串预定义选项（下拉列表），包括逻辑类型（#true/#false）；文件路径

Resolved value： 如果值以 ${variable} 的形式指定，则此列包含变量的值；否则，它与值相同。

特性描述将显示在工具提示中。

5.7.4 Commands view

该视图允许指定将存储在 SB 文件中并且必须在目标处理器中执行的命令的顺序。有两种类型的命令：高级命令和低级命令。高级命令序列是通常在 SB 文件中使用的低级命令序列。这些命令简化了 SB 文件的修改，并将在生成配置 YAML 文件期间被低级命令替换。所有高级命令的名称均以“$”开头。高级命令没有任何参数，并且内部命令的参数不能修改。然而，总是可以用一组相应的低级命令替换高级命令，然后自定义低级命令。高级命令可以包含一个可选的低级命令，只有在定义了低级命令的所有变量时，该命令才会存储到 YAML 中。如果在某些配置中可能扩展高级命令，则使用此选项。使用高级命令而不是低级命令，因为高级命令会随配置更新。

命令页面包含以下面板：

The command sequence： 要生成到 SB 文件中的命令列表。可以选择是否显示 $ 变量或实际值。

Buttons bar： 该栏用于通过以下按钮控制命令序列：

按钮	说明
	**Add**- 将所选命令添加到当前序列中。
	**Delete**- 从当前序列中删除所选命令。
	**Move**- 在当前序列中向上或向下移动所选命令。
	**Expand**- 将高级命令序列扩展到低级命令列表中。

All available commands： 包含所有可用命令的树分为“高级命令序列”和“低级命令”。

Variables： $ 变量表可作为命令参数使用。可以选择该表是否包含所有变量或特定于命令的变量（特定于属性的变量将不显示）。

Selected command： 此面板显示所选命令的参数。

Command parameters： 所选命令的参数可以在命令页面底部的表中指定。第一列表示命令本身，其他列表示命令参数。如果参数是必需的，则不指定会导致错误。参数的描述将显示为工具提示。参数值使用 `${variable}`，因为这些变量在 SEC 工具配置中更新。

对于高级命令，将显示信息性描述而不是参数。

5.7.5 $ Variables

为了允许为SB文件生成重新使用SEC工具配置，提供了 $ 变量。这些变量的名称由 $ 字符组成，名称用大括号括起来（例如，${family}）。变量用于特性和命令参数。对于属性， $ 变量名称与属性的名称相匹配。所有变量的列表可以在右侧的命令页面上找到。可以过滤变量以隐藏 “properties” 的变量或显示所有变量。变量不可编辑；但是，可以将名称或值复制到剪贴板中。

5.7.6 Validation

所有输入值都经过验证，任何问题都会显示在窗口中。在生成 SB 文件之前，使用 JSON 模式验证 YAML 文件。如果任何验证失败，则无法生成 SB 文件。

建议使用 SB 文件创建 SEC 工具 workspace，将工作配置导入 SB 编辑器并开始更新工作配置。

5.7.7 Creating a manufacturing package button

如果SB文件应用于另一台计算机（或工厂），则可以创建包含所有所需文件的制造包。该程序包可以在另一台计算机上导入，请参阅 Manufacturing workflow。

5.7.8 To Manufacturing button

该按钮允许打开制造工具并将创建的 SB 文件并行应用于多个处理器，而无需创建制造包。

5.8 Merge Tool

合并工具允许用户将最多 8 张用户提供的镜像合并成一张二值镜像。

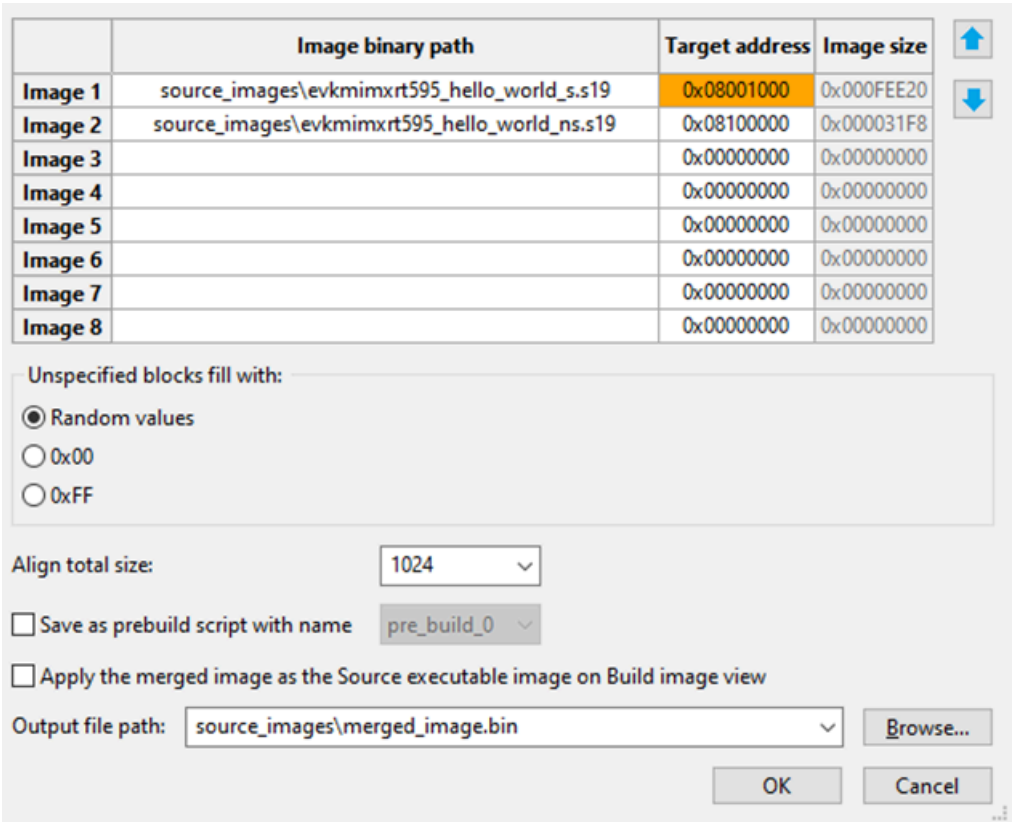

Figure 46. 合并 **image** 工具

该配置允许用户指定：

Images： 用于指定每个要合并的镜像的路径和目标地址的表格

Fill pattern： 用于填充镜像之间空白的图案，例如随机值、0x00 或 0xFF

Align total size : 对齐整个合并镜像的尺寸

Pre-build script option： 按下 **OK** 按钮时执行合并脚本，但可以将其存储为给定的预构建脚本，以便在构建脚本之前执行。

Apply the merged image as the source executable image： 合并脚本完成后，允许将合并后的镜像作为源可执行镜像应用于“Build image”视图。

Output file path： 合并的二进制镜像将存储在磁盘上的路径

6 Processor specific workflows

本章将指导完成设备安全启动全流程。其中包括bootable image 的创建、如何连接 MCU、如何设置引导首选项以及如何将 image 写入选定的引导设备几个部分。本章首先描述常规步骤，然后介绍特定 MCU 的内容。假设所有 image 的启动都基于 NXP 开发板。

本章介绍了下列工具链下的image生成：

- MCUXpresso IDE 11
- Keil Microcontroller Development Kit (MDK) 5 µVision
- IAR Embedded Workbench 8

- CodeWarrior Development Studio
- MCUXpresso for Visual Studio Code (VS Code)

接下来您将学习如何：

- 从 MCUXpresso 获取特定芯片 SDK 例程
- 使用工具链打开特定芯片例程
- 首先使用 SEC 工具
- 为已认证镜像准备非对称密钥
- 使用所选工具链生成 plain image
- 使用 SEC 工具生成 bootable image
- 连接开发板
- 将bootable image 写入处理器并（可选）保护处理器并延长生命周期

6.1 Common steps

本节提供该过程的常规步骤。

6.1.1 Downloading MCUXpresso SDK

MCUXpresso 提供的 SDK 例程包括开源驱动程序、中间件和参考例程，为用户开发提供便利。本节将会介绍如何下载 ZIP 包或 CMSIS 包形式的 MCUXpresso SDK，然后介绍如何从 SDK ZIP 包中打开例程。建议从 **iled_blinky** 例程开始。因为它提供了一个简单的检查方法，可以确认生成的应用程序是否正常工作 —— LED 指示灯以 1 秒的周期闪烁。

- 下载适用于 **MCUXpresso** IDE、VSCode 或 **CodeWarrior Development Studio** 的 **MCUXpresso SDK** 包
  1. 访问 MCUXpresso SDK Builder。
  2. 选择开发板。
  3. 为所选工具链构建 SDK 包并下载。 注意： 建议使用 MCUXpresso IDE v11.1.0，可以直接下载和使用 SDK 例程包。
- 下载**MCUXpresso CMSIS SDK** 例程包
  如果使用工具链 MDK μVision 和 IAR Embedded Workbench，可以下载所选处理器或开 发板的 CMSIS SDK 例程包。
  – Device Family Pack (DFP): *NXP.{processor}_DFP.#.#.#.pack*
  – Board Support Pack (BSP): *NXP.EVK-{processor}_BSP.#.#.#.pack*
- 为 **Keil MDK**或**IAR Embedded Workbench** 下载 **SDK** 例程包
  对于使用 Keil MDK 或者 IAR Embedded Workbench 来说，可以只下载某个例程。首先在 MCUXpresso SDK Builder上生成 SDK 例程包，然后点击下载链接并选择 **Download Standalone Example Project**。该例程包含所有需要的源和项目文件。

6.1.2 Opening example project

- **MCUXpresso IDE**
  1. 使用鼠标将需要下载的 MCUXpresso SDK 例程包拖拽到 Installed SDKs 界面进行安装。
  2. 选择 **File > New > Import SDK examples…**。
  3. 选择处理器和开发板型号，然后选择 iled_blinky 例程。
- 使用 **Keil MDK 5** 打开例程包
  1. 解压 SDK 例程包，然后双击打开 *boards\evkmimxrt10##\demo_apps\led_blinky\mdk\iled_blinky.uvmpw*。
  2. 如果只下载了一个单独的例程，请直接将其解压缩并打开 .uvmpw 文件。

3. 单击 **Project > Options > Output**，勾选 **Create HEX File**。

- 使用 **Keil MDK 5** 打开 **CMSIS** 例程包
  1. 选择 **Project > Manage > Pack Installer**。
  2. 在 **Devices** 里，选择 **All Devices > NXP > MIMXRT10##**。
  3. 在 **Packs** 里，安装 *NXP::{processor}#_DFP*和 *NXP::EVK-{processor}_BSP* 包。
  4. 选择BSP包。
  5. 在 **Examples** 里，拷贝 **iled_blinky** 例程到选定目录。
  6. 单击 **Project > Options > Output**，勾选 **Create HEX File**。
- 使用 **IAR Embedded Workbench** 打开 **MCUXpresso SDK** 例程包
  1. 解压 SDK 例程包，然后双击打开 *boards\evkmimxrt10##\demo_apps\led_blinky\iar\iled_blinky.eww*。
  2. 如果只下载了一个单独的例程，请直接将其解压缩并打开 .eww 文件。
- **CodeWarrior Development Studio**
  1. 将 SDK 包解压到所选文件夹中，并在 **Commander** 窗格中 **import project**。
  2. 如果使用 MCUXpresso 配置工具创建项目模板，请按照第 1 点中所述导入此项目模板。
- **MCUXpresso for Visual Studio Code**
  1. 将 SDK 包解压到选定的文件夹中。
  2. 使用命令 `> MCUXpresso for VSCode: Import Local/Remote repository`；选择 local，选择 SDK 文件夹路径。
  3. 使用命令 `> MCUXpresso for VSCode: Import Example Application from Installed Repository`；填写所有项目并单击 **Create** 确认。

6.1.3 Building example project

有关项目配置和生成的详细信息，请参见下面特定芯片章节。为了快速评估，安装布局中提供了来自 NXP 评估板 SDK 示例的预构建应用程序 image。详细信息，请参考安装子文件夹 **<SEC>sample_data/targets/{processor}/source_images**。

6.1.4 Setting up Secure Provisioning Tool

1. Secure Provisioning Tool - 安全配置工具
   - Windows：双击桌面快捷图标或者在开始菜单中点击该工具。
   - MacOS：单击停靠栏中的快捷图标或者使用启动台打开该工具。
   - Linux：单击快捷栏中的图标或者使用命令行窗口打开该工具。
2. 从 菜单栏 中选择 **File > New Workspace ...**。选择新工作区的路径。然后选择目标处理器并单击 **Create**。
3. 使用 USB、UART、SPI、或 I2C 连接设备。
4. 从 菜单栏 中选择 **Target > Connection ...**，然后单击 **Test** 来测试连接是否正常。必要时进行调整。

6.1.5 Preparing secure keys

本节介绍创建经过认证或加密的 image 所需的非对称密钥的生成。此操作只执行一次，生成的密钥可用于所有用例。

1. 进入 **PKI management** 界面 。
2. 确保 workspace 目前没有包含任何密钥。
3. 单击 **Generate keys**。
4. 在 **Generate keys** 的弹出框中，确认默认配置并单击 **Generate**。

注意： 生成的密钥位于 **keys/** 子文件夹中，证书（如果有）位于 **crts/** 子文件夹中。建议在烧写到 fuses 之前将生成的密钥进行备份。

6.1.6 Build and write

构建和写入可启动 image 的步骤因处理器而异，将在下文针对每个处理器的章节中列出。在此工作流程中，建议写入构建时准备的 image；SEC 工具也支持写入外部构建的 image，但配置可能需要额外的配置和更深厚的经验。

6.1.6.1 BCA and FCF pages

本节仅适用于包含Bootloader Configuration Area (BCA)和Flash Configuration Field (FCF)页面的处理器，例如 MCXC 系列。

BCA 和 FCF 配置块存储在 flash 中，并且是可启动 image 的一部分。可以通过 BCA 和 FCF 页面，使用 BCA/FCF 配置对话框来配置这些模块。

BCA 和 FCF 页面不会设置到可启动 image 中，并且原始内容保持不变，除非用户对页面有任何要求。如果设置了任何要求，则写入整个页面，没有用户值的字段将设置为默认值。在这种情况下，请确保源 image 中设置的任何值不会被无意地用默认值覆盖。

注意： 当生命周期设置为安全 flash 状态时，即使没有用户要求，FCF 也会设置到 bootable image 中。

在 BCA/FCF 配置对话框中，当前字段值可以从两个来源读取：

1. 来自所连接处理器的 flash。
2. 从 **Build**页面上选择的源 image。

6.2 i.MX 9x device workflow

本节详细介绍 i.MX 95 和 i.MX 95 设备工作流程。

6.2.1 Preparing images for build for i.MX 9x devices

在此步骤中，必须选择将要执行 image 的目标存储器。

i.MX 9x 设备具有以下可执行 image 的内核：

- i.MX 93：
  - Cortex-M33，启动内核
  - Cortex-A55，启动内核
- i.MX 95;
  - Cortex-M33，启动内核
  - Cortex-M7
  - Cortex-A55

以下目标存储器可用于 i.MX 9x 设备：

- image 运行在 RAM 中

该 image 可以被放置到 SD 卡/eMMC/FlexSPI 或者外部 flash（FlexSPI NOR，FlexSPI NAND 和 SEMC NAND）中，运行时先复制到 RAM 中再引导执行。支持以下 RAM 类型：

- 内部 RAM
- SDRAM (DDR SDRAM)

构建包含 AHAB 容器集的bootable image 需要多个 image：

- i.MX 93 images：
  - 主 image 容器集的 image：
    - ELE 固件
    - 带有 U-Boot SPL 的 LPDDR4 固件文件
    - [可选] Cortex-M33应用
  - 次 image 容器集的 image：
    - ARM 可信固件（bl31 二进制）
    - U-Boot
    - [可选] TEE 二进制
- i.MX 95 images：
  - 主 image 容器集的 image：
    - ELE 固件
    - 带有 OEI DDR 固件的 DDR（LPDDR4 或 LPDDR5）固件文件
    - CM33 OEI TCM
    - CM33 系统管理器
    - U-Boot SPL
    - [可选] Cortex-M7应用
  - 次 image 容器集的 image：
    - ARM 可信固件（bl31 二进制）
    - U-Boot
    - TEE 二进制

对于 Cortex-M7 和 Cortex-M33 应用程序的 image，请使用 MCUXpresso SDK 示例。无需修改默认配置。该 image 是为在内部 RAM 中运行而构建的。

DDR 固件文件和 ELE 固件可以从 Yocto Project 下载：

- ELE 固件示例：ELE firmware
- DDR 固件文件示例：DDR firmware

有关软件包的详细信息，请参阅 *i.MX Linux Release Notes*（文档RN00210）。

其余 image 是根据源代码构建的。请使用 NXP i.MX main repository中的以下存储库：

- OEI DDR 固件，CM33 OEI TCM：OEI DDR firmware
- CM33 系统管理器：CM33 system manager repository
- U-Boot SPL, U-Boot：U-Boot SPL, U-Boot repository，更多详情，请参阅Build U-Boot with AHAB secure boot features。
- ARM 可信固件（bl31 二进制）：ARM trusted firmware
- TEE 二进制文件：TEE Binary Repository

有关其他 Linux 发行版材料，请参阅 Embedded Linux for i.MX Applications Processors

有关示例详情，请参阅 **main menu > Help > SPSDK Online Documentation**。

- 示例 - AHAB - 带 U-Boot 的 i.MX 95 AHAB
- 示例 - AHAB - i.MX 93 签名并加密的 AHAB image

6.2.2 Connecting the board for i.MX 9x devices

本节包含有关配置评估板并将其连接到 SEC 工具的信息：

- IMX95LPD5EVK-19（IMX95LPD5BB-19 底板，带 IMX95LP5CPU-19 CPU 板）
- MCIMX93-EVK（MCIMX93-BB 底板，带 MCIMX93-SOM CPU 板）
- MCIMX93-QSB

6.2.2.1 Table: Boot mode selection for board for i.MX 9x Cortex-M33 core

启动模式/设备	串行引导程序（ISP 模式）	eMMC	SD 卡	FlexSPI NOR
IMX95LPD5EVK-19	SW7: 1001	SW7: 1010	SW7: 1101	SW7: 1100 (N/A)
MCIMX93-EVK	SW1301: 1101	SW1301: 0001	SW1301: 0101	SW1301: 1011（不适用）
MCIMX93-QSB	SW601: 1001	SW601: 1010	SW601: 1011	SW601: 1100（不适用）

6.2.2.2 Table: Boot mode selection for board for i.MX 9x Cortex-M33 core

启动模式/设备	串行引导程序（ISP 模式）	eMMC	SD 卡	FlexSPI NOR
MCIMX93-EVK	SW1301: 1100	SW1301: 0000	SW1301: 0100	SW1301: 1010（不适用）
MCIMX93-QSB	SW601: 0001	SW601: 0010	SW601: 0011	SW601: 0100（不适用）

6.2.2.3 Step by step process

1. 有关如何使用 DIP 开关设置启动模式的说明，请参阅 Connecting the board for i.MX 9x devices。 中的 **DIP 开关：Cortex-M33** 主板的启动模式选择 或 **DIP 开关：Cortex-A55** 主板的启动模式选择。
2. 使用 USB 电缆将 USB1/USB 端口连接到您的 PC 以获取下载链接。
3. 使用 USB 电缆将 DBG 端口连接到您的 PC 以获取控制台输出。
4. 将开发板连接到电源 JACK/USB PD 并打开电源开关。
5. 确认 SEC 工具已使用为所选设备创建的工作区运行。更多详情，请参阅 Setting up Secure Provisioning Tool。
6. 打开 **Connection** 对话框并测试连接。

从 **SD** 卡中启动

从 SD 卡中启动需要以下两个步骤：

1. 将 micro SDHC 卡插入开发板中。
2. 在 **Boot Memory Configuration** 中选择 **SD card, SDHC SD-card 64 GB**。

从 **eMMC** 启动：

要从 eMMC 启用，请执行以下操作：

1. 检查连接的主板是否已包含 eMMC 64 GB。
2. 选择 eMMC：在 **Boot Memory Configuration** 中选择 **SDHC eMMC 64 GB**。

6.2.3 Booting images for i.MX 9x devices

本章描述了 bootable images 的生成和写入过程。

6.2.3.1 i.MX 93 bootable image examples

- 包含 Cortex-M33 应用程序的容器集。它在写入过程中由 nxpuuu 工具写入片上 RAM，然后从那里开始。

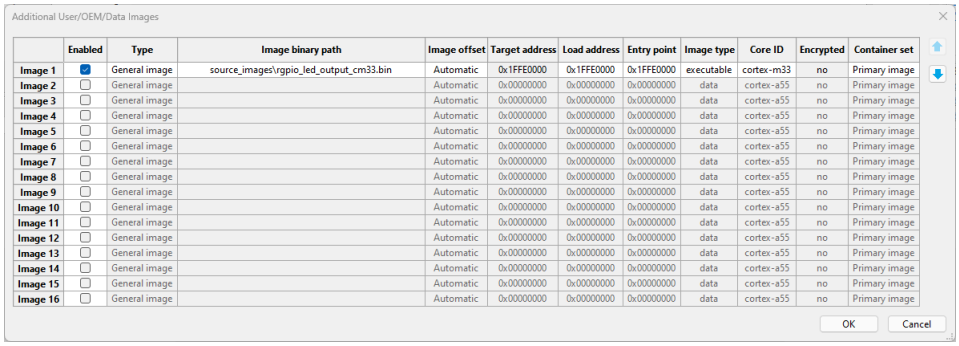

Figure 47. Cortex-M33 应用程序容器集的附加 image

- 带有 Cortex-A55 U-Boot（bootloader）的容器集。写入时通过 nxpuuu 工具写入 eMMC/SD 卡。在引导过程中，该 bootloader 用于引导 Linux 内核映像。

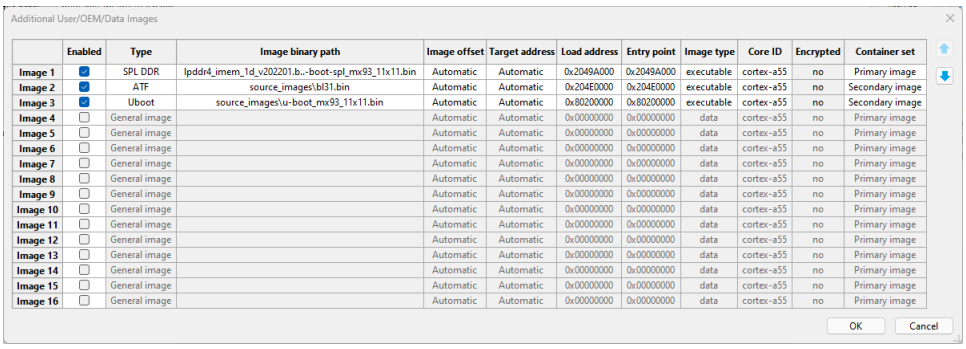

Figure 48. 带有 Cortex-A55 U-Boot 的容器集的附加 image

6.2.3.2 i.MX 95 bootable image examples

- 包含 Cortex-M7 应用程序的容器集。它在写入过程中由 nxpuuu 工具写入片上 RAM，然后从那里开始。

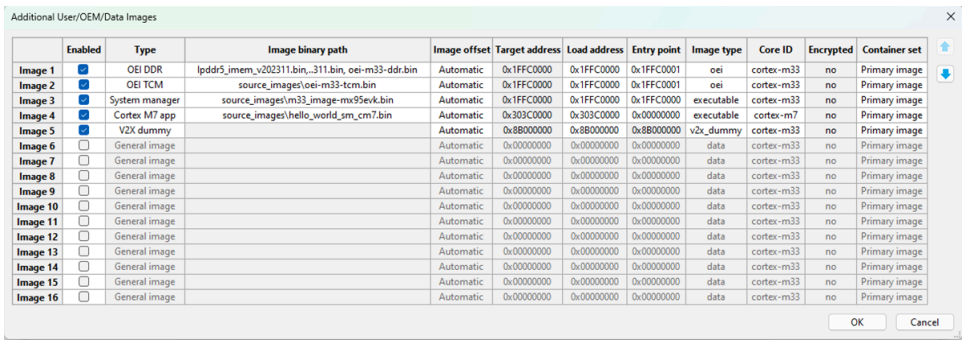

Figure 49. Cortex-M7 应用程序容器集的附加 image

- 带有 Cortex-A55 U-Boot（bootloader）的容器集。写入时通过 nxpuuu 工具写入 eMMC/SD 卡。在引导过程中，该 bootloader 用于引导 Linux 内核映像。

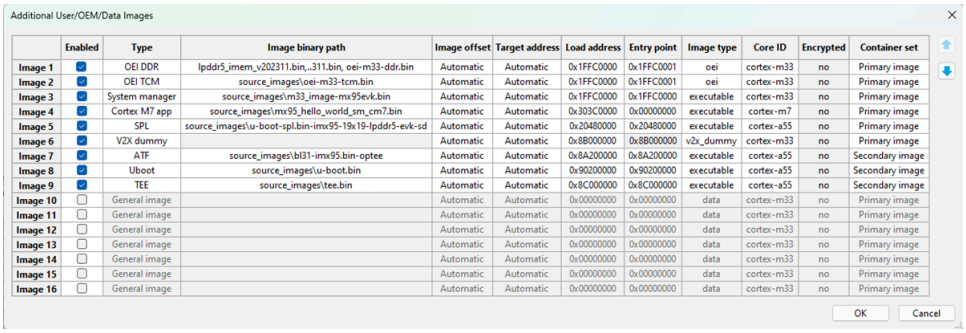

Figure 50. 带有 **Cortex-A55 U-Boot** 的容器集的附加 **image**

6.2.3.3 Booting/loading unsigned image

首先，生成一个 bootable image：

1. 在工具栏中，选择**Unsigned boot type**。
2. 在工具栏中，选择**boot device**。
3. 切换到 **Build image** 视图。
4. 单击**Additional images**按钮以打开**Additional User/OEM Image**对话框。
5. 配置 image 容器中的 image，请参阅 i.MX 93 bootable image examples 或者 i.MX 95 bootable image examples。从 Preparing images for build for i.MX 9x devices 中选择已准备的 image。
   - 启用**Enabled**列中的 image。
   - 使用**Type**选择 image 条目
   - 选择**Image binary path**中的二进制文件。
   - 选择**Container set**，它应该是主 image 或辅助 image。
   - 所有其他参数均预设为所选 image 输入类型的默认值。这些参数可以定制。
6. 单击 **OK**关闭对话框。
7. 单击 **Build image**按钮以生成一个 bootable image，`flash.bin`.。

bootable image 生成成功后，请执行以下操作：

1. 确保该开发板为串行引导加载程序 (ISP) 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
   - nxpuuu 工具用于将bootable image 加载到设备。
4. 如果写入操作执行成功：
   - 对于 **eMMC** 或 **SD** 卡，切换启动模式（请参阅Connecting the board for i.MX 9x devices中的 **Cortex-M33** 主板的启动模式选择 或 **Cortex-A55** 主板的启动模式选择，然后重置板。应用程序应在 Cortex-M7 中`hello_world`和/或 Cortex-A55 中的 U-Boot 运行。
   - 对于**Onchip RAM**，应用程序写入后在 Cortex-M7/Cortex-M33 中运行。

6.2.3.4 Booting signed image (bootloader for Cortex-A55) for i.MX 93

本章节描述了带签名的 image 的生成和写入过程，i.MX 93 的 Cortex-A55 引导加载程序。此步骤需要在 **PKI management** 视图进行密钥生成。对于更多关于密钥生成的信息，请参阅Generate keys。

首先，构建一个bootable image（Cortex-A55 的引导加载程序）：

1. 在工具栏中，设置**Boot type**为**Signed**。
2. 在工具栏中，选择**boot device**。
3. 切换到 **Build image** 视图。
4. 打开 **Additional User/OEM Image** 对话框并配置引导加载程序的 image。在 i.MX 93 bootable image examples 和 Build U-Boot with AHAB secure boot features 中查阅 **Additional images for Container set with Cortex-A55 U-Boot** 的示例配置。
5. 对于 **Authentication key**，选择任意密钥，例如 *SRK1*。
6. 勾选生命周期为 **OEM Open**或 **OEM Closed**。欲了解如何在 OEM Open 生命周期中验证应用程序，请参阅 Get ELE events with nxpele utility。
7. 单击 **Build image**。
8. 检查 bootable image 是否成功生成。

bootable image 生成成功后，请执行以下操作：

1. 确保该开发板为串行引导加载程序 (ISP) 模式。
2. 切换到 **Write image** 视图。
3. 确保 **Use build image** 已被选择。这会将构建的引导加载程序写入选定的引导设备。
   注意： 要写入另一个准备好的bootable image、另一个引导加载程序或带有 Linux 内核的完整 image，请取消选中复选框并选择您的 image。在写入过程中仍然使用基于 **Build image**构建的引导加载程序来写入 fuse 并使用 nxpele 工具更新生命周期。
4. 单击 **Write image**。
   - nxpuuu 工具用于将带有 U-Boot 的引导加载程序加载到 RAM。
   - nxpele工具用于写入 fuse 和更新生命周期。
   - nxpuuu 工具用于将引导加载程序/您准备的bootable image 加载到设备。
5. 在下列弹出框中，确认写入 fuses并更新生命周期：
   - **OK** — 确认并烧写 image 和 fuses。
   - **Cancel** — 取消烧写 image 和 fuses。

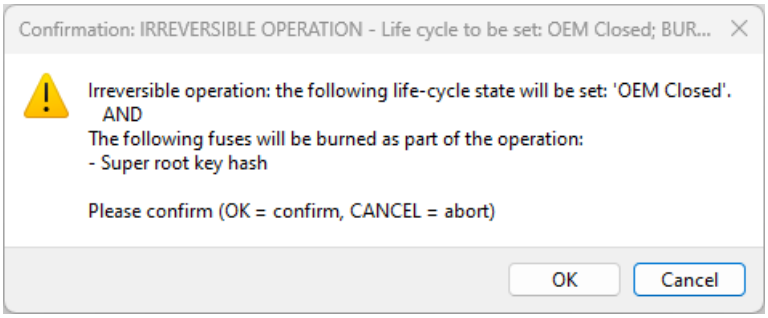

Figure 51. i.MX 93 上烧写 fuses

1. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for i.MX 9x devices中的**i.MX 9x Cortex-A55** 主板的启动模式选择）并复位开发板。应用程序应在 Cortex-A55 中运行 U-Boot/您准备的bootable image。

6.2.3.5 Write fuses and update life cycle with nxpele over U-Boot

nxpele工具用于写入 fuse 和更新生命周期。它还可用于读取 fuses **OTP Configuration** 对话框。nxpele 工具通过 U-Boot 与 EdgeLock Enclave 通信，U-Boot 必须在目标上运行。

在这些情况下，**Additional User/OEM Image** 对话框中配置的bootable image 将用作 U-Boot 的引导加载程序。U-Boot 必须使用 AHAB 功能构建，更多详情，请参阅 Build U-Boot with AHAB secure boot features。在设备配置期间，引导加载程序被加载到目标并启动。

6.2.3.6 Build U-Boot with AHAB secure boot features

U-Boot/SPL https://github.com/nxp-imx/uboot-imx 提供了额外的安全启动功能。这些功能使 nxpele 工具能够与 EdgeLock Enclave 通信。通过在构建中设置 `CONFIG_AHAB_BOOT=y` 来启用支持。

如果使用 fastboot 上的 nxpele 工具，则必须通过设置`CONFIG_CONSOLE_MUX=y`启用控制台输出到 fastboot 的多路复用。

6.2.3.7 Get ELE events with nxpele utility

`nxpele get-event` 命令可以检索 EdgeLock Enclave 中存储的事件。nxpele 工具通过 U-Boot 与 ELE 通信，U-Boot 必须在目标上运行。

例如，借助这一点，我们可以检查 OEM Open 生命周期中的容器身份验证状态：

1. 请参阅 Booting signed bootloader image from Cortex-A55 core on i.MX 93。 在此过程中，保持 **OEM Open** 中的生命周期。在OEM Open生命周期中，认证结果被忽略。
2. 将 DBG 端口连接到您的 PC。
3. 切换至 ISP 模式并复位开发板。
4. 检查 U-Boot 控制台输出的串行 COM 端口是什么。
5. 如果在终端中打开了串行连接会话，请关闭它。
6. 执行 `nxpele get-events`

认证成功示例：

```
C:\nxp\SEC_Provi_25.09\bin_internal\tools\spsdk>nxpele -f mimx9352 -p COM21 get-
events
ELE get events ends successfully.
Event count:     0
```

认证失败示例：

```
C:\nxp\SEC_Provi_25.09\bin_internal\tools\spsdk>nxpele -f mimx9352 -p COM21 get-
events
ELE get events ends successfully.
Event count:     2
Event[0]:      0x0287FAD6
  IPC ID:        Application Processor message unit
  Command:       OEM Container authenticate
  Indication:    The key hash verification does not match OTP
  Status:        The request was successful
Event[1]:      0x0287FAD6
  IPC ID:        Application Processor message unit
  Command:       OEM Container authenticate
  Indication:    The key hash verification does not match OTP
  Status:        The request was successful
```

6.3 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx device workflow

本章介绍 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 处理器的工作流程。

6.3.1 Preparing source image for KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx devices

在此步骤中，必须选择将要执行 image 的目标存储器。KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 设备提供以下选项：

- 从内部 flash 运行的 image（XIP image）

无需修改默认配置，可以照原样构建 MCUXpresso SDK 示例。

6.3.2 Connecting the board for KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx devices

本章节包含如何配置以下型号的评估板及如何将其连接到 SEC 工具的示例：

- KW45B41Z-EVK
- K32W148-EVK
- MCXW71-FRDM
- MCXW71-EVK
- KW47-EVK

假设 SEC 工具已经打开运行，并已为 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 设备创建 workspace。更多详情，请参阅 Setting up Secure Provisioning Tool。

对于 KW45B41Z-EVK 和 K32W148-EVK 开发板：

1. 使用USB电缆将J14端口连接到您的PC。
2. 设置JP25跳线以启用SW4按钮。
3. 按住SW4按钮并重置，以启用ISP启动模式。
4. 在**Connection**对话框中，测试与处理器的连接。

对于 MCXW71-FRDM 开发板：

1. 使用 USB 电缆将 J10 端口连接到您的 PC。
2. 按住SW3按钮并重置，以启用ISP启动模式。
3. 在**Connection**对话框中，测试与处理器的连接。

对于 MCXW71-EVK 开发板：

1. 使用USB电缆将J14端口连接到您的PC。
2. 按住SW4按钮并重置，以启用ISP启动模式。
3. 在**Connection**对话框中，测试与处理器的连接。

对于KW47-EVK开发板：

1. 使用USB电缆将J14端口连接到您的PC。
2. 按住SW4按钮并重置，以启用ISP启动模式。
3. 在**Connection**对话框中，测试与处理器的连接。

6.3.3 Booting images for KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx devices

本章描述了 bootable images 的生成和写入过程。对于KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx，SEC工具仅支持XIP images。

6.3.3.1 Booting plain or CRC image

明文（plain）image 通常用于开发。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。仅安全启动类型支持双 image 启动。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择了 **Plain unsigned**或**Plain with CRC** 启动类型。
2. 切换到 **Build image** 视图。

3. 选择在 Building example project 中构建的应用程序 image 作为源可执行 **image**。
4. 如果存在一个 binary image，请将起始地址设置为 0x0。
5. 单击 **Build image**按钮以生成一个 bootable image。结果是二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

如果写入操作执行成功，请复位开发板。

6.3.3.2 Booting signed image

本章节描述了带签名的 image 的生成和写入过程。

生成bootable image：

1. 在工具栏中，选择 **Plain signed** 启动类型 。
2. 切换到 **Build image** 视图。
3. 选择在 Building example project 中构建的应用程序 image 作为 源可执行 **image**。
4. 请确保在 PKI 管理页面中拥有密钥。KW45B41Z-EVK、K32W148-EVK、MCXW71-FRDM 和 KW47-EVK 评估板使用预编程的 ROKTH 和 SB3KDK 键在 `CUST_PROD_OEMFW_AUTH_PUK` 和 `CUST_PROD_OEMFW_ENC_SK` 中生产。这些键也分布在SEC工具中，可以从工具文件夹`sample_data\targets\<processor>\board_example_keys`中导入。请参阅 Import/Export keys。这些密钥仅用于评估目的，不得用于生产。
5. 对于 **Authentication key**，选择任意密钥，例如 ROT1: IMG1_1。
6. 为 **SB3KDK**对称密钥使用导入的值或创建您自己的（随机）值 。
7. 如果需要，打开 **Dual image boot** 并进行配置。image 必须连接到 **Flash Logical Window**。
8. 生命周期保持为 **OEM Open**。
9. 单击 **Build image**以生成一个 bootable image。结果是SB3封装体被安装到处理器中。

成功构建 bootable image 和 SB3 封装后，可以上载到处理器：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 生命周期保持为 **OEM Open**。
4. 请确保 otp_config.sb 中写入的 IFR 字段尚未被刻录。所有 IFR 字段均可一次性编程。
5. 单击 **Write image**。

注意： 如果otp_config.sb中写入了 IFR 字段，则无法多次接收otp_config.sb。

6.3.3.3 Booting PRINCE encrypted image

支持加密的未签名 image、带 CRC 的 image、或签名的 image。创建加密 image的过程类似于签名 image。此外，还可以在 **Build image** 视图中配置加密区域。使用 **PRINCE regions** 的按钮配置加密区域。结合双启动，为image0设置一个区域，为image1设置一个区域。仅为 image0 设置区域不会对 image1 进行加密。

当 image 被写入目标存储器时，执行 image 加密。

区域配置包含在ROMCFG页面中。

注意： 打开 OTP/IFR 配置以检查 ROMCFG 块中的 PRINCE 设置，因为该块必须完全指定并且只能写入一次。这个操作不可逆。

6.3.3.4 Life cycle for KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx devices

默认的生命周期应该用于开发，它是 **OEM Open**。部署应用程序之前，请设置 **OEM Closed** 或 **OEM Locked**生命周期（更多详情，请参阅目标处理器的文档）。

注意： 生命周期的变化是不可逆的。

一旦处理器处于 OEM Closed 或 OEM Locked 模式，该工具就不允许初始化 ROMCFG 页面。仍可通过 SB 文件来更新应用程序。

KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 的启动类型和生命周期

	OEM OPEN	OEM CLOSED/LOCKED
明文未签名或 **CRC** 启动类型	— 仅用户 fuse 被烧毁— 没有 SB 文件被使用	— 生命周期 fuse 被烧毁 — 没有 SB 文件被使用
明文已签名启动类型	— RKTH 和 SB3KDK 烧录脚本 — 写入脚本中的用户 fuse 或OTP SB文件— SB 文件被使用	— RKTH 和 SB3KDK 烧录脚本 — OTP SB 文件中的用户 fuse— SB 文件被使用
加密启动类型	— RKTH 和 SB3KDK 烧录脚本 — 加密写入脚本或 SB 文件中的用户 fuse — SB 文件被使用	— RKTH 和 SB3KDK 烧录脚本 — OTP SB文件中的用户 fuse 和 IFR 加密— SB 文件被使用

下表显示了不同生命周期和信任配置类型中安装的安全资产：

	开放生命周期，无需信任配置	封闭生命周期，无需信任配置	开放生命周期，**Edge Lock 2GO**	封闭生命周期，**Edge Lock 2GO**
SB3KDK	写入脚本	写入脚本	sb3kdk.asc	sb3kdk.asc
RKTH fuses	写入脚本	写入脚本	rkth.bin	rkth.bin
其余定制 **fuse** 和 **IFR**	写入脚本或otp_config.sb	otp_config.sb	otp_config.sb	otp_config.sb

6.3.3.5 EdgeLock 2GO

有关EdgeLock 2GO，请查阅EdgeLock 2GO Trust Provisioning workflow。以下是 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 的具体描述：

- 生命周期设置为安全对象所设置的生命周期。RKTH 和 SB3KDK fuse 在配置期间总是被烧毁，即使在开发生命周期中也是如此。
- 安全对象的地址必须位于内部 flash 中，配置固件会在写入安全对象之前擦除该区域。
- 对于 KW47 和 MCXW72，如 OEM_SEC_BOOT_EN 果安全对象设置为“OEM 关闭”或“OEM 锁定”生命周期，则配置固件会烧毁 fuse 。

6.3.3.6 Update NBU firmware

此功能仅适用于带有无线电模块的处理器。NBU 固件在 MCUxpresso SDK 中以 SB3 文件或二进制文件 (*.xip) 的形式分发，位于文件夹 `middleware\wireless\ble_controller\bin`。 有两种方式可以更新 NBU 固件。首先，使用准备好的带有 NBU 固件的 SB 文件。它可用于配备有 NXP 密钥的 EVK 和 FRDM 板。另一种方法是准备一个将加载 NBU 固件的自定义 SB 文件。

带有 **NXP** 密钥的 **SB** 文件

1. 为 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 创建或打开 workspace。

2. 在菜单栏上，选择**Tools > Manufacturing Tool**。
3. 选择**Apply SB file**。
4. 提供 SB 文件。
5. 单击**Auto detect**以检测连接的设备。
6. 单击**Start**以加载 SB 文件。

自定义 **SB** 文件

1. 使用为设备配置的自定义密钥为 KW45xx/K32W1xx/MCXW71xx/KW47xx/MCXW72xx 打开一个 workspace。
2. 将NBU固件复制到 workspace `source_images\nbu_firmware.xip`
3. 在菜单栏上，选择**Tools > SB Editor**。
4. 填写**Properties**页面或从 **Build** 页面上创建的 SB 文件导入设置。
5. 切换到 **Commands** 页面。
6. 选择高级命令 *update-nbu-firmware*
7. 单击**Generate**以准备最终的 SB 文件。
8. 单击 **To Manufacturing tool** 以切换到制造窗口，其中已预先选择 **Apply SB file** 操作和预先选择的 SB 文件。
9. 单击**Auto detect** 以检测连接的设备。
10. 单击开始以加载 SB 文件。

6.4 LPC55(S)0x/1x/2x/6x device workflow

本章介绍 LPC55(S)0x/1x/2x/6x，NHS52S04，和 MCXW236 处理器的工作流程。

6.4.1 Preparing source image for LPC55(S)0x/1x/2x/6x devices

在此步骤中，必须选择将要执行 image 的目标存储器。以下选项适用于LPC55Sxx设备：

- 从内部 flash 运行的 image（XIP image）

6.4.1.1 Image running from internal flash

- **MCUXpresso IDE**

1. 构建编译工程。
2. 打开 debug 文件夹。
3. 右键点击命名为 <your.project>.axf 的文件。
4. 选择 **Binary Utilities** > **Create binary**。

- **IAR**

1. 打开 **Project** > **Options** > **Output Converter**，选中 **Generate additional output** 并选择 **Raw binary output format**。

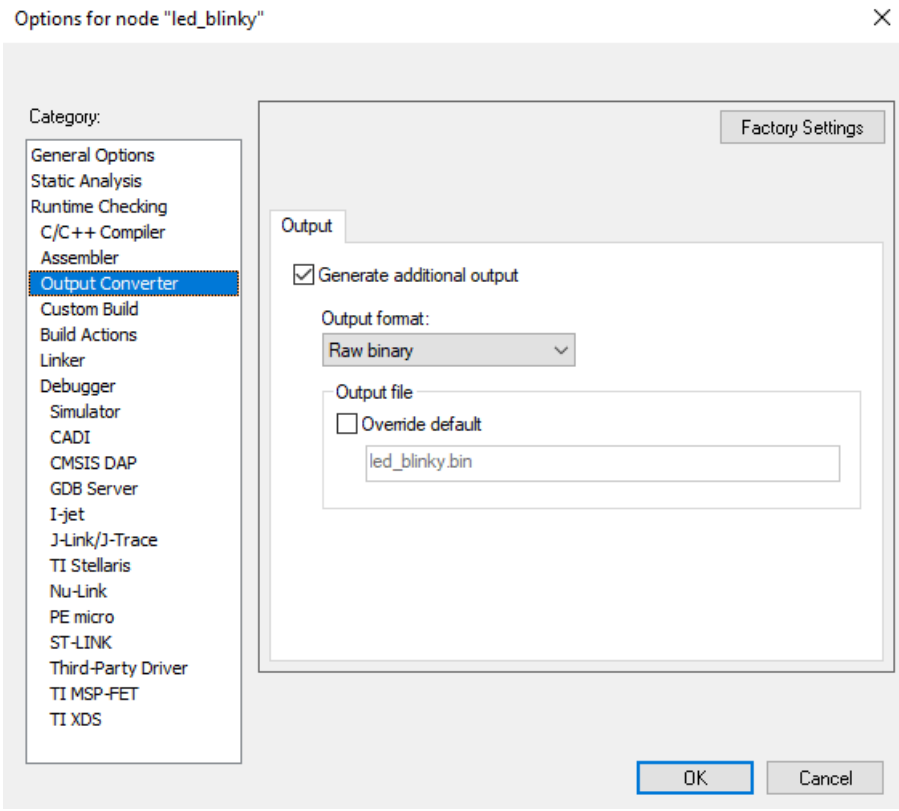

Figure 52. IAR 选项

1. 构建工程。输出 image 构建为 `boards\\lpc55s\#\#\\demo\_apps\\led\_blinky\\iar\\led\_blinky\\led\_blinky.bin`。

- **KeilMDK 5**

1. 打开 **Project** > **Options** > **User** > **After Build/Rebuild**，选中 **Run #1**。
2. 在 **User Command** 路径中输入下列命令（其中的 *myprog* 是可执行文件名称）：`C:\Keil\ARM\ARMCC\bin\fromelf.exe --bin --output=myprog.bin myprog.axf.`
3. 生成 image 文件。输出 image 构建为 `boards\lpc55s\#\#\\demo\_apps\led\_blinky\mdk\led\_blinky\led\_blinky.bin`。

6.4.2 Connecting the board for LPC55(S)0x/1x/2x/6x devices

本节将介绍如何配置以下 LPC5Sxx 评估板并将其连接到 SEC：

- LPCexpresso55S69
- LPCexpresso55S66
- LPCexpresso55S28
- LPCexpresso55S26
- LPCexpresso55S16
- LPCexpresso55S14
- LPCexpresso55S06
- LPCexpresso55S04
- NHS52Sxx-EVK
- FRDM-MCXW236B

假设 SEC 工具已经打开运行，并已为 LPC 设备创建 workspace。更多详情，请参阅Setting up Secure Provisioning Tool。

1. 如果选择使用串口通信，请将 USB 线插入 P6 接口。如果选择使用 USB 通信，请将 USB 线插入 P9 接口。
2. 按住 ISP 按键并复位，以进入 ISP 模式。
3. 确保在工具栏 中将 boot 类型选为 **Unsigned**。
4. 在 **Connection** 对话框，根据选择的通信方式，将连接设置为 **USB** 或 **UART**，并测试 工具和处理器的连接。

6.4.3 Booting images for LPC55(S)0x/1x/2x/6x devices

本节描述了 image 的构建和生成。

6.4.3.1 Security levels

SEC 支持以下安全等级：

未签名的启动类型： 默认的处理器配置不提供任何安全性。建议从无签名启动 类型开始，以确保处理器可以正确执行bootable image。无签名启动类型仅用于开发过程。

签名或加密的启动类型 — 非密封： 非密封启动类型同样被设计为在开发阶段使用，以确保选定的启动类型可以正常工作。KeyStore，CFPA 和 CMPA 被写入芯片中。CMPA 并未被密封锁死，可以被更新或擦除。

签名或加密的启动类型 — 密封： 建议制作密封的 CMPA 页面。在 **Write image** 视图中选择 **Deployment** 生命周期来密封 CMPA。一旦密封，它就不能被更改或擦除。

6.4.3.2 Booting Plain/Plain with CRC image

本节将描述明文（plain）和带有 CRC 的明文 image 的构建和生成。

1. 在工具栏上，设置 **Boot Type** 为 **Plain unsigned** 或 **Plain with CRC**。
2. 作为 **Source executable image**，使用 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 为 **Source executable image**。
3. 如果是二进制 image，请将起始地址设置为 *0x0*。
4. 如果需要，打开 **Dual image boot** 并进行配置。
5. 单击 **Build image**。
6. 检查 bootable image 是否成功生成。

成功构建 image 后，请执行以下操作：

1. 确保开发板已进入 ISP 模式。
2. 单击 **Write image** 视图。
3. 单击 **Write image**。

如果写入操作执行成功，请复位开发板。

6.4.3.3 Booting plain signed or PRINCE encrypted image

本节将描述认证或 PRINCE 加密 image 的构建和生成。 本步骤需要在此步骤需要在 **PKI management** 视图进行密钥生成。对于更多关于密钥生成的信息，请参阅Generate keys。

注意： 这些密钥也被用于 **PRINCE encrypted with CRC** 启动类型，因为使用 SB 包裹更新 bootable image 时，image 必须要被签名。

1. 在工具栏中，设置 **boot type** 为 **Plain signed**，**Encrypted (PRINCE) unsigned, Encrypted (PRINCE) with CRC**，或**Encrypted (PRINCE) signed**。
2. 在 **Build image** 视图上，使用 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 Source executable image。
3. 对于 **Authentication key**，选择任意密钥链，例如 *ROT1: IMG1_1*。
4. 打开PRINCE配置并检查配置。根据 bootable image 的大小设置 PRINCE 区域的大小。
5. 如果需要，打开 **Dual image boot** 并进行配置。对于 PRINCE 加密 image，为 image0 设置一个区域，为 image1 设置一个区域。仅为 image0 设置区域不会对 image1 进行加密。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 选择 **Write image** 视图。
2. 确保板子已连接且进入 ISP 模式（更多详情，请参阅Connecting the board for LPC55(S)0x/1x/2x/6x devices）。
3. 单击 **Write image**。
   如果写入操作执行成功，请复位开发板。

一旦 image 可以成功地在芯片中执行， 选择 **Deployment** 生命周期，以使用 CMPA 的 sha256 摘要永久地密封芯片。如果这个选项是未选中状态，则可以重新配置安全属 性。选择 **Deployment**生命周期，单击 **Build image**视图中的**Build image**。然后再次单击 **Write image**并确认以下消息框：

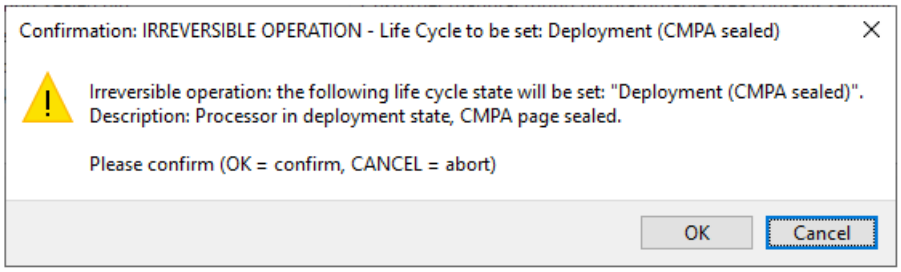

Figure 53. 确认写入

如果写入操作执行成功，请复位开发板。

注意： 从 PRINCE 加密返回到非加密 image 之前，必须彻底擦除整个处理器。

6.4.3.4 PFR and PUF KeyStore

本节提供有关 PFR 和 PUF 密钥库的信息。

PUF KeyStore initialization

SEC 在 LPC55Sxx 设备的生命周期中，仅进行一次 key store 的初始化。

单击 **Test** 按钮，**Connection** 对话框将报告设备的 key store enrollment 状态。

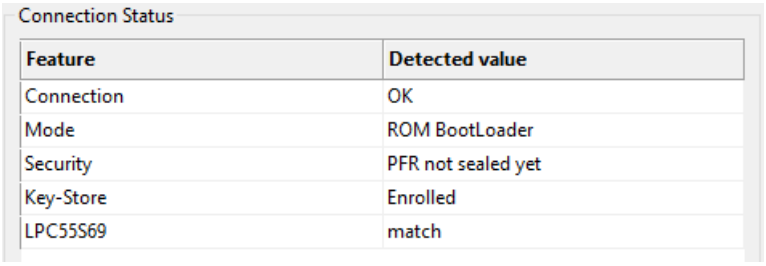

Figure 54. KeyStore 连接

如果需要更新 KeyStore，不需要重新初始化 KeyStore。如果出现意外问题，可以尝试擦除KeyStore，但不建议这样做。但是推荐同时也擦除 CFPA page，因为在 CFPA 中的 PRINCE IV 区域也依赖于 KeyStore。如果注册（enroll）KeyStore 但是尝试使用之 前的 IV 区域，芯片将不会启动。

如何擦除 **KeyStore**（以 **LPC55S69** 为例）

```
bin/tools/blhost/win/blhost -u 0x1FC9,0x0021 -j -- set-property 29 1

bin/tools/blhost/win/blhost -u 0x1FC9,0x0021 -j -- write-memory 0x9E600
 zero_1536.bin
```

zero_1536.bin 是一个内容全部为 0 的文件，并且文件的大小是 3*512 字节（byte）。

如何更新 **CFPA**（以 **LPC55S69** 为例）

1. `cfpa.json` 中的增量版本：建议从上一个已知版本中至少增加 0x10， 因为在 PRINCE IV 更新时版本号也会被增加。
2. 执行下列命令将 CFPA 更新到芯片中：

```
bin/tools/spsdk/pfr generate -c cfpa.json -o cfpa.bin
bin/tools/spsdk/blhost -u 0x1FC9,0x0021 -j -- write-memory 0x0009DE00
 cfpa.bin
```

DCFG_CC_SOCU LPC55Sxx 上的问题

LPC55S0x/1x/2x/6x 处理器不支持 CFPA 中配置的 `DCFG_CC_SOCU` 字段，而 CMPA 中的 `DCFG_CC_SOCU` 字段为零（尚未配置），如果配置发生，处理器可能会停止工作（锁定）。建议始终先配置 CMPA，然后擦除 CFPA（+重置），以防需要擦除 CMPA。

6.5 LPC55(S)3x device workflow

6.5.1 Preparing source image for LPC55(S)3x devices

在此步骤中，必须选择将要执行 image 的目标存储器。以下选项可用于 LPC55(S)3x 设备：

- 从内部 **flash** 运行的 **image** - XIP (eXecution In Place) image，这意味着该 image 直接从其所在的内存中执行。
  对于几乎所有的 SDK 示例，这是缺省选项。无需修改缺省配置，可按原样构建示例。
- 从外部 **flash** 运行的 **image** - XIP (eXecution In Place) image，这意味着该 image 直接从其所在的内存中执行。
  image 必须从地址 0x8001000 开始。目前不支持其他位置。无需修改默认配置。
  对于定制的外部 FLASH，可在 CMPA 中调整用于启动的外部 FLASH 配置。

6.5.2 Connecting the board for LPC55(S)3x devices

本节将介绍如何配置 LPC55S36-EVK 评估板并将其连接到 SEC：

1. 选择 ISP 启动模式，请参阅 **LPC55S36 EVK** 板的启动模式选择。
2. 使用 USB 电缆将 J3 端口连接到您的 PC。
3. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅 Setting up Secure Provisioning Tool。
4. 确保工具栏中的引导设备与 EVK 主板上使用的 NOR FLASH（例如，flex-spi-nor/ISxxxx）或内部 image 匹配。
5. 选择连接方式为 USB ，然后测试是否连接上。

6.5.2.1 Table: Boot mode selection for LPC55S36 EVK board

板	**ISP** 启动模式	从内部 **FLASH** 启动	从外部 **FLASH** 启动
LPC55S36-EVK	**J43:** 1-2 开 3-4 关	**J43:** 1-2 关 3-4 关	**J43:** 1-2 关，3-4 开

6.5.3 Booting images for LPC55(S)3x devices

本章描述了 bootable images 的生成和写入过程。对于 LPC55S3x，高效密码学标准 (SEC) 工具仅支持 XIP image。

6.5.3.1 Booting plain unsigned or CRC image

明文（plain）image 通常用于开发。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择了 **Plain unsigned**或**Plain with CRC** 启动类型。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for LPC55(S)3x devices生成的 image 作为 **Source executable image**。
4. 如果存在二进制 image，请将内部 flash 的起始地址设置为 0x0，将外部 flash 的起始地址设置为 0x8001000。
5. 如果需要，打开 **Dual image boot** 并进行配置。
6. 单击 **Build image**按钮以生成一个 bootable image。结果是二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for LPC55(S)3x devices中的**LPC55S36 EVK** 板的启动模式选择）并复位开发板。

6.5.3.2 Booting plain signed image

本章节介绍如何构建和编写明文签名图像。

生成 bootable image：

1. 在工具栏中，选择 **Plain signed**启动类型 。
2. 切换到 **Build image** 视图。

3. 选择在 Preparing source image for LPC55(S)3x devices中生成的 image 作为 **Source executable image**。
4. 对于 **Authentication key**，选择任意密钥，例如 ROT1: IMG1_1。
5. 对CUST_MK_SK和OEM seed对称密钥，使用随机值。
6. 如果需要，打开 **Dual image boot** 并进行配置。
7. 打开PFR配置，并在CMPA页面检查中配置`SECURE_BOOT_CFG`字段中的位字段 `SEC_BOOT_EN` 。必须选择任何类型的image检查。
8. 确保主板已连接且处理器处于 ISP 模式。在构建过程中，准备预配SB文件以安装 `CUST_MK_SK` 到处理器中。

   注意： SB 文件构建完成后，处理器会重置。
9. 保持生命周期的取值为 **Develop**
10. 单击 **Build image**按钮以生成一个 bootable image。结果生成一个二进制bootable image以及用于将该image安装到处理器的SB3封装。

成功构建 bootable image 后，可以上载到处理器：

1. 确保处理器处于 ISP 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

6.5.3.3 Booting encrypted image

支持带有CRC或签名 image 的加密 image。创建加密 image 的过程与签名 image 类似。此外，在 **Build image** 视图中配置加密区域：

- 使用 **PRINCE Regions** 按钮为内部FLASH配置加密区域
- 使用 **IPED Regions** 按钮为外部FLASH配置加密区域

这两种情况下，在应用程序期间，默认加密 image。有关使用加密 image 的时钟限制，请参阅目标处理器文档。结合双启动，为image0设置一个区域，为image1设置一个区域。仅为 image0 设置区域不会对 image1 进行加密。

当 image 被写入目标存储器时，执行 image 加密。加密区域在 SB 文件中配置。解密区域是在CMPA页面中配置的，因此请确保这两个区域对齐。

6.5.3.4 Test life cycle

要在高级生命周期中测试处理器的行为，可以通过将 PMC->LIFECYCLESTATE 设置为所需的级别，临时将生命周期更改为某个更高的级别。在重设硬件之前，此生命周期状态一直有效。

所需步骤如下：

1. 准备 image 并生成密钥。
2. 在 SOCU 寄存器中设置访问控制。
3. 生成 image。
4. 执行写操作。
5. 运行应用。
6. 连接调试探测器。
7. 在 Write 选项卡上，点击**Test life cycle**，并在显示的对话框中设置所需的生命周期状态。
8. 点击 **Apply** 将处理器移至选定的生命周期。现在，可以测试处理器的行为。

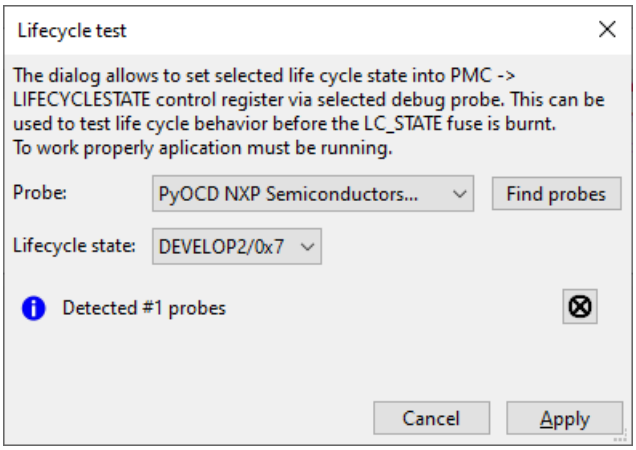

Figure 55. 生命周期测试

6.5.3.5 Life cycle for LPC55(S)3x devices

默认的 development 生命周期是 **Develop**。在部署应用程序之前，请设置“In Field”或“In Field Locked”生命周期（请参阅目标处理器的文档以获得详细说明）。

注意： 生命周期的变化是不可逆的。

当更改生命周期为 **In Field** 时，CMPA 和 CFPA 页安装在 `dev_hsm_provi.sb` 文件中 。在此模式下，假定这些页面被安装到空处理器中，因此没有任何故障（页面更新可能会失败，因此在开发模式下，这些页面在写入脚本中更新，其中进度和错误报告更好）。一旦处理器处于 In Field 状态，SEC 工具就只支持更新应用程序 image；不支持 CMPA 和 CFPA 的更新。

6.6 LPC865 workflow

本节详细介绍 LPC865 设备工作流程。

6.6.1 Preparing images for build for LPC865

该处理器仅支持从内部 flash（XIP）执行的 image。使用默认工具链设置来构建应用程序 image。

6.6.2 Connecting the LPCXpresso860-MAX board for LPC865

本节包含有关配置 LPCXpresso860-MAX 评估板并将其连接到 SEC 的信息：

1. 如要为电路板供电，请使用 USB 电缆将 J4 USB 端口连接到电源适配器或 PC。
2. 将 J2 连接器处的引脚 2（RX）和 4（TX）连接到 UART 转换器并连接到您的 PC。
3. 要将处理器切换到 ISP 模式，请按住 ISP（SW1）按钮并按下 RESET（SW3）。
4. 确认 SEC 工具已使用为所选设备创建的工作区运行。更多详情，请参阅Setting up Secure Provisioning Tool。
5. 打开 **Connection** 对话框并测试连接。请注意第一次进行开发板同步需要较长的时间。

6.6.3 Booting images for LPC865

本章描述了生成和写入明文 bootable images 的过程。

6.6.3.1 Booting unsigned plain image

Plain unsigned 是该处理器支持的唯一启动模式。

首先，生成一个 bootable image：

1. 切换到 **Build image** 视图。
2. 选择在 Preparing source image for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices中生成的 image 作为 **Source executable image**。
3. 如果存在一个 binary image，请将起始地址设置为 0x0。
4. 单击 **Build image**以生成一个 bootable image。在此操作期间，将对 image 应用以下更改：生命周期（=代码读取保护）和 CRC。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作执行成功，请复位开发板。处理器将在几秒钟后启动。

6.6.4 Life cycles

处理器不支持真正的生命周期，因此可以配置代码读保护而不是生命周期。

6.7 MC56F818xx/7xx/6xx and MWCT2xD2/12 devices workflow

本章介绍 MC56F818xx/7xx/6xx 和 MWCT2xD2/12 处理器的工作流程。

6.7.1 Preparing source image for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices

在此步骤中，必须选择将要执行 image 的目标存储器。MC56F818xx/7xx/6xx 和 MWCT2xD2/12 设备提供以下选项：

- 在内部 FLASH中执行 image — XIP (eXecution In Place) image，这意味着该 image 直接从其所在的内存中执行。无需修改默认配置，可以照原样构建 MCUXpresso SDK 示例。

6.7.2 Connecting the board for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices

本节包含有关配置评估板并将其连接到 SEC 的信息：

- MC56F81868-EVK
- MC56F81000-EVK
- WCT-QI2

1. 从 USB_TYPE_C 端口 (WCT-QI2) 为开发板供电。
2. 通过 MCU-Link 将 J4 1-TX、3-RX、5-GND 端口 (WCT-QI2) 连接到您的 PC。
3. 在**Connection**对话框中，将连接设置为 **UART**。
4. 要进入 ISP 启动模式，请在板启动后 5 秒内向处理器发送 blhost 命令，例如单击**Connection** 对话框中的**Test connection** 按钮。

注意： 当启动后 5 秒内没有发送 blhost 命令时，处理器将从内部 flash 启动。

6.7.3 Booting images for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices

本章描述了明文和签名的 bootable images 的生成和写入过程。

6.7.3.1 Booting plain unsigned or CRC image

明文（plain）image 通常用于开发。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择了 **Plain unsigned** 或 **Plain with CRC** 启动类型。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices中生成的 image 作为 **Source executable image**。
4. 如果存在一个 binary image，请将起始地址设置为 0x0。
5. 如果需要，请打开 **BCA/FCF** 配置。
6. 单击 **Build image**以生成一个 bootable image。结果是一个二进制 bootable image。对于安全处理器，还有单独的引导标头二进制文件（BCA、FCF）和带有应用程序的二进制文件。这是由于内部 flash 中的不间断空间，DUKB 区域在写入过程中必须保持不变。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作执行成功，请复位开发板。处理器将在 5 秒后启动。

6.7.3.2 Booting plain signed image

本章节介绍如何构建和编写明文签名图像。

生成 bootable image：

1. 在工具栏中，选择 **Plain signed**启动类型 。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices中生成的 image 作为 **Source executable image**。
4. **Authentication key**，选择 ROT1。
5. 确保主板已连接且处理器处于 ISP 模式。在构建过程中，会准备一个 SBx 文件。该处理器用作 HSM 来创建加密的 SBx 文件。
6. 生命周期保持为 **OEM Open**。关于**OEM Closed**生命周期的详情，请参阅Life cycle and device HSM trust provisioning for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices。
7. 单击 **Build image**以生成一个 bootable image。结果是一个二进制bootable image，二进制启动标头（BCA、FCF），以及用于将该 image 安装到处理器的 SB3 封装。

成功构建 bootable image 后，可以上载到处理器：

1. 确保处理器处于 ISP 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

6.7.3.3 Life cycle and device HSM trust provisioning for MC56F818xx/7xx/6xx and MWCT2xD2/12 devices

默认的生命周期应该用于开发，它是 **OEM Open**。部署应用程序之前，请设置 **OEM Closed** 生命周期（请参阅目标处理器的文档以获得详细说明）。

当在工具栏中设置 **OEM Closed** 生命周期和 **Device HSM** 时，将在构建期间创建信任配置 SBx 文件。`IFR ISK_CERT_HASH` 字段由 `dev_hsm_provi.sbx` 文件写入 IFR。假设在此模式下，IFR 字段在处理器中具有默认值。

警告：

- IFR 字段的写入是不可逆的。
- WPC配置必须在推进生命周期之前完成。一旦电路板进入高级生命周期，配置命令就不可用了。
- 如果需要，可以将生命周期重置为 OEM Open 状态。有两种可能：
  - 保留Plain signed启动类型，设置 OEM Open 生命周期，禁用信任配置并进行构建和写入。这样，应用程序就会更新，生命周期就会设置为 OEM Open。IFR 中的字段保持不变。
  - 设置Plain unsigned启动类型，OEM Open 生命周期，禁用信任配置并进行构建和写入。对于某些设备，可能需要在执行写入操作之前禁用 flash 安全（以启用 FlashEraseAllUnsecure 命令）。这样，整个 flash 就会被擦除，生命周期将设置为 OEM Open。IFR 中的字段保持不变。
- blhost flash-erase-all 命令会擦除整个内存，这也会影响生命周期状态。flash-erase-all 命令擦除全部内存和设备重置后，生命周期处于关闭状态。

生命周期和设备 **HSM** 信任配置

	ISK_CERT_HASH	**SBx** 文件
OEM Open 生命周期，已签名的启动类型	包含在 FCB 中	SBx 用于写入/更新应用程序 image
OEM Closed 生命周期，已签名的启动类型	不可逆地写入信任配置 SBx 文件中的 IFR	SBx 用于写入/更新应用程序 image

6.7.4 EdgeLock 2GO WPC workflow

EdgeLock 2GO 平台提供 WPC Qi 认证所需的无线充电联盟 (WPC) 证书链的配置。SEC 工具支持 WPC 配置场景，其中 NXP 作为 WPC 制造 CA 服务提供商。对于该工作流程，假设使用 NXP 示例项目。请联系 NXP 销售以获取示例。

6.7.4.1 EdgeLock 2GO WPC flow step by step

要配置 EdgeLock 2GO 配置，请执行以下步骤：

1. 打开安全启动模式并验证应用程序是否正常工作。
2. 在 EdgeLock 2GO server上，创建 Qi ### *CAs* 和 ######。
3. 创建 API 密钥。更多详情，请参阅 API key to access the EdgeLock 2GO server。
4. 在 **main menu > Target > Trust Provisioning Mode**中，选中**enable WPC**并填写EdgeLock2GO 参数；使用 Qi ID 作为 Qi PUC 公钥。
5. 将未签名的 ISK 块二进制文件 `<workspace>/keys/ROT1_cert_block_wpc_signing.bin` 上传到 EdgeLock 2GO server 进行签名。签名后，下载已签名的块并将其保存为 `<workspace>/keys/isk_nxp_signed.bin`。
6. 在 **write** 选项卡上，单击**Extract WPC certificate**按钮以获取 C 数组形式的制造商 CA 证书 `<workspace>/el2go/mfg_ca_cert_array.c`。板子必须在 ISP 模式下连接。

7. 使用此数组替换 `systemAuthentication.c`中的 WpcMfgCaCert[]；证书放置在内存地址 0x600-0x7FF。预期应用程序在地址 0x5E0-0x5FF 包含 WPC 根 CA 证书哈希。
8. 在 Codewarrior 中重新构建项目。
9. 使用更新的源文件，在 SEC 工具中执行构建操作。
10. 将 image 写入目标。作为写入操作的一部分，板子配置了 WPC PU 证书。

6.8 MCX A1/A2/A3/L2 device workflow

本章介绍 A13x/A14x/A15x/A16x/A17x/A25x/A26x/A35x/36x/L25x 处理器的工作流程。

6.8.1 Preparing source image for MCX A1/A2/A3/L2 devices

- 从内部 FLASH 运行 image 是几乎所有 SDK 示例的默认选项。无需修改缺省配置，可按原样构建示例。
- 从内部 RAM 运行 image；创建此示例时，选择将应用程序链接到 RAM。

6.8.2 Connecting the board

本节包含有关配置评估板并将其连接到 SEC 工具的信息：

MCXA/MCXL 板的启动模式

板	**ISP** 启动模式	从内部 **FLASH** 启动
FRDM-MCXA153	SW2/JP8	默认情况下
FRDM-MCXA156	SW3	默认情况下
FRDM-MCXA266	SW3	默认情况下
FRDM-MCXA346	SW3	默认情况下
FRDM-MCXA366	SW3	默认情况下
FRDM-MCXL255	SW3	默认情况下

请按照以下步骤来连接开发板：

1. 选择 ISP 启动模式，请参阅上表。
2. 使用 USB 电缆将 UART/USB 端口连接到您的 PC。
3. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅 Setting up Secure Provisioning Tool。
4. 进入 **main menu > Target > Connection**，选择 UART/USB 并测试连接。

6.8.3 Booting images MCX A1/A2/A3/L2 devices

关于如何启动明文（plain）未签名 image 和带 CRC 的 image 的方法，与 MCXN 设备相同。更多详情，请参阅Booting plain or CRC image for MCX Nx4x/N23x devices。

6.8.3.1 Setting CMPA flash access control

CMPA 配置包括标记为 `FLASH_ACL_#_#`。这些字段用于初始化内存块检查器 (MBC) 配置寄存器。CMPA 中定义的值在启动期间加载，如果未被锁定，则可以在以后进行修改。

处理器出厂时闪存为空，这意味着 CMPA（客户制造编程区）页面最初无效。在这种默认状态下，闪存完全可访问——允许读取、写入和执行操作。

一旦在芯片上配置了 CMPA，就必须根据预期的用例明确定义访问权限。访问权限设置不当可能会导致应用程序运行异常。

默认情况下，SEC 工具将访问权限设置为 “read + execute”。但是，在配置应用程序 image 时，SEC 工具不接受此默认设置。相反，它要求用户明确定义访问权限，以确保配置是有意为之且经过验证的。

需要注意的是，运行时可能访问的任何其他内存区域都必须配置正确的访问权限。否则可能导致访问违规或运行时错误。

6.8.3.2 Provisioning with device HSM for MCXA25x/MCXA26x/MCXA36x

本节介绍如何构建和写入由设备 HSM sb 文件提供的纯文本 image 和带 CRC 的纯文本 image。

生成bootable image：

1. 在工具栏中选择所需的启动类型。
2. 切换到 **Build image** 视图。
3. 选择 image 作为 **Source executable image**。
4. “OEM seed”对称密钥使用随机值。
5. 如果需要，打开 **Dual image boot** 并进行配置。
6. 在开始构建过程之前，请确保电路板已正确连接，并且处理器处于 ISP 模式。 构建过程中，会准备一个配置 SB3 文件。配置文件包含 CMPA、可执行 image 和其他 image。 如果没有连接开发板，则在准备配置 SB3 文件时构建将失败。
7. 单击 **Build Image**以生成一个 bootable image。 此过程会创建一个设备 HSM SB3 胶囊，用于将 image 安装到处理器中。

注意： 在任何高级 ROP 状态下，CMPA 都配置为删除安全安装程序 (SI)，因为高级生命周期不支持 SI。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于 ISP 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

6.8.4 Life cycle for MCX A1/A2/A3/L2 devices

默认的生命周期应该用于开发，它是 **OEM Open**。在部署应用程序之前，请设置“In Field ROP 1”，“In Field ROP 2”，或“In Field ROP 3”生命周期（更多详情，请参阅目标处理器的文档）。对于内部闪存中含有 CMPA 的设备，可以通过执行批量擦除命令来恢复其生命周期。

注意： 在现场 ROP 3 中，生命周期无法逆转，ISP 命令不可用。

6.8.4.1 Reverting life cycle

要恢复生命周期，请执行批量擦除命令。

对于带有安全安装程序 (SI) 的设备，必须先删除 SI 固件，否则批量擦除命令将失败。这是因为包含 SI 的内存区域受到保护，无法访问。一旦 SI 被擦除，并且 CMPA 中的 ERASE_TOKEN[x] 寄存器被设置，闪存保护就会被禁用。 注意： 批量擦除操作还会擦除 CMPA 寄存器和擦除令牌寄存器。因此，在执行批量擦除之前，请保存擦除令牌值，以便之后可以恢复该值，从而重新访问 SI 所在的闪存区域。

6.8.5 MCUboot for MCX A1/A2/A3/L2 devices

MCUboot 在 MCUboot 工作流程中有描述。以下是针对MCX的评论：

- 在 CMPA 中，MCUboot 管理应用程序时，flash 访问字段设置必须设置为任何“解锁”规则。必须设置解锁规则，因为 MCUboot 会在应用程序更新期间将区域规则设置为“写入”规则，并在运行应用程序时将其设置为“执行”规则，从而修改区域规则。

6.9 MCX C041/C242/C444 device workflow

本章介绍 Cx4x 处理器的工作流程。

6.9.1 Preparing source image for MCX C041/C242/C444 devices

- 从内部 FLASH 运行 image 是几乎所有 SDK 示例的默认选项。无需修改缺省配置，可按原样构建示例。
- 创建此示例时，image 从内部 RAM 运行。选择链接应用程序到 RAM。MCXC041 设备不支持该选项。

6.9.2 Connecting the board for MCX C041/C242/C444 devices

本节包含有关配置 FRDM-MCXC041，FRDM-MCXC242，和 FRDM-MCXC444 并将其连接到 SEC 的信息：

MCXC 板的启动模式

板	**ISP** 启动模式	从内部 **FLASH** 启动
FRDM-MCXC041	SW3	默认情况下
FRDM-MCXC242	SW3	默认情况下
FRDM-MCXC444	SW3	默认情况下

1. 选择 ISP 启动模式更多详情，请参阅上表。
2. 使用 USB 电缆将 UART/USB 端口连接到您的 PC。
3. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅 Setting up Secure Provisioning Tool。
4. 进入 **main menu > Target > Connection**，选择 UART 并测试连接。

6.9.3 Booting images for MCX C041/C242/C444 devices

本章描述了构建 bootable images 并将其写入内部 flash 以及启动。

6.9.3.1 Booting plain unsigned image

首先，生成一个 bootable image：

1. 切换到 **Build image** 视图。
2. 选择在 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 **Source executable image**。
3. 如果存在一个二进制 image，请将起始地址设置为 0x00000000。
4. 如果需要，请打开 **BCA/FCF** 配置。
5. 单击 **Build image**以生成一个 bootable image。结果是一个二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于 ISP 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，请重置开发板以启动 image。

6.9.3.2 Life cycle for MCX C041/C242/C444 devices

默认的 development 生命周期是 **Flash unsecured**。部署应用程序之前，请设置 **Flash secured** 生命周期（更多详情，请参阅目标处理器的相关文档）。在切换生命周期的时候，最佳做法是设置 BCA 和 FCF 页面。如果 ISP 可用并且 FCF 字段“批量擦除”已启用，则可以擦除启用了 flash 安全性的芯片。

6.9.3.3 Backdoor key

如果设置了后门密钥并且在 FCF 中启用了后门密钥比较，则可以暂时禁用高级生命周期。后门密钥验证后，处理器的行为就像处于 OEM Open 生命周期中一样，直到下次重置。如果验证失败，则无法进一步验证，直到 flash 重置。可以使用“写入”选项卡下的“禁用闪存安全”对话框禁用闪存安全。

6.10 MCX E24 device workflow

本章介绍 MCX E24 处理器的工作流程。

6.10.1 Preparing source image for MCX E24x devices

从内部 flash 运行 image 是几乎所有 SDK 示例的默认选项。无需修改缺省配置，可按原样构建示例。

MCX E24x 设备仅支持内置 flash 作为启动存储器。

6.10.2 Connecting the board for MCX E24x devices

本节包含有关配置 FRDM-MCXE247 评估板并将其连接到 SEC 的信息：

6.10.2.1 Flashloader boot behavior on MCX E24x devices

MCX E24x 设备不支持 ISP 启动模式。相反，它们在出厂时就预先编程了一个存储在内部 flash 中的 flashloader image，其中包含flashloader 加载程序和 flashloader 固件。

默认情况下，这些设备从内部 flash 启动。在启动过程中，预编程的 flashloader 会自动加载到 RAM 中并从那里执行。这使得无需单独的 ISP 模式即可实现 flash 编程功能。

6.10.2.2 Flashloader initialization via debug probe

如果 flashloader 未在设备上运行，可以使用调试探针将其加载并从 RAM 中执行。FRDM-MCXE247 上的板载 MCU-Link 支持调试探针功能，因此可用于此目的。

注意：Flashloader 可以通过 MCU-Link 或 J-Link 调试探针初始化。

注意：使用调试探针主机软件（LinkServer或JLink）的 SEC 工具脚本要求主机软件的安装目录包含在 `PATH` 环境变量中。

在 **main menu > Target > Debug Probe** 或在 **Toolbar**、**Dbg**中选择调试探针。

6.10.2.3 FRDM-MCXE247 UART pin connections

要在预编程闪存加载器（使用LPUART1）和板载MCU-Link（使用LPUART2）之间建立通信，请使用两根杜邦线将MikroBUS UART连接到Arduino UART，连接方式如下：

- 将Arduino J1 - Pin 2 (PTD17/LPUART2_RX-Arduino_D0)连接到MikroBUS J5 - Pin 3 (PTC8/LPUART1_RX-MIKROE)

- 将Arduino J1 - Pin 4 (PTD12/LPUART2_TX-Arduino_D1)连接到MikroBUS J5 - Pin 4 (PTC9/LPUART1_TX-MIKROE)

通过此连接，可以使用板载MCU-Link的USB转UART桥接功能在主机PC和闪存加载器之间通过UART进行通信。

6.10.2.4 FRDM-MCXE247 connection via SPI

要在预编程闪存加载器和外部USB转SPI桥接器（例如MCU-Link Pro调试探针）之间建立SPI通信，请将桥接器连接到目标板，连接方式如下：

- SPI MOSI连接到MicroBUS J6 - Pin 5 (PTB3/LPSPI0_SIN-MIKROE)
- SPI SCK 连接到 Arduino J3 - Pin 1 (PTB2/FTM1_QD_PHB-MC_ENC_B)
- SPI MISO 连接到 IO Expander J8 - Pin 12 (PTB1)
- SPI PCS0 连接到 IO Expander J8 - Pin 9 (PTB0)

通过此连接，外部USB转SPI桥接器可用于在主机PC和闪存加载器之间通过SPI进行通信。

6.10.2.5 Connecting the board to host PC

1. 使用USB电缆将J13 (MCU-Link USB)端口连接到您的PC。
2. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅 Setting up Secure Provisioning Tool。
3. 进入 **main menu > Target > Connection**，选择 UART 并测试连接。

注意： 当单击 **Test Connection** 按钮时，如果闪存加载器尚未运行，它将通过调试探针初始化。在这种情况下，必须选择调试探针，请参见Flashloader initialization via debug probe。

6.10.3 Booting images for MCX E24x devices

本章描述了构建 bootable images 并将其写入内部 flash 以及启动。

注意： 目前SEC工具仅支持MCX E24x的 **Unsigned** 启动。

6.10.3.1 Booting plain unsigned image - unsecure boot

本节介绍不安全启动的 image 构建和写入。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择**Unsigned**启动类型。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for MCX E24x devices 中生成的 image 作为**Source executable image**。
4. 如果存在一个二进制 image，请将起始地址设置为 0x00000000。
5. 如果需要，请打开 **FCF** 配置。
6. 如果需要，请打开 **SHE keys configuration** 并配置，参阅SHE 密钥配置。
7. 单击 **Build image**以生成一个 bootable image。结果是一个二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器已连接。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，请重置开发板以启动 image。

6.10.3.2 Booting plain signed image - secure boot

本节介绍如何构建和编写用于安全启动的可启动 image。在安全启动期间，对可启动 image 代码段进行身份验证，并将生成的 MAC 与先前存储在安全密钥存储中的值进行比较 (BOOT_MAC)。

首先，生成一个 bootable image：

1. 请确保在工具栏中选择已验证 **(SHE)** 串行 或 已验证 **(SHE)** 并行 或 已验证 **(SHE)** 严格**t** 启动类型。有关安全启动类型的更多信息，请参阅启动类型工具提示。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for MCX E24x devices 中生成的 image 作为**Source executable image**。
4. 如果存在一个二进制 image，请将起始地址设置为 0x00000000。
5. 如果需要，请打开 **FCF** 配置并配置。
6. 打开 **SHE keys configuration** 并配置，参阅SHE keys configuration。
   - 确保BOOT_MAC_KEY已配置为启用安全启动。
   - 有关BOOT_MAC的详情，请参阅Manual and automatic BOOT_MAC calculation。
7. 单击 **Build image**以生成一个 bootable image。
   - 这将生成一个二进制 bootable image。
   - 如果启用，则会计算 BOOT_MAC。
   - 同时还创建了用于加载 SHE 密钥的二进制文件。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器已连接。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，请重置开发板以启动 image。

注意： 已验证 **(SHE)** 严格启动模式一旦设置即为永久性，无法更改。此外，此模式下不支持自动BOOT_MAC计算。在设置此启动模式之前，必须先进行BOOT_MAC计算并存储。否则，设备将保持重置状态，无法启动。

6.10.3.3 SHE keys configuration

在 **SHE** 密钥配置对话框中，您可以通过设置密钥存储大小来配置处理器以进行 CSEC/SHE 安全操作。这决定了闪存的分区方式。

将密钥数量配置为 **No setup** 会导致跳过 CSEC/SHE 密钥存储设置，因此不会执行闪存分区。此选项仅供开发之用。

最多可配置 20 个按键。前三个按键都有其特定用途。

- MASTER_ECU_KEY 用于将 CSEC/SHE 重置为出厂状态或修改任何其他密钥。
- BOOT_MAC_KEY 用于在安全启动过程中验证软件的真实性。
- BOOT_MAC 用于存储安全启动过程中使用的应用程序 image 的 MAC 值。
- USER_KEY_1 至 USER_KEY_17 用户密钥可供应用程序特定使用。

注意：如果任何按键设置了写保护标志，则设备无法重置为出厂状态。

6.10.3.3.1 Manual and automatic BOOT_MAC calculation

可以通过两种方式进行BOOT_MAC计算和编程：

- 手动：使用`nxpshe calc-boot-mac`命令离线计算BOOT_MAC，使用`blhost key-provisioning set_user_key`命令编程。要使用此选项，请在 **Build image**视图的 **SHE keys configuration**对话框中启用 BOOT_MAC。有关BOOT_MAC更新的信息，请参阅 SHE 密钥更新。
- 自动使用 CSEC：仅当尚未在 CSEC/SHE 密钥库中编程 BOOT_MAC 时才应使用此选项。要启用自动计算，请在 **Build image**视图的 **SHE keys configuration**对话框中禁用 BOOT_MAC。

6.10.3.3.2 SHE key update

已在 CSEC/SHE 密钥存储中设置的密钥可以进行更新，前提是该密钥未启用 **Write Protection** 标志。要成功执行更新，您必须在 **SHE** 密钥配置 对话框中增加密钥的 计数器值。

在安全模式下，当手动离线计算 BOOT_MAC 时，源可执行**image** 发生更改时，都需要进行BOOT_MAC更新。

6.10.3.4 Life cycle for MCX E24x devices

默认的 development 生命周期是 **Flash unsecured**。部署应用程序之前，请设置 **Flash secured** 生命周期（更多详情，请参阅目标处理器的相关文档）。在切换生命周期的时候，最佳做法是设置 FCF 页面。

6.10.3.5 Revert SHE keys configuration and/or flash security

下表提供了有关特定处理器配置是否能够将 SHE 密钥存储重置为出厂状态（即删除分区）以及通过批量擦除闪存来禁用闪存安全性的信息。

生命周期	**SHE** 密钥配置	写保护标志	是否可以重置**SHE**密钥库？	是否可以禁用闪存安全？
闪存不安全	密钥库大小设置为 0	N/A	有	有
闪存不安全	当前密钥库	无	是	有
闪存不安全	当前密钥库	任何受保护的密钥	否	否
Flash已加密	密钥库大小设置为 0	N/A	否	有
闪存安全	当前密钥库	无	否	否
Flash安全	当前密钥库	任何受保护的密钥	否	否

6.10.3.5.1 Reset SHE key store

如果密钥未受写保护且生命周期为 Flash Unsecure，则可以将 CSEC/SHE 密钥库重置为出厂状态。如果生命周期设置为，则只有 **Disable flash security…** 可用，这将触发对闪存的大规模擦除。

重置密钥库：

1. 在**Write image**视图中，通过**Reset SHE key store…** 按钮打开**Reset SHE key store**对话框。
2. 根据处理器上当前的密钥存储设置选择重置方法。
   - 密钥存储存在 - 密钥存储大小配置为 5、10 或 20。在这种情况下，需要使用`nxpshe reset`命令。只有 SHE 密钥存储被重置，应用程序Image仍保留在闪存中。
   - 无密钥存储 - 密钥存储大小配置为 0。在这种情况下，将使用`blhost flash-erase-all-unsecure`命令擦除整个闪存，并恢复闪存安全部分。
3. 单击 **Reset** 按钮。

6.10.3.5.2 Disable flash security via debug probe

如果满足以下条件，则可以使用调试探针解除已保护设备的安全性：

- FCF 配置中的“mass erase”字段未被禁用。
- SHE 密钥库未配置（密钥数量设置为 **No setup**或 **0**）。 当设备未加密时，会自动触发对内部flash的大规模擦除。

注意： 批量擦除操作不会影响密钥存储设置。

要解除设备安全，请在 **Write image**视图中单击 **Disable flash security...**。大容量擦除后，flashloader image 将写入内部flash并启动。

6.11 MCX Nx4x/N23x device workflow

本章介绍 Nx4x/N23x 处理器的工作流程。

6.11.1 Preparing source image for MCX Nx4x/N23x devices

- 从内部 FLASH 运行 image 是几乎所有 SDK 示例的默认选项。无需修改缺省配置，可按原样构建示例。
- 从外部 FLASH 运行的 image 必须从地址 0x80001000 开始，在预先设置中在 MCUXpressoIDE 中创建示例时编辑此地址。
- 创建此示例时从内部 RAM 运行的 image 选择将应用程序链接到 RAM。

6.11.2 Connecting the board for MCX Nx4x/N23x devices

本节包含有关配置 FRDM-MCXN947、MCX-N9XX-EVK、MCX-N5XX-EVK 并将其连接到 SEC 的信息。

MCXCN 板的启动模式

板	**ISP** 启动模式	从外部 **FLASH** 启动	从内部 **FLASH** 启动
MCX-N9XX-EVK	SW3/JP49	CMPA 中定义的启动源	
FRDM-MCXN947	SW3	CMPA 中定义的启动源	
MCX-N5XX-EVK	SW3/JP49	CMPA 中定义的启动源	
FRDM-MCXN236	SW3	无	默认情况下

1. 选择 ISP 启动模式，请参阅上表。
2. 使用 USB 电缆将 UART/USB 端口连接到您的 PC。
3. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅Setting up Secure Provisioning Tool。
4. 进入 **main menu > Target > Connection** 并选择 UART 并测试连接。

6.11.3 Booting images for MCX Nx4x/N23x devices

本章描述了构建 bootable images 并将其写入内部 flash 以及启动。与从外部 flash 启动类似，但使用链接到外部 flash 的 image。

6.11.3.1 Booting plain or CRC image for MCX Nx4x/N23x devices

明文（plain）image 通常用于开发。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择了 **Plain unsigned**或**Plain with CRC** 启动类型。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 **Source executable image**。
4. 如果存在一个 binary image，请将起始地址设置为 0x00000000（外部 flash 0x80001000）。
5. 如果需要，打开 **Dual image boot** 并进行配置。
6. 单击 **Build image** 按钮以生成一个 bootable image。结果是二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，请移除 ISP 跳线并重置板以启动 image。

6.11.3.2 Booting plain signed image

本章节介绍如何构建和编写明文签名图像。

生成 bootable image：

1. 在工具栏中，选择 **Plain signed**启动类型 。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 **Source executable image**。
4. 在 PKI 选项页面生成密钥。
5. 返回**Build image** 视图，从下拉菜单中选择任何验证密钥，例如 ROT1：IMG1_1。
6. 对“CUST_MK_SK”和“OEM seed”对称密钥，使用随机值。
7. 如果需要，打开 **Dual image boot** 并进行配置。
8. 确保主板已连接且处理器处于 ISP 模式。在构建过程中，准备配置 SB3 文件以安装 CUST_MK_SK 到处理器中。如果没有连接开发板，则在准备配置 SB3 文件时构建将失败。但其他构建过程已完成。
   注意： SB 文件构建完成后，处理器会重置。
9. 单击 **Build image**按钮以生成一个 bootable image。结果生成一个二进制bootable image以及用于将该image安装到处理器的SB3封装。

成功构建 bootable image 后，可以上载到处理器：

1. 确保处理器处于 ISP 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

6.11.3.3 Booting encrypted image

支持带有CRC或签名 image 的加密 image。创建加密 image 的过程与签名 image 类似。此外，在 **Build image** 视图中配置加密区域：

- 使用 **PRINCE Regions** 按钮为内部FLASH配置加密区域
- 使用 **IPED Regions** 按钮为外部FLASH配置加密区域

这两种情况下，在应用程序期间，默认加密 image。有关使用加密 image 的时钟限制，请参阅目标处理器文档。结合双启动，为 image0 设置一个区域，为 image1 设置一个区域。仅为 image0 设置区域不会对 image1 进行加密。

当 image 被写入目标存储器时，执行 image 加密。加密区域在 SB 文件中配置。解密区域是在CMPA页面中配置的，因此请确保这两个区域对齐。

6.11.3.4 Life cycle for MCX Nx4x/N23x devices

默认的 development 生命周期是 **Develop**。在部署应用程序之前，请设置“In Field”或“In Field Locked”生命周期（更多详情，请参阅目标处理器的文档）。切换生命周期时，建议将 ROTKH 和 PRINCE 或 IPED 区域设置烧入 fuse。ROTKH fuse 由工具自动设置，但 PRINCE 或 IPED 设置由用户设置。

注意： 生命周期的变化是不可逆的。

生命周期配置还会影响安全资产的安装方式。在开发生命周期中，一些资产安装在编写脚本中，因此更容易调试更改和问题。

下表包含有关如何根据所选生命 周期安装安全资产的详细信息：

生命周期和信任配置

信任配置生命周期	设备 **HSM** 开发	现场设备 **HSM**	**EdgeLock 2GO** 开发	现场 **EdgeLock 2GO**
CMPA	设备 HSM SB 文件	设备 HSM SB 文件	el2go_provi.sb3	el2go_provi.sb3
CUST MK SK	设备 HSM SB 文件	设备 HSM SB 文件	CUST-MK_SK	CUST-MK_SK
ROKTH in CMPA	设备 HSM SB 文件	设备 HSM SB 文件	RKTH	RKTH
CFPA	写入脚本	设备 HSM SB 文件	写入脚本	el2go_provi.sb3
RKTH fuses	除非客户要求，否则不会烧录	设备 HSM SB 文件	除非客户要求，否则不会烧录	el2go_provi.sb3
NPX 和 **IPED fuse**	除非客户要求，否则不会烧录	设备 HSM SB 文件	除非客户要求，否则不会烧录	el2go_provi.sb3
其余定制 **fuses**	写入脚本	设备 HSM SB 文件	默认编写脚本；可选 el2go_provi.sb3	el2go_provi.sb3

在 **Develop** 生命周期中，CFPA 被写入写入脚本中，因此可以轻松检测到任何故障。在 **In Field** 生命周期中，应该为空处理器完成配置。

一旦处理器处于 In Field 状态，写入脚本仍然允许更新应用程序映像；写入脚本不支持其他更新，但可以使用自定义 SB 文件完成。

6.11.3.5 Test life cycle

MCX Nx4x/N23x 变体可以通过设置 HEADER 寄存器中的位字段 CFPA_LC_STATE 和 INV_CFPA_LC_STATE 来设置 CFPA 页上的测试生命周期。通过写入此 CFPA 设置，MCU 的行为就像 LF 已在 OTP 中移动一样。

注意： 仅使用预期的 LC 值，其他值可能会导致芯片变砖。

要从高级生命周期返回，请将位字段 CFPA_LC_STATE 设置回 0x0 并将其写入芯片。如果 LC 仅在 CFPA 中移动，则可以在上电复位后通过执行 blhost -u 0x1fc9, 0x014f - write-memory 0x01000000 cfpa.bin 重写 CFPA 页。

推进 LC 和返回的步骤：

1. 打开或准备具有安全启动类型（签名或加密）的工作区
2. 在 **OTP/PFR configuration** 中，设置 CFPA_LC_STATE 为 0xF，INV_CFPA_LC_STATE 为 0xF0。
3. 构建并写入 image
4. 将设备重新通电进入 ISP

5. 在 OTP/PFR 配置中，阅读 CFPA 页面
6. 单击 **Reset** 按钮或使用 blhost reset 命令重置设备
7. 测试安全生命周期行为（有限命令、基于 SOCU 寄存器的调试权限）
8. 测试完成后，将设备重新启动以进入 ISP
9. 打开 **OTP/PFR configuration**，设置 CFPA_LC_STATE 为 0x0，INV_CFPA_LC_STATE 为 0xFF。
10. 启用高级模式并将 CFPA 页写入设备
11. 重置后，设备将恢复正常 LC
    注意： 对于用于电源循环的 EVK 变体，请按照以下步骤操作：
    a. 关闭开发板电源。
    b. 为 JP22 添加跳线。
    c. 将电源更改为 USB (J28)。
    d. 按住 ISP 按钮（或短接 ISP 跳线）并通过 J28 连接到开发板。
    e. 拆下跳线 JP22

6.11.3.6 EdgeLock 2GO

有关EdgeLock 2GO，请查阅EdgeLock 2GO Trust Provisioning workflow。以下是 MCXN 具体注释：

- EdgeLock 2GO 安装的资产对于开发和现场生命周期是不同的，请参阅Life cycle for MCX Nx4x/N23x devices 。
- SEC 工具中生成了 `ed2go_provi.sb3` 文件，其中包含附加 fuse 的烧录。对于 EdgeLock 2GO 固件，此文件是可选的；但是，SEC 工具使用该文件在现场生命周期中安装fuse。
  要通过 Web 门户将文件添加到 EdgeLock 2GO 服务器，请使用以下步骤：
  – 创建安全的二进制对象
  – 选择二进制文件
  – 指定任意名称
  – 将对象标识 (OID) 设置为 0x7FFF817C
  – 选择非机密文件
  – 在策略中，选择“NONE”
- 对于开发，请使用 SEC 工具中的开发生命周期以及 EdgeLock 2GO 服务器上安全对象配置中的 **Open** 策略。通过这些设置，生命周期不会改变，并且仍然会擦除处理器。对于生产，请使用现场生命周期和 **Closed** 策略。这不适用于上面的 SB3 文件，其中策略应始终为 NONE。
- 在启动配置固件之前必须擦除 CMPA 页，以便可以正确配置它。el2go-host 应用程序使用参数 `-clear` 来清除 CMPA。如果使用新的/空的处理器执行生产，则可以删除该参数。
- 安全对象的地址必须位于内部 flash 中，配置固件会在写入安全对象之前擦除该区域

6.12 RT10xx/RT116x/RT117x device workflow

本节详细介绍 RT10xx/RT116x/RT117x 设备工作流程。

6.12.1 Preparing source image for RT10xx/RT116x/RT117x devices

在此步骤中，必须选择将要执行 image 的目标存储器。以下选项可用于 RT10xx/RT116x/RT117x 设备：

- 从外部 **NOR flash** 运行的 **image**
  该 image 被称为 **XIP**（e**X**ecution **I****n P**lace）image，直接从其所在的内存执行。
- 在内部 **RAM** 中运行的 **image**
  该 image 可以被放置到 SD 卡/eMMC 或者外部 flash（SPI NOR、SPI NAND、或 SEMC NAND）中，运行时先复制到 RAM 中，并在启动期间从那里执行。

- 在 **SDRAM** 中运行的 **image**
  该 image 可以被放置到 SD 卡或者外部 flash（SPI NOR、SPI NAND、或 SEMC NAND）中，运行时先复制到 SDRAM 中，并在启动期间从那里执行。

6.12.1.1 Image running from external NOR flash

- **MCUXpresso IDE**
  led_blinky工程默认目标为外部 flash。
  1. 可选择打开 **Project > Properties > C/C++ Build > Settings > MCU C Compiler > Preprocessor > Defined symbols**，设置 **XIP_BOOT_HEADER_ENABLE**为 **0**。此步骤现在是可选的，因为bootable image 可以用作 build 选项卡上的输入。
  2. 生成 image 文件。最终生成 image 为 `Debug\evkmimxrt10##\_iled_blinky.axf`。该 image 可以在 SEC 工具中的 **Source executable image** 中进行使用。
- **KeilMDK 5**
  1. 在工具栏中，选择 **iled_blinky flexspi_nor_debug**。
  2. 或者，打开 **Project > Options > "*C/C++*"**，禁用 define symbol **XIP_BOOT_HEADER_ENABLE=0**（设置为 0）。此步骤现在是可选的，因为bootable image 可以用作 build 选项卡上的输入。
  3. 打开 **Project > Options > Linker**，删除所有 *-keep* 选项，然后定义 **XIP_BOOT_HEADER_ENABLE**。最后，**Misc. controls** 中只会包含 *-remove*的选项。
  4. 生成 image 文件。输出 image 为 `boards\evkmimxrt10\#\#\demo\_apps\led_blinky\mdk\flexspi_nor_debug\iled_blinky.hex`。
- **IAR Embedded Workbench**
  1. 在 **Project > Edit Configurations …** 中，选择 **flexspi_nor_debug**。
  2. 或者，在 Project **Options > C/C++ Compiler >Preprocessor>Defined Symbols**，将现有 **XIP_BOOT_HEADER_ENABLEdefine** 添加或更改为 **0**。此步骤现在是可选的，因为bootable image 可用作构建选项卡上的输入。
  3. 对于多核处理器来说，在 **Project > Options… > General Options > Target**中选择使用核。例如，在 RT1176 平台的 *iled_blinky_cm7* 工程中，选择处理器使用核为 *Cortex-M7*。
  4. 生成 image 文件。输出 image 为 `boards\evkmimxrt10##\demo_apps\led_blinky\iar\flexspi_nor_debug\iled\_blinky.out`.

6.12.1.2 Image running in internal RAM

注意： 本节中的内存地址和大小仅用作示例，具体取决于所选芯片。

- **MCUXpresso IDE**
  1. 在 **Project > Properties - C/C++ Build > Settings > Tool Settings > MCU Linker > Managed Linker Script**中检查是否勾选 **Link application to RAM**。
  2. 在 **Project > Properties > C/C++ Build > MCU settings**中，删除 **Flash**，修改 SRAM_ITC 为从0x3000开始，大小为0x1D000。

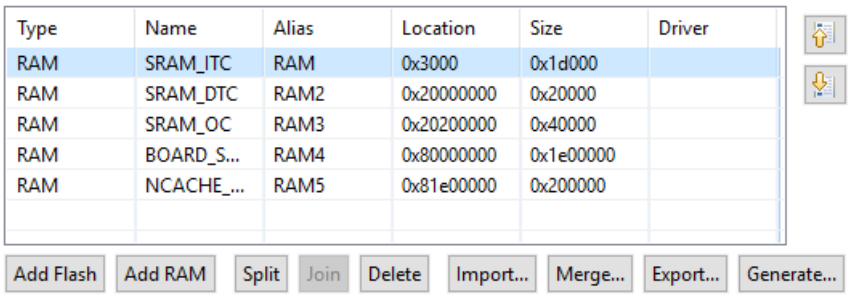

Figure 56. **SRAM_ITC**

1. 将 **SRAM_ITC**移动到第一个，使其成为默认配置。
2. 生成 image 文件。您可以找到生成的源 image 名为 `Debug\\evkmimxrt10\#\#\_iled\_blinky.axf`。

- **KeilMDK 5**
  1. 在工具栏中，选择 **iled_blinky** debug target。
  2. 打开 **Project > Options > Linker** ，点击 **Edit**以修改 Scatter 文件。
  3. 关掉窗口，在 linker 文件中作以下修改：（修改部分加亮显示）。

```
#define m_interrupts_start 0x00003000
#define m_interrupts_size 0x00000400

#define m_text_start 0x00003400
#define m_text_size 0x0001DC00
```

  4. 生成 image 文件。
     您可以找到生成的源 image 名为 `boards\\evkmimxrt10\#\#\\demo\_apps\\led\_blinky\\mdk\\debug\\iled\_blinky.hex`。
- **IAR Embedded Workbench**
  1. 选择 **Project < Edit Configurations … > Debug**。
  2. 在工程的根目录打开 *MIMXRT10##xxxxx_ram.icf* ，然后在文件中作以下修改：

```
define symbol m_interrupts_start = 0x00003000;
define symbol m_interrupts_end = 0x000033FF;

define symbol m_text_start = 0x00003400;
define symbol m_text_end = 0x0001FFFF;
```

  3. 对于多核处理器来说，在 **Project > Options… > General Options > Target**中选择使用核。例如，在 RT1176 平台的 *iled_blinky_cm7* 工程中，选择处理器使用核为 *Cortex-M7*。
  4. 保存修改文件并生成 source image。
     最终，image 生成为 *boards\evkmimxrt10##\demo_apps\led_blinky\iar\debug\iled_blinky.out*。

6.12.1.3 Image running from external SDRAM

- **MCUXpresso IDE**
  1. 在 **Project > Properties - C/C++ Build > Settings > Tool Settings > MCU Linker > Managed Linker Script** 中检查是否勾选 Link application to RAM。
  2. 选择 **Project > Properties - C/C++ Build > Settings > Tool Settings > MCU C Compiler > Preprocessor**，并添加 defined symbol **SKIP_SYSCLK_INIT=1**。
  3. 在 **Project > Properties > C/C++ Build > MCU settings** 中，删除 **Flash**， 修改 BOARD_SDRAM 从*0x80002000*开始，大小设置为 *0x1dfe000*。将 BOARD_SDRAM 移动到第一个，使其成为默认值。

4. 生成 image。 最终，您可以找到名为 *Debug\evkmimxrt10##_iled_blinky.axf* 的最终源 image。

- **KeilMDK 5**
  1. 在工具栏中，选择 **iled_blinky sdram_debug**。
  2. 打开 **Project > Options > Linker** ，点击 **Edit**以修改 Scatter 文件。
  3. 关掉窗口，在 linker 文件中作以下修改：（修改部分加亮显示）。

```
#define m_interrupts_start 0x80002000
#define m_interrupts_size 0x00000400

#define m_text_start 0x80002400
#define m_text_size 0x0001DC00

#define m_data_start 0x80020000
#define m_data_size 0x01DE0000
```

  4. 生成 image。
     最终，image 生成为 *boards\evkmimxrt10##\demo_apps\led_blinky\mdk\sdram_debug\iled_blinky.hex*。
- **IAR Embedded Workbench**
  1. 选择 **Project > Edit Configurations … > sdram_debug**。
  2. 在工程的根目录打开 *MIMXRT10##xxxxx_sdram.icf* ，然后在文件中作以下修改：

```
define symbol m_interrupts_start = 0x80002000;
define symbol m_interrupts_end = 0x800023FF;

define symbol m_text_start = 0x80002400;
define symbol m_text_end = 0x8001FFFF;

define symbol m_data_start = 0x80020000;
define symbol m_data_end = 0x8002FFFF;

define symbol m_data2_start = 0x80200000;
define symbol m_data2_end = 0x8023FFFF;

define symbol m_data3_start = 0x80300000;
define symbol m_data3_end = 0x81DFFFFF;

define symbol m_ncache_start = 0x81E00000;
define symbol m_ncache_end = 0x81FFFFFF;
```

  3. 对于多核处理器来说，在 **Project > Options… > General Options > Target**中选择使用核。例如，在 RT1176 平台的 *iled_blinky_cm7* 工程中，选择处理器使用核为 *Cortex-M7*。
  4. 保存修改文件并生成 source image。您可以找到构建后的最终 image 名为 `boards\\evkmimxrt10\#\#\\demo\_apps\\led\_blinky\\iar\\sdram\_debug\\iled\_blinky.out`。

6.12.2 Connecting the board for RT10xx/RT116x/RT117x devices

本章节包含如何配置以下型号的评估板及如何将其连接到 SEC 的示例：

- MIMXRT1010-EVK
- MIMXRT1015-EVK
- MIMXRT1020-EVK
- MIMXRT1024-EVK
- MIMXRT1040-EVK
- MIMXRT1050-EVKB
- MIMXRT1060-EVK

- MIMXRT1064-EVK
- MIMXRT1160-EVK
- MIMXRT1170-EVKB

1. 有关如何使用 DIP 开关设置启动模式的说明，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的 **RT1xxx EVK** 板的启动模式选择。
2. 将板子上的 **J1**（在 RT1176，RT1166 上为 J38）**3-4** 引脚短接，使板子使用 USB OTG 供电。
3. 使用 USB 线连接 J9（在 RT1176，RT1166 上为 J20），另一端连接 PC 电脑。
4. 确认 SEC 工具已经运行在所选开发板的环境中。更多详情，请参阅 Setting up Secure Provisioning Tool。
5. 将工具栏中的 **Boot device** 选择为 EVK 开发板上所用的 NOR flash（例如 *flex-spinor/ISxxxx*）。
6. 选择连接方式为 **USB**，然后测试是否连接上。

从 **SD** 卡中启动

从 SD 卡中启动需要以下两个步骤：

1. 将 micro SDHC 卡插入开发板中。
2. 在安全配置工具中， 选择工具栏中的 **Boot memory: sdhc_sd_card/SDHC SD card 8 GB**。

RT1xxx EVK 板的启动模式选择

启动模式/设备	串行引导程序（**ISP** 模式）	**Flex-SPI NOR (QSPI, HyperFlash)**	**Flex-SPI NOR + 加密 XIP (BEE 或 OTFAD/IEE)**	**SD** 卡	**eMMC**	**SEMC NAND**	**FlexSPI NAND**
RT1010-EVK	SW1: 0001	SW8: 0010	SW8: 0010	N/A	N/A	N/A	N/A
RT1015-EVK	SW1: 0001	SW8: 0010	SW8: 1010	N/A	N/A	N/A	N/A
RT1020-EVK	SW1: 0001	SW8: 0010	SW8: 1010	SW8: 0110	N/A	N/A	N/A
RT1024-EVK	SW1: 0001	SW8: 0010	SW8: 1010	SW8: 0110	N/A	N/A	N/A
RT1040-EVK	SW4: 0001	SW4: 0010	SW4:0010 或 0110 SW2: 1000	SW4: 1010	N/A	N/A	N/A
RT1050-EVKB	SW7: 0001	SW7: 0110	SW7:0010 或 0110 SW5: x100	SW7: 1010	N/A	N/A	N/A
RT1060-EVK	SW7: 0001	SW7: 0010	SW7:0010 或 0110 SW5: x100	SW7: 1010	N/A	N/A	N/A
RT1064-EVK	SW7: 0001	SW7: 0010	SW7:0010 或 0110 SW5: x100	SW7: 1010	N/A	N/A	N/A
RT1160-EVK	SW1: 0001 SW2: 0000000000	SW1: 0010 SW2: 0000000000	SW1: 0010; SW2: 0100000000	SW1: 0010; SW2: 0000001000	SW1: 0010 SW2: 0000010000	SW1: 0010 SW2: 0000010000	N/A
RT1170-EVKB	SW1: 0001 SW2: 0000000000	SW1: 0010 SW2: 0000000000	SW1: 0010 SW2: 0100000000	SW1: 0010 SW2: 0000001000	SW1: 0010 SW2: 0000010000	SW1: 0010 SW2: 0000010000	N/A

6.12.3 Booting images for RT10xx/RT116x/RT117x devices

本章描述了 bootable images 的生成和写入过程。

您可以使用以下几种组合：

执行 **images** 的内存	写入 **image** 的内存	需要DCD/XMCD	XIP
外部 NOR flash	外部 NOR flash	无	有
内部RAM	外部 NOR 或 NAND flash	无	无
内部 RAM	SD 卡或 eMMC	无	无
SDRAM	外部NOR 或 NAND flash	有	无
SDRAM	SD 卡或 eMMC	有	无

附加信息：

- 内存，执行 **image** 的地方 — 请参阅 Preparing source image for RT10xx/RT116x/RT117x devices.
- 内存，写入 **image** 的地方 — 在 SEC 工具中的 **Boot memory**中进行配置。

注意： RT116x/7x 设备中，FlexSPI NOR/NAND 启动设备支持两个 FlexSPI 实例。`FLEXSPI_INSTANCE`fuse`\(BOOT_CFG2\[3\]\)` 或 GPIO 启动引脚决定使用哪个FlexSPI实例。设置相应的GPIO启动引脚以使用实例2，而不烧断fuse。

6.12.3.1 Booting unsigned image

开发中经常使用未签名 image。建议在使用签名 image 前先使用未签名 image 验证是否能 正常工作。

首先，生成一个 bootable image：

1. 请确保您已在**Toolbar**中选择**Unsigned boot type**。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for RT10xx/RT116x/RT117x devices 中生成的 image 作为**Source executable image**。
4. 对于在 SDRAM 中运行的 image，需要额外使用 DCD 或 XMCD (RT116x/7x) 文件进行配置。EVK 开发板可以使用该 DCD 文件：`data\targets\MIMXRT1###/evkmimxrt1xxx_SDRAM_dcd.bin`。对于 RT116x/7x，可以使用以下 XMCD 配置文件： `data\targets\MIMXRT11##/evkmimxrt11xx_xmcd_semc_sdram_simplified.yaml`.
   注意： 有关 DCD 文件的定制，请参阅Creating/Customizing DCD files。
5. 如果需要，打开 **Dual image boot** 并进行配置（仅支持 RT116x/7x 和 FlexSPI NOR）。
6. 单击 **Build image**按钮以生成一个 bootable image。

bootable image 生成成功后，请执行以下操作：

1. 将开发板设置为串口下载模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

6.12.3.2 Booting authenticated (HAB) image

本节对签名 image 的生成和烧写进行描述。如果要使用加密 image，可以跳过此步骤。

1. 在 工具栏的 **Boot type** 中选择 **Authenticated (HAB)**。
2. 在 **Build image** 视图上，使用 Preparing source image for RT10xx/RT116x/RT117x devices 中生成的 image 作为**Source executable image**。

3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
4. 如果需要，打开 **Dual image boot** 并进行配置。(RT116x/7x - FlexSPI NOR)
5. 勾选生命周期为 **HAB Closed**。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

进入 **Write image** 界面烧写 image。

1. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择，并重置开发板 。
2. 单击 **Write image**。
3. 在下列弹出框中，确认写入 fuses：
   - **Yes** — 确认并烧写 image 和 fuses。
     注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。
   - **No** — 不烧写 fuses，直接烧写 image。
   - **Cancel** — 取消烧写 image 和 fuses。

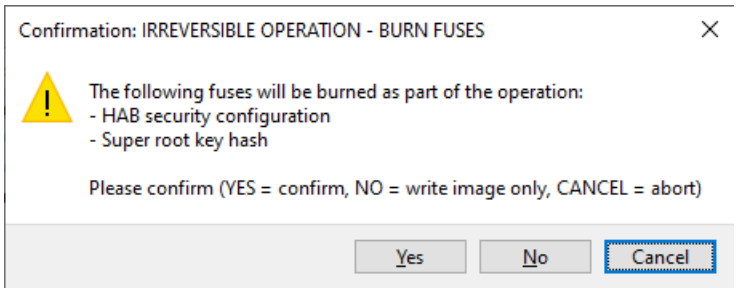

Figure 57. Burn fuses on i.MX RT1xxx for signed image

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择.）并重置开发板。

6.12.3.3 Booting encrypted (HAB) authenticated image

本节对加密 image 的生成和烧写进行了描述。该 image 将在启动操作期间解密到 RAM 中，因此无法使用 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏 中，将 **Boot type** 设置为验证的 **Encrypted (HAB)**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
4. 如果需要，打开 **Dual image boot** 并进行配置。(RT116x/7x - FlexSPI NOR)
5. 勾选生命周期为 **HAB Closed**。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。

3. 单击 **Write image**。
4. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。
   - **Cancel** — 取消烧写 image 和 fuses。

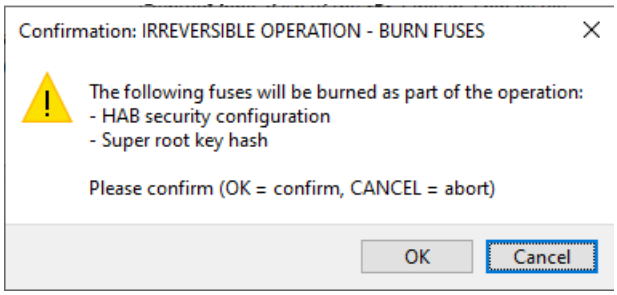

Figure 58. Burn fuses on i.MX RT1xxx for HAB encrypted and signed image

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意：加密 image 的一部分是使用来自处理器的主密钥加密的 DEK 密钥块。每个处理器的主密钥都不同，不能用于其他芯片。

注意： RT101x 和 RT102x 芯片不支持在 NOR flash 中运行加密 image。如果这些处理器不支持其他启动设备，则 **Encrypted (HAB)**验证启动类型不可用。

6.12.3.4 Booting XIP encrypted image (BEE OTPMK) authenticated (RT10xx)

本节对使用 OTP 主密钥生成和烧写 XIP 加密 image 进行描述。首先生成签名 image，然后在烧写时实时加密。使用 BEE 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏的**Boot type** 中选择 **XIP encrypted (BEE OTPMK) authenticated**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
4. 单击 **XIP encryption (BEE OTPMK)** ，打开 **BEE OTPMK**窗口 。在该窗口中，使用默认配置加密整个 image 或者在 第一个 BEE 加密区内配置 FAC 保护区域范围。
5. 勾选生命周期为 **HAB Closed**。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 通过设置相应的 GPIO 引脚启用 XIP 加密（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
4. 单击 **Write image**。
5. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。

注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。

- **Cancel** — 取消烧写 image 和 fuses。

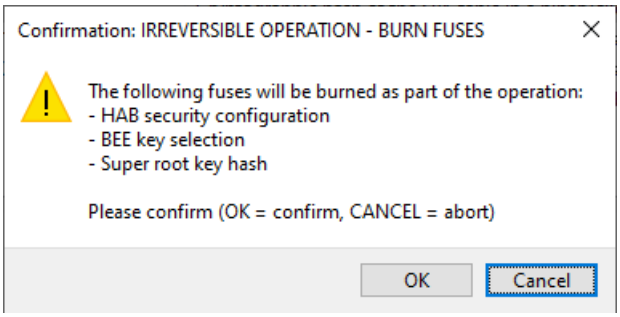

Figure 59. Burn fuses on i.MX RT10xx for BEE OTPMK encrypted image

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 步骤 5 可以通过在 OTP 配置中设置加密 XIP fuse 来代替。

6.12.3.5 Booting XIP encrypted image (BEE user keys) unsigned (RT10xx)

本节对使用用户密钥生成和烧写 XIP 加密 image 进行描述。image 生成分为两步。首先，生成未签名 bootable image。然后对该未签名 image 进行加密，以便与启用加密的 XIP 一 起使用。使用 BEE 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏的 **Boot type** 中选择 **XIP encrypted (BEE user keys) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 单击 **XIP encryption (BEE user keys)** ，打开 BEE user keys 窗口。使用默认配置加密整个 image 或者编辑 **User keys data** 为自定义密钥。此外，在这个窗口中可以配置 BEE 的其他参数（两个区域（引擎）、区域的用户密钥、FAC 保护区域范围、随机密钥生成）。
4. 单击 **Build image**。
5. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 通过设置相应的 GPIO 引脚启用 XIP 加密（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
4. 单击 **Write image**。
5. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： fuse 只能烧写一次，烧写之后不可修改。
   - **Cancel** — 取消烧写 image 和 fuses。

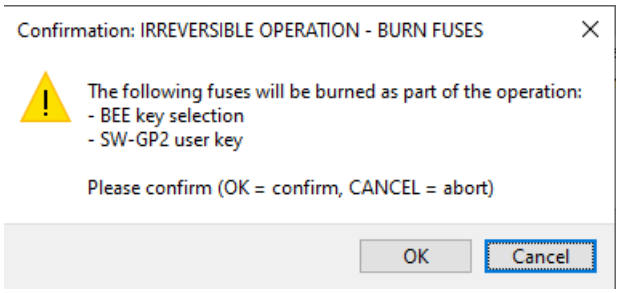

Figure 60. 使用用户密钥对 **i.MX RT10xx** 上的 **BEE** 加密进行烧写 **fuses**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 步骤 5 可以通过在 OTP 配置中设置加密 XIP fuse 来代替。

6.12.3.6 Booting XIP encrypted image (BEE user keys) authenticated (RT10xx)

本节对使用用户密钥生成和烧写 XIP 加密 image 进行描述。image 生成分为两步。首先， 生成未签名 bootable image。然后对该未签名 image 进行加密，以便与启用加密的 XIP 一 起使用。使用 BEE 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏的**Boot type**中，选择 **XIP encrypted (BEE user keys) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
4. 单击 XIP encryption (BEE user keys)，打开 BEE user keys 窗口，在窗口中，使用默认配置加 密整个 image 或者编辑 User keys data 为自定义密钥。此外，在这个窗口中可以配置 BEE 的其他参数（两个区域（引擎）、区域的用户密钥、FAC 保护区域范围、随机密钥生成）。
5. 勾选生命周期为 **HAB Closed**。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 通过配置相应的 GPIO 引脚来启用XIP加密。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
4. 单击 **Write image**。
5. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。
   - **Cancel** — 取消烧写 image 和 fuses。

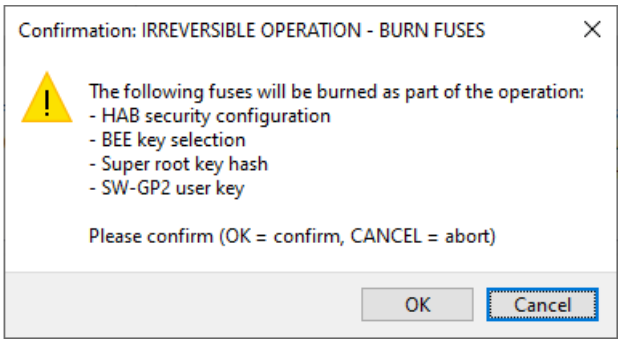

Figure 61. 在 **i.MX RT10xx** 上烧写 **BEE** 加密和签名**image**的**fuse**

如果写入操作成功，请切换启动模式（请参阅为 RT10xx/RT116x/RT117x 设备准备源 image 文件），并复位开发板。

注意： 步骤 5 可以通过在 OTP 配置中设置加密 XIP fuse 来代替。

6.12.3.7 Booting XIP encrypted image (OTFAD OTPMK) authenticated (RT10xx)

本节对使用 OTP 主密钥生成和烧写 XIP 加密 image 进行描述。首先生成签名 image，然后在烧写时实时加密。使用 OTFAD 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏上，设置 **Boot type** 为 **XIP encrypted (OTFAD OTPMK) authenticated**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
4. 点击 **XIP encryption (OTFAD OTPMK)**，打开 **OTFAD OTPMK**窗口。使用默认配置加密整个 image 或者在自定义配置保护区域范围。
5. 勾选生命周期为 **HAB Closed**。
6. 单击 **Build image**。
7. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 单击 **Write image**。
4. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。
   - **Cancel** — 取消烧写 image 和 fuses。

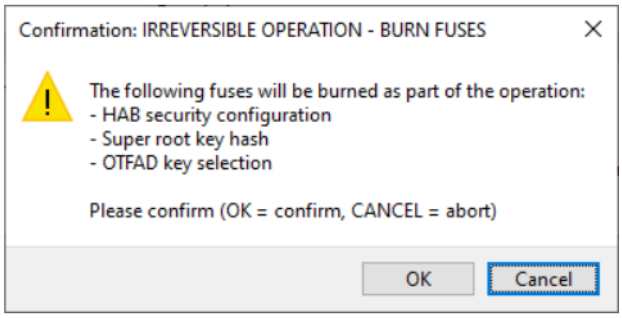

Figure 62. 在 **i.MX RT10xx** 上烧写 **OTFAD** 加密和签名**image**的**fuse**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

6.12.3.8 Booting OTFAD encrypted image unsigned with user keys.

本节对使用用户密钥生成和烧写 OTFAD 加密 image 进行描述。image 生成分为两步。

下面对如何生成 image 进行详细描述：

1. 在工具栏中，将 **Boot Type** 设置为 **Encrypted**，对于 RT116x/7x 芯片，选择 **(OTFAD) unsigned**，对于 RT10xx 芯片，选择 **XIP encrypted (OTFAD user keys) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 单击 **OTFAD encryption/XIP encryption**（**OTFAD**用户密钥），打开OTFAD configuration（OTFAD 配置）窗口。在该窗口生成随机密钥。也可以配置 OTFAD regions (contexts)，KEK source (OTP or KeyStore)，KEK，Key scramble，user keys for regions，regions ranges 和 random key generation 等功能。
4. 打开 **OTP configuration** 窗口，检查并修改错误配置。
5. 单击 **Build image**。
6. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 如果OTFAD KEK源设置为PUF KeyStore，则重置主板。必须确保 KeyStore 成功注册。
4. 通过设置相应的 GPIO 引脚启用 XIP 加密(RT116x/7x)（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
5. 单击 **Write image**。
6. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： fuse 只能烧写一次，烧写之后不可修改。
   - **Cancel** — 取消烧写 image 和 fuses。

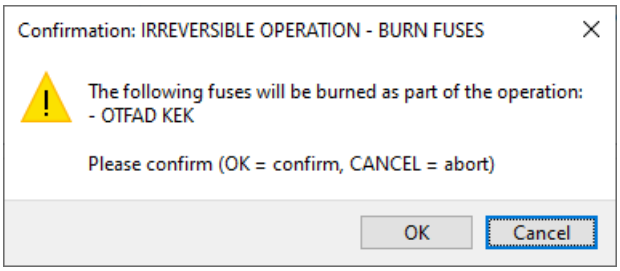

Figure 63. 烧写 **i.MX RT10xx/RT116x/RT117x** 上的**fuse**，以进行 **OTFAD** 加密**image**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 第 6 步（RT116x/7x）可以通过在 OTP 配置中设置 `ENCRYPT_XIP_EN` fuse 来代替。

6.12.3.9 Booting OTFAD encrypted image authenticated with user keys

本节对生成和烧写 OTFAD 加密 image 进行描述。image 生成分为两步。首先，生成签名 bootable image。然后对该签名 image 进行加密，以便与 OTFAD 一起使用。使用 OTFAD 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 对于 RT116x/7x 芯片，在 工具栏的 **Boot type** 中选择 **Encrypted (OTFAD) authenticated**。对于 RT10xx 芯片，选择 **XIP encrypted (OTFAD user keys) authenticated**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 请确保您在 **PKI management** 视图中生成了密钥。更多详情，请参阅 PKI management。
4. 对于 **Authentication key**，选择任意密钥，例如 *SRK1: IMG1_1+CSF1_1*。
5. 单击 **OTFAD encryption/XIP encryption**（**OTFAD**用户密钥），打开OTFAD configuration（OTFAD 配置）窗口。生成随机密钥。该窗口可以设置随机密钥，也可以配置 OTFAD regions (contexts)，KEK source (OTP or KeyStore)，KEK，Key scramble，user keys for regions，regions ranges 和 random key generation 等功能。
6. 勾选生命周期为 **HAB Closed**。
7. 打开 **OTP configuration** 窗口，检查并修改错误配置。
8. 单击 **Build image**。
9. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 如果 OTFAD KEK 已经配置到 KeyStore 中，请复位开发板。必须确保 KeyStore 成功 注册。
4. 通过设置相应的 GPIO 引脚启用 XIP 加密(RT116x/7x)（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
5. 单击 **Write image**。
6. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： 烧断 fuse 只能进行一次，之后处理器只能执行经过认证的 image。
   - **Cancel** — 取消烧写 image 和 fuses。

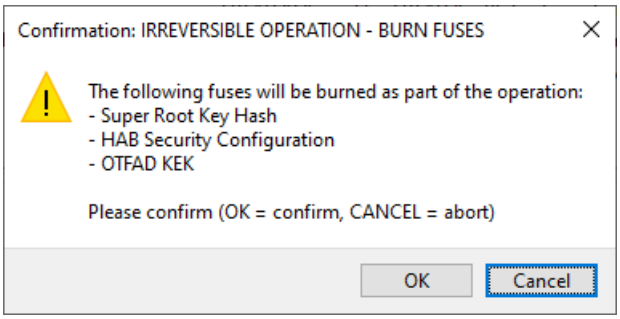

Figure 64. 对 **i.MX RT10xx/RT116x/RT117x** 进行烧写**fuse**，以生成 **OTFAD** 加密和调整大小的**image**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 第 6 步（RT116x/7x）可以通过在 OTP 配置中设置 `ENCRYPT_XIP_EN` fuse 来代替。

6.12.3.10 Booting IEE encrypted image unsigned (RT116x/7x)

本节对生成和烧写 IEE 加密 image 进行描述。image 本身分两步生成。首先生成未签名 bootable image，然后将此未签名 image 使用 IEE 加密。使用 IEE 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏 中，将 **Boot type** 设置为 **Encrypted (IEE) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 单击 **IEE encryption**，打开 IEE Configuration 窗口。在该窗口生成随机密钥。该窗口允许配置 IEE regions (contexts)，KEK，AES encryption mode，user keys for regions，regions ranges 和 random key generation 等功能。
4. 打开 **OTP configuration**，查看设置，并修复所有报告的问题。
5. 单击 **Build image**。
6. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 视图。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 复位开发板。必须确保密钥库成功注册。
4. 通过设置相应的 GPIO 引脚启用 XIP 加密（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
5. 单击 **Write image**。
6. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： fuse 只能烧写一次，烧写之后不可修改。
   - **Cancel** — 取消烧写 image 和 fuses。

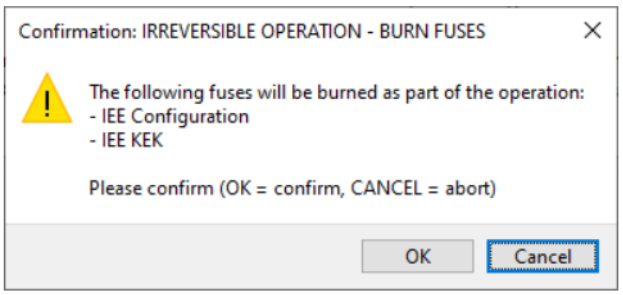

Figure 65. 为 **RT116x/7x** 上的 **IEEE** 加密**image**烧写**fuse**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 步骤 5 可以通过在 OTP 配置中设置 `ENCRYPT_XIP_EN` fuse 来代替。

6.12.3.11 Booting IEE encrypted image authenticated (RT116x/7x)

本节对生成和烧写 IEE 加密 image 进行描述。image 本身分两步生成。首先生成未签名 bootable image，然后将此未签名 image 使用 IEE 加密。使用 IEE 功能的 source image 必须是 XIP image。

下面对如何生成 image 进行详细描述：

1. 在工具栏 的 **Boot type** 中选择 **Encrypted (IEE) authenticated**。
2. 作为**Source executable image**，使用从 Preparing source image for RT10xx/RT116x/RT117x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 请确保您在 **PKI management** 视图中生成了密钥。更多详情，请参阅 PKI management。
4. 对于 **Authentication key**，选择任意密钥，例如 SRK1: IMG1_1+CSF1_1。
5. 单击 **IEE encryption**，打开 IEE Configuration 窗口。在该窗口生成随机密钥。该窗口允许配置 IEE regions (contexts)，KEK，AES encryption mode，user keys for regions，regions ranges 和 random key generation 等功能。
6. 勾选生命周期为 **HAB Closed**。
7. 打开 **OTP configuration**，查看设置，并修复所有报告的问题。
8. 单击 **Build image**。
9. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择。
3. 复位开发板。必须确保密钥库成功注册。
4. 通过设置相应的 GPIO 引脚启用 XIP 加密（更多详情，请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）。
5. 单击 **Write image**。
6. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
     注意： fuse 只能烧写一次，烧写之后不可修改。
   - **Cancel** — 取消烧写 image 和 fuse。

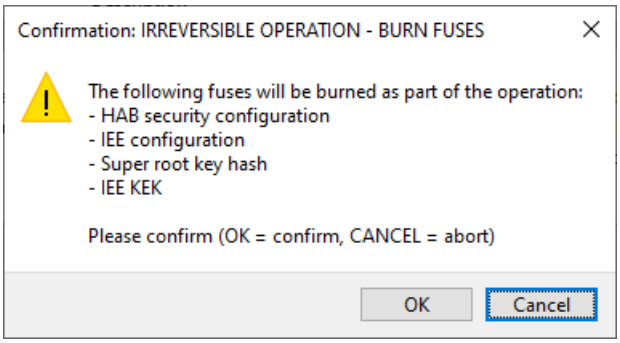

Figure 66. 在 **RT116x/7x** 上为 **IEEE** 加密和签名**image**烧写**fuse**

如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT10xx/RT116x/RT117x devices 中的**RT1xxx EVK** 板的启动模式选择）并重置开发板。

注意： 步骤 6 可以通过在 OTP 配置中设置 `ENCRYPT_XIP_EN` fuse 来代替。

6.12.4 Creating/Customizing DCD files

推荐使用 MCUXpresso Config Tools 或 MCUXpresso IDE 创建 DCD 文件。

1. 在上述两个工具中打开任意芯片工程。
2. 在 **File > Import > MCUXpresso Config Tools > Import Source** 中添加 SDK 文件中自带的 DCD 配置文件。
3. 使用 SDK 例程包中的 *boards\evkmimxrt10##\xip\evkmimxrt10##_sdram_ini_dcd.c*文件。
4. 如要切换到设备配置工具，选择 **main menu > Config Tools > Device Configuration**。
5. 在工具栏的 **DCD view**中，切换 **Output Format**为 **binary**。
6. 进入 **Code Preview** 界面，然后在工具栏点击 **Export**，填写 DCD 文件生成目标文件夹路径。
   注意： 更多详情，请参阅设备配置工具文档。

6.13 RT118x device workflow

本节详细介绍 RT118x 设备工作流程。

6.13.1 Preparing source image for RT118x devices

在此步骤中，选择将要执行 image 的目标存储器。以下选项可用于 RT118x 设备：

- 从外部 **NOR flash** 运行**image**
  这就是所谓的XIP **XIP**(e**X**ecution **I**n **P**lace) image。这意味着 image 直接从其所在的内存中执行。
- 在 **RAM** 中运行**image**
  该image可以存储在 SD 卡/eMMC 或外部闪存（FlexSPI NOR、FlexSPI NAND）中。启动时，它将被复制到 RAM 中并从那里执行；或者，也可以使用串行下载器将其加载到 RAM 中。支持以下 RAM 类型：
  – 内部 RAM
  – SDRAM
  – HyperRAM 源 image 的准备和 RT116x/7x 相似。请参阅Preparing source image for RT10xx/RT116x/RT117x devices。一些 RT118x 特性：

- 选择 core cm33 的示例，然后将目标设置为 Cortex-M33。 ELE 固件分布在 MCUXpresso SDK 的“ELE crypto”组件中，通常是 `firmware\edgelock\mxrt1180-ahab-container.img` 文件。
- 对于 MCUXpresso IDE 和内部 RAM 中的 CM33 镜像，需要对项目进行以下修改。这些更改已在 MCUXpresso IDE 版本 25.06 和 MCUXpresso SDK 版本 25.09 中得到验证：
  - 确保 ITC 是可用内存区域中列出的第一个 RAM（请参阅 *project Properties* > *C/C++ Build* > *MCU settings*）。
  - 将全局数据放置和 “CodeQuickAccess” 的值更改为 DTC_cm33，并确保选中 “Link application to RAM”（请参阅 *project Properties* > *C/C++ Build* > *Settings* > *Managed Linker Script*）。
  - 更改`startup_mimxrt1189_cm33.c`代码中`data_init()`和`bss_init()`函数： ``` **attribute** ((章节(“.after_vectors.init_data”))) void data_init(unsigned int romstart, unsigned int start, unsigned int len) { if (start == romstart) return; unsigned int *pulDest* = *(unsigned int)* start; unsigned int *pulSrc* = *(unsigned int)* romstart; unsigned int loop; for (loop = 0; loop < len; loop = loop + 4) *pulDest++ = *pulSrc++; }

  attribute ((section(“.after_vectors.init_bss”))) void bss_init(unsigned int start, unsigned int len) { unsigned int *pulDest* = *(unsigned int)* start; unsigned int loop; for (loop = 0; loop < len; loop += 4, ++pulDest) if (*pulDest != 0) *pulDest = 0; } ```

6.13.2 Connecting the board for RT118x devices

本节包含有关配置 RT118x 板并将其连接到 SEC 工具的信息。

EVK/FRDM boards 开发板概述：

EVK/FRDM 开发板	电源选择跳线	USB	UART	FlexSPI实例1	FlexSPI实例2
MIMXRT1180-EVK	J1（通过 USB 接口 3-4 路；通过 UART 接口 7-8 路）	J33	J53	FlexSPI NOR	HyperRAM
MIMXRT1180A-EVK	J1（通过 USB 接口 3-4 路；通过 UART 接口 7-8 路）	J33	J53	FlexSPI NOR	HyperRAM
MIMXRT1180-144	J1（通过 UART 接口 3-4 路）	N/A	J53	FlexSPI NOR	HyperRAM
FRDM-IMXRT1186	J1（通过 USB 接口 3-4 路；通过 UART 接口 5-6 路）	J63	J23	HyperRAM	FlexSPI NOR

注意：

- 对于 FRDM-IMXRT1186，SEC 工具会自动配置从 FlexSPI 实例 2（FlexSPI NOR）启动，这会导致 BOOT_CFG2 号fuse 被烧断。

1. 有关如何使用 DIP 开关设置启动模式的说明，请参阅 连接 RT10xx/RT116x/RT117x 设备的开发板 中的 **RT1xxx EVK** 板的启动模式选择。
2. 确保将电源选择跳线设置为 3-4，以便通过 USB（RT1180-144 上的 UART）为电路板供电。
3. 通过 USB 线缆将开发板连接到 PC，再通过 UART 或 USB 接口进行连接。
4. 确认 SEC 工具已经运行在所选开发板的环境中。更多详情，请参阅设置安全配置工具 。
5. 确保 **Boot Memory Configuration** 中的 **Boot device**与 EVK/FRDM 板上使用的 FlexSPI NOR flash 相匹配。

6. 将连接设置为您选择的连接方式（**USB** 或 **UART**），并测试板连接。

6.13.2.1 Booting from SD card

从 SD 卡中启动需要以下两个步骤：

1. 将 micro SDHC 卡插入开发板中。
2. 在 **Boot Memory Configuration** 中选择 **SD card, SDHC SD-card 8GB USDHC1**。

6.13.2.2 Booting from eMMC

要从 eMMC 启用，请执行以下操作：

1. eMMC 可以安装在主板上，也可以通过 SD/eMMC 适配器的 SD 插槽连接 eMMC。
2. 在 **Boot Memory Configuration** 中选择 **eMMC, SDHC eMMC 8 GB USDHC1**。

6.13.3 Booting from serial downloader

要从串口下载器启动，请执行以下操作：

1. 在板载启动模式开关上设置ISP启动模式
2. 在**Boot Memory Configuration**中选择**RAM via serial downloader**。

6.13.3.1 Table: Boot mode selection for RT118x EVK boards

启动模式/设备	串行引导程序（**ISP** 模式）	**FlexSPI NOR**	**SD** 卡/**eMMC**	**FlexSPI NAND**
RT1180-EVK	SW5: x001	SW5: x100	SW5: x011	SW5: x101 (N/A)
RT1180A-EVK	SW5: x001	SW5: x100	SW5: x011	SW5: x101 (N/A)
RT1180-144	SW5: x100	SW5: x001	SW5: x110	SW5: x101 (N/A)
FRDM-IMXRT1186	J60: 100	J60: 001	J60: 110	J60: 101（不适用）

6.13.4 Booting images for RT118x devices

本章描述了 bootable images 的生成和写入过程。

您可以使用以下几种组合：

执行 **images** 的内存	启动存储器：写入 **image** 的内存	需要的 **XMCD**	**XIP**
外部 NOR flash	外部 NOR flash	无	有
内部RAM	外部 NOR 或 NAND flash	无	无
内部 RAM	SD 卡或 eMMC	无	无
SDRAM/HyperRAM	外部 NOR 或 NAND flash	有	无
SDRAM/HyperRAM	SD 卡或 eMMC	有	无

注意：

- 内存，执行 image 的地方 — 请参阅Preparing source image for LPC55(S)0x/1x/2x/6x devices。
- 写入 image 的内存被配置为 SEC 中的引导内存。

6.13.4.1 Booting unsigned image

开发中经常使用未签名 image。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。

首先，生成一个 bootable image：

1. 请确保您已在**Toolbar**中选择**Unsigned boot type**。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for RT118x devices 中生成的 image 作为**Source executable image**。
4. 对于从 SDRAM/HyperRAM 执行的 image，请使用 XMCD 配置 SEMC SDRAM/FlexSPI HyperRAM。对于 EVK 开发板，可以使用以下 XMCD 配置文件：<SEC>/sample_data/targets/MIMXRT118#/evkmimxrt118#_xmcd_*_simplified.yaml。
5. 如果需要，打开 **Dual image boot** 并进行配置。
6. 单击 **Build image**按钮以生成一个 bootable image。

bootable image 生成成功后，请执行以下操作：

1. 确保该开发板为串行引导加载程序 (ISP) 模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT118x devices 中的 **RT118x EVK** 板的启动模式选择）并重置开发板。

6.13.4.2 Booting signed image

本章节描述了带签名的 image 的生成和写入过程。本步骤需要在此步骤需要在 **PKI management** 视图进行密钥生成。对于更多关于密钥生成的信息，请参阅生成密钥。如果要使用加密 image，可以跳过此步骤。

首先，生成一个 bootable image：

1. 在工具栏中，设置 **Boot type** 为**Signed**。
2. 在 **Build image** 视图上，使用 Preparing source image for RT118x devices 中生成的 image 作为**Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1*。
4. 勾选生命周期为 **OEM Closed**。
5. 单击 **Build image**。
6. 检查 bootable image 是否成功生成。

进入 **Write image** 界面烧写 image。

1. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择。
2. 单击 **Write image**。
3. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
   - **Cancel** — 取消烧写 image 和 fuses。

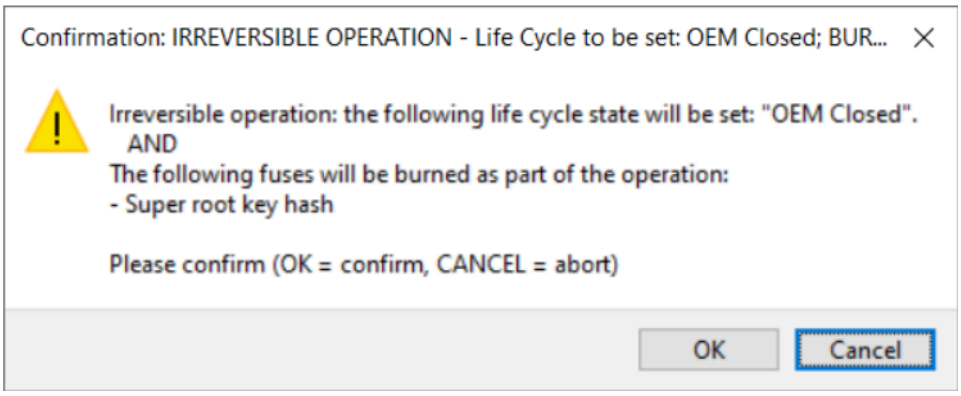

Figure 67. 在 **i.MX RT118x** 上烧写**fuse**以获取签名 **image**

1. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择）并重置开发板。

6.13.4.3 Booting encrypted (AHAB) image

本节对加密 image 的生成和烧写进行了描述。该 image 将在启动操作期间解密到 RAM 中，因此无法使用 XIP image。本步骤需要使用在 **PKI management** 视图下生成的密钥。对于更多关于密钥生成的信息，请参阅Generate keys。

下面对如何生成 image 进行详细描述：

1. 在工具栏上，选择**Boot type**为**Encrypted (HAB)**。
2. 作为**Source executable image**，使用从 Preparing source image for RT118x devices中生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1*。
4. 勾选生命周期为 **OEM Closed**。
5. 单击 **Build image**。
6. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 视图。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅**RT118x EVK** 板的启动模式选择。
3. 单击 **Write image**。
4. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
   - **Cancel** — 取消烧写 image 和 fuses。
5. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择）并重置开发板。

注意： 加密 image 的一部分是使用来自处理器的主密钥加密的 DEK 密钥块。每个处理器的主密钥都不同，不能用于其他芯片。DEK 密钥 blob 在写入期间生成，然后使用此特定于处理器的密钥 blob 更新 AHAB image。

6.13.4.4 Booting OTFAD encrypted image

本节对使用用户密钥生成和烧写 OTFAD 加密 image 进行描述。image 生成分为两步。首先生成签名/未签名 AHAB image，然后将此 AHAB image 已加密，可与 OTFAD 一起使用。使用 OTFAD 功能的 source image 必须

是 XIP image。本步骤需要使用在 **PKI management** 视图下生成的密钥。对于更多关于密钥生成的信息，请参阅生成密钥。

下面对如何生成 image 进行详细描述：

1. 在 **Toolbar** 中，将 **Boot type** 设置为 **Encrypted (IEE) signed** 或 **Encrypted (IEE) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT118x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1*。
4. 单击 **OTFAD encryption**，打开 OTFAD Configuration 窗口。该窗口可以设置随机密钥。该窗口可以设置随机密钥，也可以配置 OTFAD regions (contexts)，KEK source (OTP 或者 DUK)，KEK，Key scramble，user keys for regions，regions ranges 和 random key generation 等功能。
5. 勾选生命周期为 **OEM Closed**。
6. 打开 **OTP configuration** 窗口，查看设置，并修复所有报告的问题。
7. 单击 **Build image**。
8. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择。
3. 单击 **Write image**。
4. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
   - **Cancel** — 取消烧写 image 和 fuses。

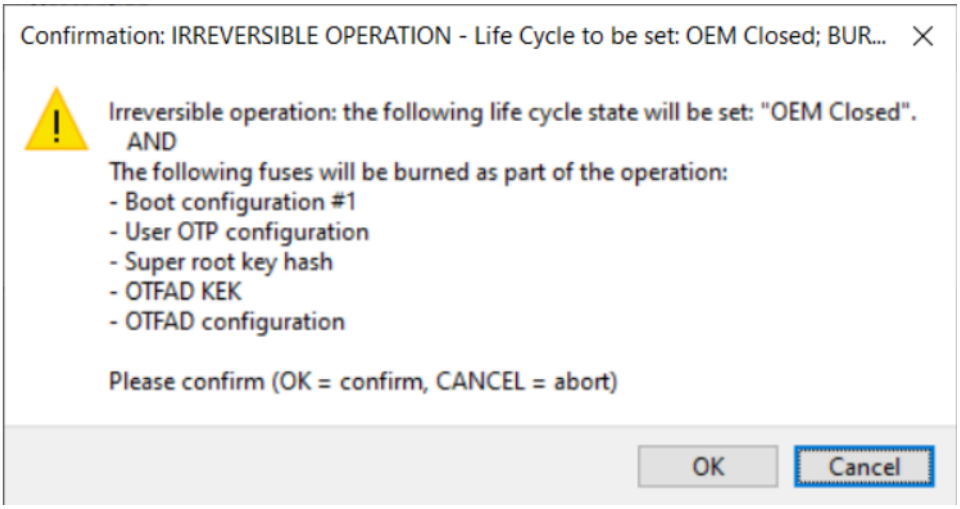

Figure 68. 在 **i.MX RT118x** 上烧写**fuse**，用于 **OTFAD** 加密 **image**

1. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择）并重置开发板。

6.13.4.5 Booting IEE encrypted

本节介绍 IEE 加密 image 的生成和写入。image 生成分为两步。首先生成签名/未签名 AHAB image，然后将此 AHAB image 使用 IEE 加密。使用 IEE 功能的 source image 必须是 XIP image。本步骤需要使用在 **PKI management** 视图下生成的密钥。对于更多关于密钥生成的信息，请参阅生成密钥。

下面对如何生成 image 进行详细描述：

1. 在工具栏中，将 **Boot type** 设置为 **Encrypted (IEE) signed** 或 **Encrypted (IEE) unsigned**。
2. 作为**Source executable image**，使用从 Preparing source image for RT118x devices中外部 NOR flash 运行生成的 image 作为 **Source executable image**。
3. 对于 **Authentication key**，选择任意密钥，例如 *SRK1*。
4. 单击 **IEE encryption**，打开 IEE Configuration 窗口。在该窗口生成随机密钥。该窗口允许配置 IEE regions (contexts)，AES encryption mode，user keys for regions，regions ranges 和 random key generation 等功能。
5. 勾选生命周期为 **OEM Closed**。
6. 打开 **OTP configuration** 窗口，查看设置，并修复所有报告的问题。
7. 单击 **Build image**。
8. 检查 bootable image 是否成功生成。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 视图。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择。
3. 单击 **Write image**。
4. 在下列弹出框中，确认写入 fuses：
   - **OK** — 确认并烧写 image 和 fuses。
   - **Cancel** — 取消烧写 image 和 fuses。

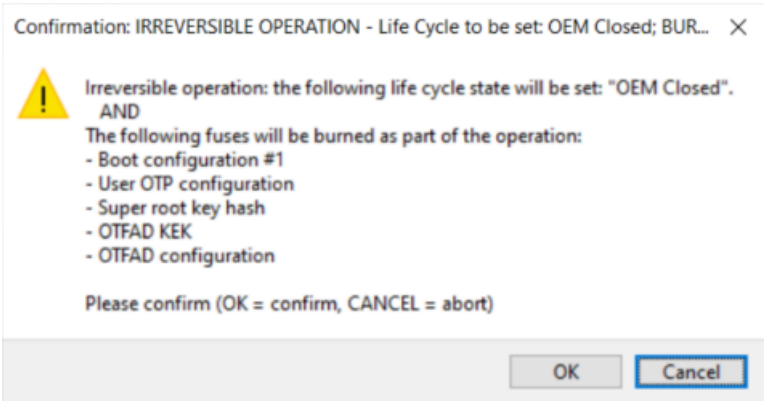

Figure 69. 在 **i.MX RT118x** 上烧写**fuse**，用于 **IEEE** 加密 **image**

1. 如果写入操作成功，则切换启动模式（请参阅 Connecting the board for RT118x devices 中的**RT118x EVK** 板的启动模式选择）并重置开发板。

6.13.4.6 Booting multicore images

本章描述了如何构建和编写多核（Cortex M33 和 Cortex M7）的 image。只有 Cortex M33 是可启动内核。用于 Cortex M33 的源 image必须触发代码中的第二个内核 (Cortex M7)。请参阅 RT118x SDK 包中的 `demo_apps\multicore_trigger` 示例。

下面对如何生成多核 image 进行详细描述：

- 在此示例中，Cortex M7 XIP image 从外部 image（起始地址 0x2800B000）运行，Cortex M33 image 从内部 RAM（起始地址 0xFFE0000）运行：

1. 在 **Build** 选项卡中设置 **Source executable image**（Cortex M33 的 image）。
2. 通过 **Additional images** 按钮打开 **Additional User/OEM Image** 对话框（应用程序二进制 image 会自动填充）。
3. 指定从 image 运行的独立 Cortex M7 可执行二进制 image 并设置以下值：
   - Image 偏移 - 0xA000。计算公式为： 加载地址 (0x2800B000) - FlexSPI NOR 基地址 (0x28000000) - FlexSPI NOR 中的 AHAB image 偏移量 (0x1000)
   - 加载地址 - 0x2800B000
   - 入口点 - 0x2800B000
   - 核心 ID - cortex-m7
   - Image 类型 - 可执行
4. 单击 **OK** 以关闭对话框。
5. 单击 **Build image**。

- 在此示例中，Cortex M7 image 从内部 ITCM RAM（起始地址 0x0）运行，Cortex M33 XIP iage从外部 image（起始地址 0x2810B000）运行。此用例需要额外的 fuses，以便可以正确访问内部 RAM。

1. 在 **Build** 选项卡中设置 **Source executable image**（Cortex M33 的image）。
2. 通过 **Additional images** 按钮打开 **Additional User/OEM Image** 对话框（应用程序二进制 image 会自动填充）。
3. 指定从 ITCM RAM 运行的独立 Cortex M7 可执行二进制 image 并设置以下值：
   - image 偏移 - 0x10F400 - 这可以是不与其他 image 重叠的任何值（在本例中，image 直接放置在 Cortex M33 image 之后）
   - 加载地址 - 0x303C0000（CM33 内核地址空间中 CM7 ITCM 的安全别名）
   - 入口点 - 0x0（CM7 地址空间中 image 的起始地址）
   - 核心 ID - cortex-m7
   - Image 类型 - 可执行
4. 单击 **OK**以关闭对话框。
5. 单击左下角的 **OTP Configuration** 按钮，打开 **OTP configuration** 对话框。
6. 将 POR_PRELOAD_MC7_TCM_ECC 和 RELEASE_M7_RST_STAT fuses（BOOT_CFG7）设置为 1 并修复所有报告的问题。
7. 单击 **OK**以关闭对话框。
8. 单击 **Build image**。

注意： 多核 image 的写入过程与 Booting IEE encrypted相同。

6.13.4.7 Life cycle for RT118x device workflow

默认的生命周期应该用于开发，它是 **OEM Open**。部署应用程序之前，请设置 **OEM Closed** 或 **OEM Locked**生命周期（更多详情，请参阅目标处理器的文档）。

注意： 生命周期的变化是不可逆的。

一旦处理器处于“OEM Closed”或“OEM Locked”模式：

- 工具处理器不允许通过 blhost 烧毁 fuse。该应用程序仍然可以更新。
- 处理器只能执行签名 image。

6.13.4.8 Firmware version

固件版本值包含在 AHAB 容器标头的 Fuse 版本字段中。当 AHAB 容器经过身份验证时，此版本值通过 ELE 提交 API 间接融合。必须从应用程序本身调用此提交 API。

启动期间无法加载 fuse 版本低于 fuse 版本的 AHAB image。

仅签名 image 支持固件版本。

6.14 RT5xx/6xx device workflow

本节详细介绍 RT5xx/6xx 设备工作流程。

6.14.1 Preparing source image RT5xx/6xx devices

在此步骤中，必须选择将要执行 image 的目标存储器。

支持的启动存储器包括 NOR flash、SD卡、和 eMMC。

以下启动类型适用于 RT5xx/6xx 处理器：

- Image 运行在外部 flash：XIP image（就地执行）
- Image 运行在内部 RAM：执行前将 image 从 FLASH 复制到 RAM。

推荐使用 **gpio_led_output** 示例进行验证，该 image 可正常运行。默认情况下，该示例仅在按下用户按钮（user button）时触发 LED 闪烁，但可以轻松将其修改为一直闪烁。

6.14.1.1 Image running in external flash

gpio_led_output工程默认链接到外部 flash。必须禁用 XIP 启动头, 它将会由 SEC 创 建。

- **MCUXpresso IDE**
  1. 或者，可选择打开 Project > Properties > C/C++ Build > Settings > MCU C Compiler > Preprocessor > Defined symbols 并将 `BOOT_HEADER_ENABLE`设置为 `0`。
  2. 生成 image 文件。最终生成的源 image 被命名为 `Debug\\evkmimxrt685_gpio_led_output.axf`。它可以用作 SEC 工具中 bootable image 文件的输入。
- **Keil MDK**
  1. 在工具栏 中, 选择 **gpio_output_flash_debug**。
  2. 或者，打开 **Project > Options > “*C/C++*”**，禁用 define symbol`BOOT_HEADER_ENABLE=0`（设置为 0）。
  3. 在 **Project > Options > Output**中，勾选 **Create HEX file**。
  4. 仔细检查应用程序是否链接到 0x8001000。如果不是，则必须对链接器应用以下修复程序作为该问题的解决方法：

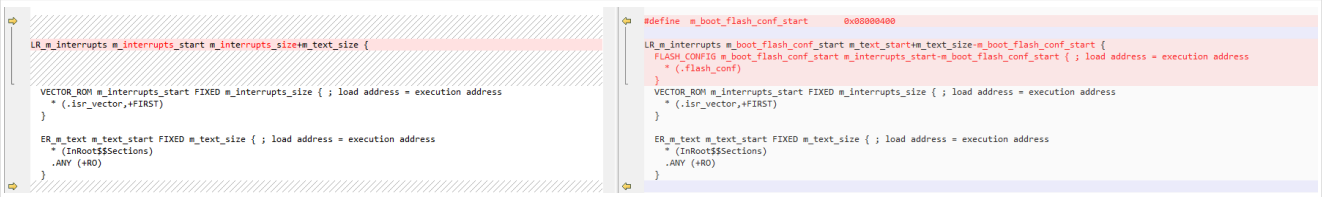

Figure 70. Keil MDK 解决方法

1. 生成 image 文件。输出 image 为 `boards\evkmimxrt685\driver_examples\gpio\led_output\mdk\flash_debug\gpio_led_output.out`。

- **IAR Embedded Workbench**
  1. 在 **Project > Edit Configurations ...** 中，选择 `flash_debug`。
  2. 或者，在 **Project Options > C/C++ Compiler > Preprocessor > Defined Symbols** 中，添加或修改现有的 `BOOT_HEADER_ENABLE`，将其定义为 `0` 。
  3. 生成 image 文件。输出 image 为 `boards\evkmimxrt###\driver_examples\gpio\led_output\iar\flash_debug\gpio_led_output.out`。

6.14.1.2 Image running in internal RAM

Image 运行在内部 RAM

- **MCUXpresso IDE**
  1. 或者，可选择打开 **Project > Properties > C/C++ Build > Settings > MCU C Compiler > Preprocessor > Defined symbols** 并将`BOOT_HEADER_ENABLE`设置为`0`。
  2. 选择 **Project > Properties - C/C++ Build > Settings > Tool Settings > MCU Linker > Managed Linker Script**，勾选 `Link application to RAM.`。
  3. 生成 image 文件。构建好的 image 文件为 `Debug\evkmimxrt685_gpio_led_output.axf`。该 image 可以在 SEC 工具中的 **Source executable image** 中进行使用。
- **Keil MDK**
  由于 MDK 的示例项目未内置到 RAM 中，因此您必须手动修改链接器文件。本文档尚未记录如何配置该工具。
- **IAR Embedded Workbench**
  1. 在 **Project > Edit Configurations ...** 中，选择 `debug`。
  2. 或者，在 **Project Options > C/C++ Compiler > Preprocessor > Defined Symbols** 中，添加或修改现有的 `BOOT_HEADER_ENABLE`，将其定义为 `0` 。
  3. 生成 image 文件。输出 image 构建为 `boards\evkmimxrt###\driver_examples\gpio\led_output\iar\debug\gpio_led_output.out`。

6.14.2 Connecting the board for RT5xx/6xx devices

本章节包含如何配置以下型号的评估板及如何将其连接到 SEC 的示例：

- MIMXRT595-EVK
- MIMXRT685-AUD-EVK

RT5xx/6xx EVK 板启动配置

启动模式/设备	ISP 模式	FlexSPI 启动	SD卡	eMMC
MIMXRT595-EVK	SW7[1:3]: 100 (UART、SPI、I2C)SW7[1:3]: 101 (USB)	SW7[1:3]: 001	SW7[1:3] 011	SW7[1:3] 110
MIMXRT685-AUD-EVK	SW5[1:3]: 100	SW5[1:3]: 101	SW5[1:3] 011	SW5[1:3] 110

其中**1**表示开关处于开启状态，**0**表示开关处于关闭状态。

1. 将开发板切换至 ISP 模式并复位。更多详细信息，请参阅上表。
2. 使用 USB 线将 **J7** 连接至 PC 机。
3. 请确保您已使用 workspace 启动 SEC。更多详情，请参阅设置安全配置工具 。
4. 选择连接方式为 **USB**，然后测试是否连接上。

从 **SD** 卡中启动：

从 SD 卡中启动需要以下两个步骤：

1. 在安全配置工具中将SDHC卡插入主板
2. 在工具栏中选择Boot Device: sd_card/SDHC SD card 8 GB。
3. 对于主板，必须将SDHC电源重启设置为“启用”。

从 **eMMC** 启动：

要从 eMMC 启用，请执行以下操作：

1. eMMC 可以安装在主板上，也可以通过 SD/eMMC 适配器的 SD 插槽连接 eMMC。
2. 在工具栏中选择 Boot memory: eMMC/SDHC eMMC 8 GB。

6.14.3 Booting images for RT5xx/6xx devices

本章描述了明文和签名的 bootable images 的生成和写入过程。

6.14.3.1 Booting a plain/plain with CRC image

明文（plain）image 通常用于开发。建议在使用签名 image 前先使用未签名 image 验证是否能 正常工作。

要生成 bootable image 文件，请执行以下步骤：

1. 在工具栏上，在**Boot type**中选择**Plain unsigned**或**Plain with CRC**。
2. 切换到 **Build image** 视图。
3. 选择 Preparing source image RT5xx/6xx devices 中生成的 image 为 **Source executable image**。如果需要，打开 **Dual image boot** 并进行配置。如果已配置，请打开 **OTP configuration**并查看所有报告的问题。*BOOT_CFG[3]* 会在写入后被锁定，它只能被编程一次，因此写入时必须整体修改。
4. 单击 **Build image**按钮以生成一个 bootable image。

bootable image 生成成功后，请执行以下操作：

1. 有关如何连接开发板，请参阅Connecting the board RT5xx/6xx devices。
2. 切换到 **Write image** 视图。
3. 复位开发板。如果在没有复位开发板的情况下执行了两次脚本写入，则外部存储器的配置可能会失败，除非 fuse 中 QSPI 复位管脚的位域没有被烧写。
4. 单击 **Write image**。

如果写操作成功，将启动模式切换到 **FlexSPI boot**（参阅Connecting the board RT5xx/6xx devices中的**RT5xx/6xx EVK** 板开发配置）并复位开发板。

6.14.3.2 Booting plain signed image using shadow registers

本章节描述了签名认证 image 的生成和写入过程。

1. 在工具栏中，设置**Boot type**为**Plain signed**。

2. 在 **Build image** 视图中，使用 Preparing source image RT5xx/6xx devices 中生成的 image 作为 **Source executable image**。
3. 请确保您在 **PKI management** 视图中生成了密钥。更多详情，请参阅 PKI management。
4. 对于 **Authentication key**，选择任意密钥，例如 *ROT1: IMG1_1*。
5. 对于 **Key source** ，选择 **OTP**或者**KeyStore**。KeyStore 代表更高的安全级别，因为它使用了 PUF。对于限制条件，请参阅Securing the processor。
6. 生成随机 **User key** 和 **SBKEK**。
7. 如果需要，打开 **Dual image boot** 并进行配置。
8. 选择 **Development** 生命周期。
9. 打开 **OTP configuration** ，并查看所有报告的问题。fuses*BOOT_CFG[0]#BOOT_CFG[3]* 会在写入后被锁定，有必要指定整个值，因为它只会被编程一次。
10. 单击 **Build image**并检查是否成功地生成了 bootable image。

下面对如何烧写 image 进行了详细描述：

1. 进入 **Write image** 视图以烧写 image。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 Connecting the board RT5xx/6xx devices。
3. 复位开发板。这是必要的，因为：
   - 影子寄存器不能被更新或写入两次，它们只能在一个“干净的”处理器上被设置。
   - 如果 Fuses 中 QSPI 复位管脚的位域没有被烧写，则需要在每次配置前手动复位 flash。
4. 单击 **Write image**。

在写操作期间，执行以下步骤：

1. 检查 Fuses 以确保开发板处于非安全模式。
2. 一个简单的应用程序被写入 RAM。应用程序初始化影子寄存器。
3. 影子寄存器数据被写入 RAM。application 开始运行。
4. application 复位处理器。
5. 启动 write_image 脚本以配置外部flash并将应用程序写入flash。

6.14.3.3 Booting OTFAD encrypted image using shadow registers

本章节介绍加密 image 文件的生成和写入过程（OTFAD 加密）。

1. 在工具栏 中，将 Boot Type 设置为 **Encrypted (OTFAD) with CRC** 或者 **Encrypted (OTFAD) signed**。
2. 在 **Build image** 视图中，使用 Preparing source image RT5xx/6xx devices 中生成的 image 作为 **Source executable image**。
3. 请确保您在 **PKI management** 视图中生成了密钥。更多详情，请参阅 PKI management。
4. 对于 **Authentication key**，选择任意密钥，例如 ROT1: IMG1_1。
5. 对于 **Key source** ，选择 **OTP**或者**KeyStore**。KeyStore 代表更高的安全级别，因为它使用了 PUF。对于限制条件，请参阅Securing the processor。
6. 生成随机 **User key** 和 **SBKEK**。
7. 打开 **OTFAD encryption** 并设置随机密钥。
8. 如果需要，请打开 **Dual image boot** 并设置随机密钥。
9. 选择 **Development**生命周期。
10. 打开 **OTP configuration** ，并查看所有报告的问题。fuses*BOOT_CFG[0]#BOOT_CFG[3]* 会在写入后被锁定，有必要指定整个值，因为它只会被编程一次。
11. 单击 **Build image**并检查是否成功地生成了 bootable image。

下面对如何烧写 image 进行了详细描述：

1. 进入 **Write image** 视图以烧写 image。
2. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅Connecting the board RT5xx/6xx devices。
3. 复位开发板。这是必要的，因为：
   - 影子寄存器不能被更新或写入两次，它们只能在一个“干净的”处理器上被设置。
   - 如果 Fuses 中 QSPI 复位管脚的位域没有被烧写，则需要在每次配置前手动复位 flash。
4. 单击 **Write image**。

在写操作期间，执行以下步骤：

1. 检查 Fuses 以确保开发板处于非安全模式。
2. 一个简单的应用程序被写入 RAM。应用程序初始化影子寄存器。
3. 影子寄存器数据被写入 RAM。application 开始运行。
4. application 复位处理器。
5. SB 文件被应用到处理器上，过程中会执行以下动作：
   - 配置外部flash
   - 擦除 flash（如果 KeyStore被使用，则可以保留该区域）
   - 在 flash 起始处创建 FCB 块
   - 写入加密 application
   - 写入 OTFAD key blobs

注意： 如果开发板未复位，则对 QSPI flash 的重复写入可能会失败。

6.14.3.4 Booting signed/encrypted image - burn fuses

烧写 fuses 是不可逆的操作，只有在通过修改影子寄存器 （shadow register） 测试过 bootable image 之后才能够执行此操作。建议在烧写 fuses 之前安全备份所有密钥。启动过程与 Booting plain signed image using shadow registers 中描述的基本相同，不同之处在于写入操作 - 在写入时必须选择 Deployment 生命周期。在写操作过程中，影子寄存器不会被初始化，write image script 通过烧写 fuses 方式实现功能。GUI 将显示要烧写的 fuses 组列表的确认信息。

注意： 有关 fuses 修改的详细信息，请参阅 OTP/PFR/IFR/BCA/FCF configuration。

6.14.3.5 Securing the processor

要在 RT5xx/6xx 处理器上使能 full security，必须烧写 `DCFG_CC_SOCU` 和 `DCFG_CC_SOCU_AP` fuses。

默认情况下，SEC 不会设置去烧写这些 fuses 位域（即使选择了 Deployment 生命周期），但可以在 **OTP/PFR configuration** 中对其进行配置。

一旦 `DCFG_CC_SOCU` fuse 位被设置后，就不能再修改安全配置参数，并且不允许使用 blhost 更改 key store。如果使用 KeyStore，则只有在以下情况下才能在 SEC 中更新 image：

- ISP 模式仍处于使能状态，且
- *BOOT_CFG[1]* fuses`QSPI_ISP_AUTO_PROBE_EN`位被使能。
  RT6xx 处理器不支持此功能特性，并且只能使用自定义启动加载程序（custom bootloader）更新 image。

如果密钥存储在 OTP fuses 中，则没有任何应用限制。

RT600 上的 OTFAD 加密不支持 KeyStore，因为在擦除 flash 期间，SB 文件不支持备份和恢复 KeyStore。

要测试影子寄存器（shadow registers）中的 `DCFG_CC_SOCU` 配置，必须遵循以下步骤：

- 在没有 DCFG_CC_SOCU 的情况下准备并测试 image（例如，所有 DCFG_CC_SOCU必须为零）。
- 一旦 image 开始运行，FCB 配置有效且在 FLASH 中可用，请配置 DCFG_CC_SOCU。处理器现在被设置为 full security，因此某些 blhost 操作不再可用。处理器只能通过 SB 文件进行更新。

6.14.3.6 Device HSM provisioning

本节描述使用设备 HSM 加密的密钥 blob 来配置处理器，它可以将密钥（fuses 值）和 application bootable image 安全地传输到工厂。所有安全启动类型都支持设备 HSM：在**deployment**生命周期中选择 Plain signed 和 OTFAD encrypted 类型。可以从信任配置类型选择对话框的工具栏中选择设备 HSM。

在设备 HSM 模式下，一些 fuses 由 trust provisioning firmware 配置。这些 fuses 必须被配置，如果未指定所需的值，将显示错误（errors）。使用评估板将 fuses 加密到密钥 blob 中。在构建（build）操作期间，fuses values 被写入 RAM，然后 trust provisioning firmware 会创建密钥 blob。

要进行构建过程，请执行以下步骤：

1. 在工具栏上，确认选择了 **Plain signed**或者 **Encrypted** 启动类型。
2. 在工具栏上，确认选择了 **Deployment** 生命周期。
3. 在工具栏上，选择 **Device HSM** 配置类型（provisioning type）。
4. 在 OTP configuration 中, 必须指定设备 HSM 的所有 fuses；可以在应用程序 SB 文件中烧写 additional fuses。
5. 必须使用 UART 或者 USB 连接器（connector）连接 EVK 板。
6. 打开 **Connection** 对话框并测试连接。
7. 执行构建操作。在构建期间，创建 bootable image 和 SB 文件，将 fuses 值写入处理器 RAM，然后使用配置固件进行加密。此步骤不影响处理器的 fuse 值。
8. 在构建脚本之后，使用 JLink 调试探测读取加密密钥 blob。

写入

写操作与安全启动中使用的写脚本相同。都使用了同样的认证固件来解密 key blob，在烧写 fuses 之后复位芯片。它使用 SB文件来安装 application image。如果启动设备由 fuse 选择，请确保启动设备为空，这样处理器复位后会进入 ISP 模式，可以烧录 application image。

量产包

对于 manufacturing 操作，建议创建一个 Manufacturing package。更多详情，请参阅 Manufacturing workflow。

6.15 RT7xx device workflow

本节详细介绍 i.MX RT7xx 设备工作流程。

6.15.1 Preparing source image for RT7xx devices

在此步骤中，必须选择将要执行 image 的目标存储器。

支持的启动存储器为 NOR flash（实例 0 或 1）或 eMMC。

以下启动类型适用于 RT7xx 处理器：

- Image 运行在外部 flash：XIP image（就地执行）
- image 在内部 RAM 中运行ROM 在启动时将 image 从 FLASH 或 eMMC 复制到 RAM。
- Image 运行在外部 RAM (PSRAM) 中。ROM 在启动时初始化外部 RAM（参见 XMCD），并将 image 从 FLASH 或 eMMC 复制到外部 RAM。

6.15.2 Connecting the board for RT7xx devices

本节包含有关配置 MIMXRT700-EVK 评估板并将其连接到 SEC 的信息：

RT7xx 板启动配置

板	ISP模式	XSPI0	XSPI1	eMMC	PSRAM
MIMXRT700-EVK	SW10: 1-4 关 2-3 开	SW10: 1-4 开 2-3 关	SW10: 1-4 开 2-3 开	SW10: 1-4 关 2-3 关	JP45 至 1-2

请按照以下步骤来连接开发板：

1. 将开发板切换至 ISP 模式并复位（SW2）。更多详情，请参阅 Connecting the board for RT7xx devices中的**RT7xx** 板启动类型。
2. 使用 USB 电缆将您的 PC 连接到 **U7** 端口。
3. 请确保您已使用 workspace 启动 SEC。更多详情，请参阅Setting up Secure Provisioning Tool。
4. 将连接设置为**UART**并测试开发板连接。
   注意： 目前尚不支持USB接口。

6.15.3 Booting images for RT7xx devices

本章描述了明文和签名的 bootable images 的生成和写入过程。

默认情况下，workspace 配置为从外部 flash 启动。 要从 eMMC 启动，在 **main menu > Target > Boot memory** 中选择 **eMMC**。

位于 XSPI0 flash 或 eMMC 中的 image 可以在启动期间复制到 XSPI1 外部 RAM (PSRAM) 中并从那里执行。必须使用 XMCD 配置来初始化外部 RAM。对于 EVK 板，XMCD 配置在 SEC 工具中分发。

RT7xx 启动镜像所用内存的组合

执行 **image** 的内存	写入**image**的内存	需要的 **XMCD**	**XIP**
外部 NOR flash，XSPI0	外部 NOR flash，XSPI0	无	有
外部 NOR flash，XSPI1	外部 NOR flash，XSPI1	无	有
内部 RAM	XSPI0，XSPI1 或 eMMC	无	无
外部 RAM, XSPI1	XSPI0 或 eMMC	有	无

6.15.3.1 Life cycle for RT7xx devices

支持两种类型的生命周期：**shadows** 和 **OTP**。在 **shadows**中，OTP fuse 不会被烧毁，而是使用影子寄存器。影子寄存器通过调试探针初始化，因此在写入期间调试探针必须与 ISP 通信电缆连接。影子寄存器只能用于开发，不能用于生产。

6.15.3.2 Booting a plain unsigned/CRC image

明文（plain）image 通常用于开发。建议在使用签名 image 前先使用未签名 image 验证是否能 正常工作。

要生成 bootable image 文件，请执行以下步骤：

1. 在工具栏上，选择**Plain unsigned**或**Plain with CRC**作为**Boot type**。
2. 在工具栏中，将生命周期设置为 **Development, OTP**。Shadow registers（影子寄存器）将在稍后使用。
3. 切换到 **Build image** 视图。

4. 选择在 Preparing source image for RT7xx devices 中生成的 image 作为**Source executable image**。如果需要，打开 **Dual image boot** 并进行配置。如果已配置，请打开 **OTP configuration**并查看所有报告的问题。
5. 单击 **Build image**按钮以生成一个 bootable image。

当 bootable image 构建后：

1. 有关如何连接开发板，请参阅Connecting the board for RT7xx devices。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作成功，请切换启动模式并复位开发板。

如果写操作成功，将启动模式切换到 **XSPI boot** （ 参阅 Connecting the board for RT7xx devices中的**RT7xx EVK** 开发配置 \）并复位开发板。

6.15.3.3 Booting encrypted unsigned or CRC image

支持对从外部 Flash 运行的镜像进行加密。要配置明文镜像或 CRC 镜像的加密，请按照以下步骤操作：

1. 切换 **Boot type** 为 **Encrypted (IPED) unsigned** 或 **Encrypted (IPED) CRC**
2. 在 Build image 视图中打开**IPED regions** 配置并查看设置。默认情况下，整个镜像会自动加密。使用 CTR 或 GCM 加密模式。
3. 单击 **Build image**以生成一个 bootable image。注意： 镜像尚未加密，将在写入闪存时进行加密。

当可启动镜像构建成功后，将其以与上一章镜像相同的方式写入开发板。

如果使用双启动，建议手动为两个区域设置相同的加密方式。

6.15.3.4 Using RT7xx shadow registers

下一步，建议通过调试探针尝试影子注册表。

1. 使用上一步的配置。
2. 在工具栏中，将生命周期设置为 **Development, shadows**。
3. 选择 **main menu > Target > Debug Probe** 并选择PyOCD调试探针。调试探针与 UART 通过同一根电缆连接（两者并行通信）。
4. 单击 **Build image**以生成一个 bootable image。

当可启动镜像构建成功后，将其以与上一章镜像相同的方式写入开发板。

在写入脚本中使用shadow registers（影子寄存器）时，应用的高级操作如下：

步骤号	未签名和 **CRC** 启动类型	签名启动类型
第1步：	Flash配置	Flash配置
第2步：	将应用程序写入 flash	应用设备 HSM SB 文件来设置 CUST-MK-SK 影子寄存器
第3步：	应用影子寄存器并重置芯片	应用影子寄存器并重置芯片
第4步：		应用 SB 文件以写入应用程序 image

6.15.3.5 Booting signed image using shadow registers

本章节描述了签名认证 image 的生成和写入过程。

签名 image 不能与 **Development, OTP** 生命周期一起使用，因为 ROM 不包含任何在不推进生命周期的情况下刻录 CUST-MK-SK 的命令。

要构建签名 image，请使用以下步骤：

1. 在工具栏中，设置**Boot type**为**Plain signed**或**Encrypted (IPED) signed**。
2. 在工具栏上，将生命周期设置为 **Development, shadows**，**InField, shadows** 或 **InField-Locked, shadows**。
3. 在 **Build image** 视图上，使用 Preparing source image for RT7xx devices 中生成的 image 作为**Source executable image**。
4. 请确保您在 **PKI management** 视图中生成了密钥。更多详情，请参阅 PKI management。
5. 对于 **Authentication key**，选择任意密钥，例如 *ROT1: IMG1_1*。
6. 生成随机 **CUST-MK-SK** 和 **OEM**种子。
7. 如果需要，打开 **Dual image boot** 并进行配置。
8. 打开 **OTP configuration** ，并查看所有报告的问题。
9. 单击 **Build image**并检查是否成功地生成了 bootable image。在构建过程中，该工具与处理器通信以创建设备 HSM SB 文件。这不会导致处理器或 flash 发生任何变化。

下面对如何烧写 image 进行了详细描述：

1. 切换到 **Write image** 界面。
2. 确保调试探针仍处于选中状态（参见工具栏）。
3. 确保该开发板设置为串行引导加载程序 (ISP) 模式。更多详情，请参阅 RT7xx device workflow。
4. 单击 **Write image**。
5. 如果写入操作成功，请切换启动模式并对电路板进行软复位（按钮 SW2）。

6.15.3.6 Device HSM

下表展示了已签名启动类型中阴影（shadows）/fuse 的配置方式：

Fuse	任何阴影的生命周期	InField, OTP 生命周期	开发，OTP生命周期
CUST-MK_SK	设备 HSM SB 文件用于初始化影子寄存器。	设备 HSM SB 文件	不支持
其余 fuse	OTP 影子寄存器通过调试探针进行初始化。	设备 HSM SB 文件	不支持

6.15.3.7 Pre-erase in write script

为了开发目的，写入脚本支持预擦除参数。它允许出于以下原因擦除闪存：

- 移除任何先前的 IPED 配置
- 移除bootable image，以便处理器在重置后进入 ISP 模式（用于在配置过程中需要重置的情况）。

预擦除参数仅在 GUI 开发中使用，在制造过程中不使用，因为制造过程中需要使用空的flash。

6.16 RW61x device workflow

本章介绍 RW61x 处理器的工作流程。

6.16.1 Preparing source image for RW61x devices

RW61x 处理器唯一可用的启动存储器是外部 flash。

可选步骤：要为 SEC 工具中的构建创建源 image，请禁用启动标头，将定义符号BOOT_HEADER_ENABLE设置为0（在 MCUXpresso IDE 中，转至 **Project > Properties > C/C++ Build > Settings > MCU C Compiler > Preprocessor > Defined symbols** 并设置 BOOT_HEADER_ENABLE 为 0)。从 SEC Tool v9 开始，此步骤是可选的。SEC 工具可以解析bootable image，并从bootable image中检索应用程序image。

image 应位于地址 0x8001000（MCUXpresso SDK 示例中的默认值）。

6.16.2 Connecting the board for RW61x devices

本节将介绍如何配置 RD-RW612-BGA 或 RD-RW61x-QFP 或 FRDM-RW612 评估板并将其连接到 SEC。

1. 选择 ISP 启动模式，请参阅下表：**RW61x boards**板的启动配置。
2. 使用 USB 电缆将 J7 端口（在 FRDM 上为 J10）连接到您的 PC。
3. 确保在为 SEC 创建的工作区的情况下运行设备。更多详情，请参阅设置安全配置工具。
4. 进入 **main menu > Target > Connection** 并选择 UART 并测试连接。

6.16.2.1 Table: Boot configuration of RW61x boards

板	In-System Programming (ISP)启动	从外部FLASH启动
RD-RW612-BGA	U38 开关：0001	U38 开关：0000
RD-RW612-QFP	U38 开关：0001	U38 开关：0000
FRDM-RW612	重置期间按住 **SW3** 按钮。	无需 **ISP** 按钮即可重置。

6.16.3 Shadow registers for fuses via debug probe

SEC 工具使用 **In-Field, shadow registers** 作为默认生命周期。这意味着将使用影子寄存器配置**In-Field**生命周期。开发过程中，从影子寄存器开始以避免 fuse 发生不可逆的变化，但这不应该用于生产。影子寄存器配置是通过调试探测器完成的，因此必须首先选择探测器：

1. 进入 **main menu > Target > Debug Probe**，打开调试探测器选择对话框。
2. 从下拉菜单中选择调试探测器
3. 将板开关切换到从外部 FLASH 启动模式并重置处理器；单击“Test connection”以检查与调试探测器的连接

处理器不允许在“Develop”生命周期中使用影子寄存器，因此默认情况下，该生命周期的影子寄存器将配置“In-Field”状态。

另外，要使用影子寄存器进行 image 引导，需要在 BOOT_CFG0 影子寄存器中配置引导源。由于 fuse 在第一次写入后被锁定，该工具将要求您指定所有其他位（即使对于影子寄存器）。转至 **OTP configuration** 来指定它们。

在影子寄存器生命周期中， fuse 影子寄存器在写入 image 成功完成后立即配置。SEC 工具启动将设置影子寄存器的 write_shadows脚本，然后重置处理器。重置后，处理器启动 image。

对于加密模式，应用程序是通过 SB 文件写入的，因此必须对 RKTH 和 CUST_MK_SK fuses 进行编程（即使选择了 **shadow registers** 生命周期）。

6.16.4 Booting images for RW61x devices

本章描述了构建 bootable images 并将其写入外部 flash 以及启动。

默认的 SEC 工具配置是为 EVK 板准备的。如果您使用的是 FRDM 板，请使用 W25Q512 模板中的 FlexSPI 配置。如果 Flash 被锁定写入，可以使用 sample_data中提供的 FCB 将其解锁：

1. 选择 **main menu > Target > Boot Memory.**
2. 选择 FlexSPI NOR — 完整的 FCB。
3. 从磁盘中选择 FCB 文件。
4. 选择文件 `sample_data\targets\RW612\source_images\frdmrw612_wb_reset_fcb_quad_spi_v2.fcb`。该文件不得用于生产。

6.16.4.1 Booting plain or CRC image

明文（plain）image 通常用于开发。建议在使用加密类型 image 之前先使用此启动类型，以验证 executable image 是否正常工作。

首先，生成一个 bootable image：

1. 确保已在工具栏中选择了 **Plain unsigned**或**Plain with CRC** 启动类型。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 **Source executable image**。
4. 如果存在一个 binary image，请将起始地址设置为 0x8001000。
5. 配置 `BOOT_CFG0` fuse（见上一段）。
6. 如果需要，打开 **Dual image boot** 并进行配置。
7. 单击 **Build image**按钮以生成一个 bootable image。结果是二进制 bootable image。

成功生成 bootable image 后，请执行以下操作：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。
4. 如果写入操作执行成功，请复位开发板。
   - 如果选择影子寄存器，image 将自动启动
   - 或者，切换启动模式（请参阅 Connecting the board for RW61x devices 中的 **RW61x** 板的开发配置）并重置开发板以启动 image）。

6.16.4.2 Booting plain signed image

本章节介绍如何构建和编写明文签名图像。

生成 bootable image：

1. 在工具栏中，选择 **Plain signed**启动类型 。
2. 切换到 **Build image** 视图。
3. 选择在 Preparing source image for LPC55(S)0x/1x/2x/6x devices中生成的 image 作为 **Source executable image**。
4. 对于 **Authentication key**，选择任意密钥，例如 ROT1: IMG1_1。
5. 对“CUST_MK_SK”和“OEM seed”对称密钥，使用随机值。
6. 确保 OTP 配置没有错误。对于 ECC p384 密钥长度，您将需要配置 BOOT_CFG三个 fuse。
7. 确保主板已连接且处理器处于 ISP 模式。在构建过程中，准备配置 SB3 文件以安装 CUST_MK_SK 到处理器中。
   注意： SB 文件构建完成后，处理器会重置。
8. 保持 现场影子寄存器 生命周期，以避免处理器发生不可逆转的变化。
9. 单击 **Build image**按钮以生成一个 bootable image。结果生成一个二进制bootable image以及用于将该image安装到处理器的SB3封装。

成功构建 bootable image 和 SB3 封装后，可以上载到处理器：

1. 确保处理器处于ISP模式。
2. 切换到 **Write image** 视图。
3. 单击 **Write image**。

在影子寄存器的生命周期中，没有 fuse 被烧毁。签名的应用程序被写入 flash，然后应用影子寄存器，并重置处理器以启动应用程序。

一旦进入没有影子寄存器的生命周期，fuse 将不可逆地被烧毁，并且 SB3 capsule 将用于将应用程序写入 flash。

6.16.4.3 Booting encrypted image

支持带有 CRC 或签名 image 的加密 image。创建加密 image 的过程与签名 image 类似。

此外，还需要通过 **Build image** 视图上的 **IPED Regions**按钮配置加密区域（默认情况下，整个应用程序都是加密的）。IPED region 的对齐基于目标 FLASH 的页面大小，该页面大小可以从 FCB FLASH 配置中检索。有关如何将 Flex-SPI NOR 简化配置转换为完整的 FCB 配置，请参阅Boot memory configuration。

由于ROM的限制，镜像必须始终位于IPED区域内，不允许擦除或写入超出加密区域的范围。

结合双启动，仅对 image0 配置加密，对 image1 应用相同的设置。

当 image 被写入目标存储器时，执行 image 加密。应用程序是通过 SB 文件编写的，因此必须烧毁 RKTH 和 CUST_MK_SK fuses，以便处理器可以解密和验证 SB 文件内容。请注意，即使选择了影子寄存器，这些 fuses 也会被烧毁。

加密区域在 SB 文件中配置。解密的区域在 CMPA 页中配置，因此请确保这两个区域对齐。

6.16.4.4 Device HSM and provisioning

CUST_MK_SK 是用于 SB 文件加密/解密的客户密钥，并且该密钥只能使用设备 HSM SB 文件安装到处理器中。密钥存储在 fuse 中，因此安装是不可逆的，一旦 fuse 被写入，它就被锁定以进行写入（该锁也用于检测密钥是否已安装）。在写入脚本中，有一个参数来决定是否烧毁密钥。

要将设备 HSM 应用到处理器中，需要使用分布在受限数据中的设备 HSM 加载器固件（更多详情，请参阅Life cycle for RW61x devices）。

设备 HSM 的内容取决于生命周期，详情请参见下一章。

6.16.4.5 Boot type and life cycle for RW61x

下表概述了针对不同生命周期和启动类型配置的写入脚本所烧毁的 fuse，并包含有关 SB3 封装是否可用于更新应用程序 image 的信息。

	Shadow regs life cycle	Develop life cycle	In-Field life cycle
明文未签名或 **CRC** 启动类型	— 没有 fuse 被烧毁 — 没有 SB 文件被使用	— fuse 被烧毁，请参阅 OTP 配置对话框 — 没有 SB 文件被使用	— 与开发 + 生命周期 fuse 烧毁相同 — 没有 SB 文件被使用
明文已签名启动类型	— 没有 fuse 被烧毁 — SB 文件未使用，但它是在构建过程中生成的	— fuse 被烧毁，请参阅 OTP 配置对话框 — RKTH 和 CUST_MK_SK 被烧毁 — SB 文件被使用	— 与开发 + 生命周期 fuse 烧毁相同 — SB 文件被使用

	Shadow regs life cycle	Develop life cycle	In-Field life cycle
加密启动类型	— RKTH 和 CUST_MK_SK 被烧毁 — SB 文件被使用 — 没有其它 fuse 被烧毁	— fuse 被烧毁，请参阅 OTP 配置对话框 — IPED, RKTH, 和 CUST_MK_SK 被烧毁 — SB 文件被使用	— 与开发 + 生命周期 fuse 烧毁相同 — SB 文件被使用

6.16.4.6 Life cycle and trust provisioning

下表显示了不同生命周期和信任配置类型中安装的安全资产：

	设备 **HSM** 开发	现场设备 **HSM**	**EdgeLock 2GO** 开发	现场 **EdgeLock 2GO**
CUST MK SK	设备 HSM SB 文件	设备 HSM SB 文件	CUST-MK_SK	CUST-MK_SK
RKTH fuses	写入脚本	设备 HSM SB 文件	el2go_provi_otp.bin	el2go_provi_otp.bin
其余定制 **fuses**	写入脚本	设备 HSM SB 文件	el2go_provi_otp.bin	el2go_provi_otp.bin

6.16.4.7 EdgeLock 2GO

有关EdgeLock 2GO，请查阅EdgeLock 2GO trust provisioning workflow。以下是 RW61x 的具体注释：

- 如果处理器启动至 ISP 模式，则配置固件将不起作用。处理器必须处于运行模式，并且外部 flash 必须被擦除。重置处理器后，由于 flash 中没有应用程序，处理器将退回到 ISP 模式，并且可以启动 EdgeLock 2GO 配置。
- 安全对象的地址必须位于外部 flash 中。配置完成后，所有安全对象（甚至是已安装的安全对象）都保留在 flash 中。
- 在开发生命周期的情况下，配置仅在**DRY RUN**模式下工作，允许您在不影响处理器的情况下测试该过程。在此模式下，安全对象被应用到处理器（外部 flash）并执行配置固件，但固件只是验证安全对象，不会烧毁 fuse，也不会改变生命周期。尚未配置应用程序 image，因为尚无法加载 SB 文件。
- 要将 SB3 文件作为安全对象上传到 EdgeLock 2GO 服务器，请使用以下参数：
  - 创建新的安全对象
  - 二进制文件
  - 指定任意名称
  - 将对象标识 (OID) 设置为 0x7FFF817C
  - 非机密文件
  - 添加策略/选择设备生命周期：已关闭
  - 选择允许的算法：无
  - 此配置文件的政策：无 要将安全对象上传到 EdgeLock 2GO 服务器，请使用 **Closed**或**Closed/Locked**策略。所有安全对象必须采用相同的策略，否则配置将失败。**Open** 策略不是一个更好的选择，因为 **DRY RUN** 模式也支持 **Closed** 策略。

7 Generic workflows

本节提供一些典型用例的工作流程信息。这些章节并不针对任何处理器。

7.1 Debug authentication workflow

本节描述打开调试端口的过程。该工具提供调试验证协议（DAP），此协议作为现场技术人员验证调试器（外部实体）的机制。在授予设备调试访问权限之前，调试验证协议 （DAP）具有产品制造商（OEM）批准的凭据。

为了使调试验证 (DA) 正常工作，必须设置处理器特定的 fuse 或 PFR 字段。有关详细信息，请参阅设备用户手册的“调试子系统”一章。

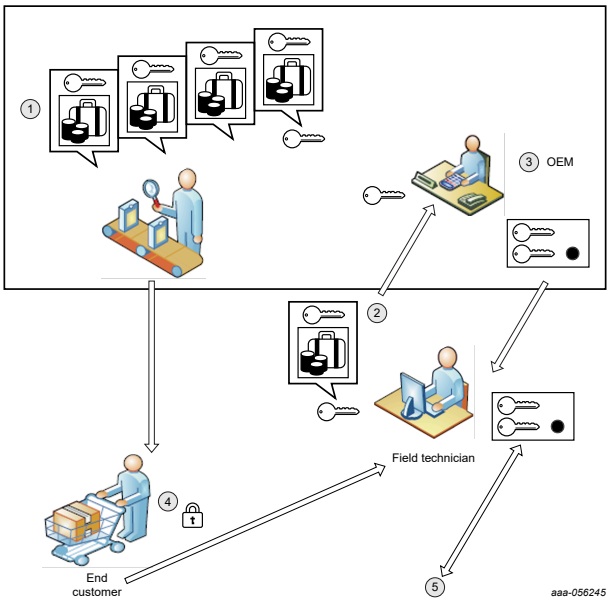

Figure 71. 调试身份验证协议使用示例

要打开调试端口，请执行以下操作：

现场技术人员（**Field Technician**）

1. 联系 OEM 以获取 ROT 的密钥类型和长度。OEM 通过指定 UUID 来决定是为具有相同 ROT 密钥的所有设备使用生成的证书，还是仅为一个设备使用生成的证书。
2. 在 SEC 工具中，为目标设备创建一个 workspace，并切换到 PKI Management 视图。
3. 点击 **Generate a debug key…** 按钮生成调试密钥，DA 密钥类型和长度必须与 ROT 密钥相同。
4. 点击 **Create debug certificate request…** 按钮。（可选）指定 UUID 以限制调试证书的使用。如果 UUID 设置为零，它可以用于任何设备。如果设备处于开发生命周期，则可以通过 UART 或 USB 从设备读取 UUID。对于处于生命周期后期阶段的处理器，调试探针有效。在大多数设备上，必须在 SOCU 寄存器/PFR 字段中进行CHECK_UUID设置，此选项才能起作用。
5. 发送证书请求（debug certificate）至 OEM。
6. OEM 发送证书后，选择 **Open debug port…**。在对话框显示时检测到已连接的探测器。检测到的探测器列表可以通过 **Find probes**进行更新。选择一个检测到的探测器。身份验证信标是一个可选参数，它独立于 OEM 提供的凭证信标。它不由调试身份验证协议解释， 而是传递给被调试的应用程序。确认对话框后，workspace\debug_auth\open_debug_port\[bat|sh] 中会生成一个脚本，并执行该脚本。如果脚本没有报告错误（操作成功），该对话框将关闭。 Troubleshooting 来寻求如何启用调试身份验证的有用提示。
   注意： nxpdebugmbox CLI工具可在`<installation_dir>/tools/spsdk/` 文件夹中找到。

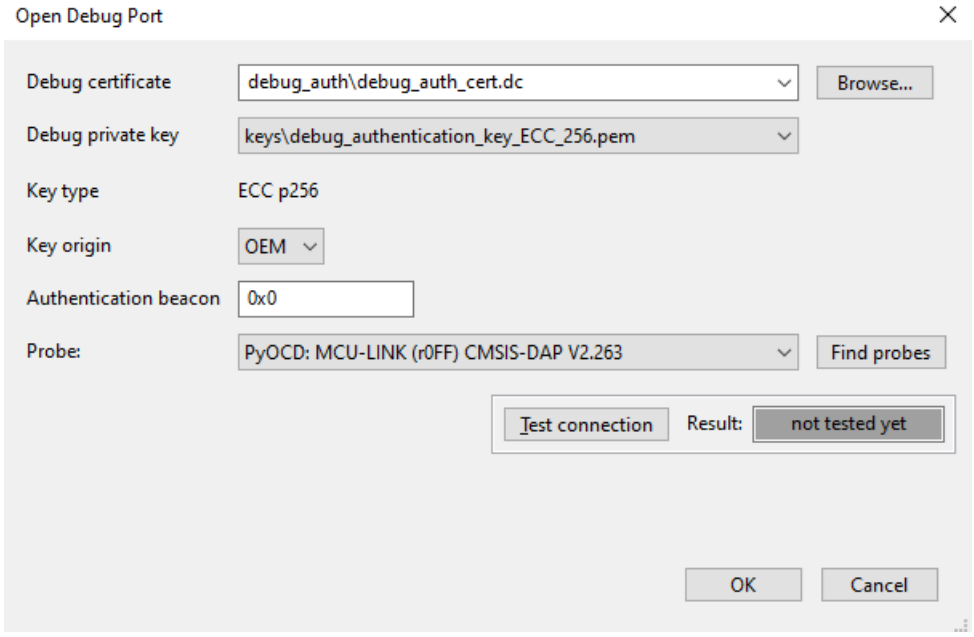

Figure 72. 打开调试端口对话框

OEM 开发商

收到请求后，点击 **Generate debug certificate**。

默认情况下，证书生成至 `<workspace>/debug_auth`文件夹， `debug_auth_cert.dc.`。创建一个同名的 *.zip 压缩文件，其中包含证书和 OEM 提供的`.txt`文件说明。现场技术人员传递给 OEM 的注释（note）显示在注释字段中（请参阅图 从证书请求生成调试证书）。

- **SoC** — `DCFG_CC_SOCU` 的掩码值控制使用身份验证协议访问哪些调试域。
- **Vendor usage** — 该字段可用于定义特定于供应商的调试策略。使用场景可以是调试凭证（Debug Credential，DC）证书吊销、部门标识符或模型标识符。
- 凭证信标（**Credential beacon**） — DAP 不会解释该值，它被传递给应用程序。该值独立于端口打开时现场技术人员提供的认证信标。
- **Note** — 一个字段，OEM 可以通过它描述关于生成证书原因。
- 使用**ROT**密钥签名 — 使用用于保护设备安全的ROT密钥之一对证书进行签名。

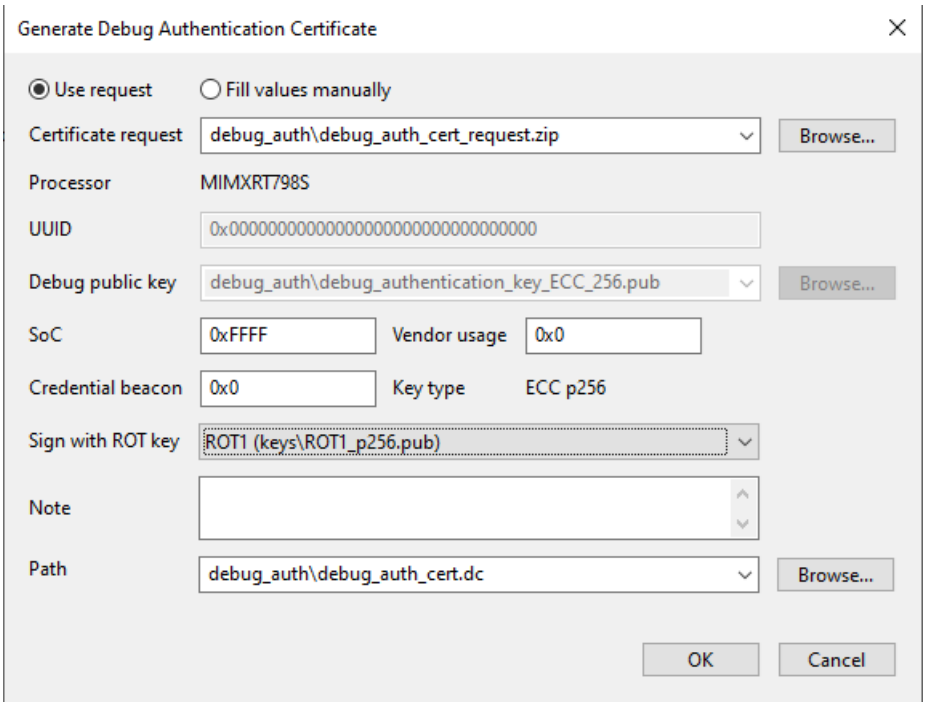

Figure 73. 从证书请求生成调试证书

7.1.1 Example of access rights to debug domains

示例用于测试目的。在最终使用之前，应重新访问和修改设置以满足安全要求。在下面的所有示例中，ISP 已启用，UUID 检查已禁用。对于某些处理器，必须设置 UUID 检查以启用调试探测器读取 UUID。

KW45xx/K32W1xx 和 MCX W71x

Fuse	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_L1, DCFG_CC_SOCU_L2	0x000000FF	0x0000FFFF	0x00004040
DBG_AUTH_DIS	0x0	0x0	0x0

KW47xx 和 MCX W72x

Fuse	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_L1, DCFG_CC_SOCU_L2	0x000001FF	0x0003FFFF	0x00008040
DBG_AUTH_DIS	0x0	0x0	0x0

LPC55S0x/1x，MCXW236 和 NHS52S04

PFR 域	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_NS_PIN、 DCFG_CC_SOCU_PIN	0xFF2000DF	0xFF2000DF	0xFFBF0040

PFR 域	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_NS_DFLT、DCFG_CC_SOCU_DFLT	0xFFFF0000	0xFF2000DF	0xFFBF0040

LPC55S2x

PFR 域	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_NS_PIN、DCFG_CC_SOCU_PIN	0xFF2000DF	0xFF2C00D3	0xFFBF0040
DCFG_CC_SOCU_NS_DFLT、DCFG_CC_SOCU_DFLT	0xFFFF0000	0xFF2C00D3	0xFFBF0040

LPC55S3x

PFR 域	一切都被禁用	一切都已启用*	由 DA 控制
DCFG_CC_SOCU_NS_PIN、DCFG_CC_SOCU_PIN	0xFE3001CF	0xFE3001CF	0xFFFF0000
DCFG_CC_SOCU_NS_DFLT、DCFG_CC_SOCU_DFLT	0xFFBF0040	0xFE3001CF	0xFFFF0000

注意： 对于调试，仍然需要身份验证，但该域无法通过 DAC 中的 SoC 掩码禁用。

LPC55S6x

PFR 域	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU_NS_PIN、DCFG_CC_SOCU_PIN	0xFD0002FF	0xFD0002FF	0xFFBF0040
DCFG_CC_SOCU_NS_DFLT、DCFG_CC_SOCU_DFLT	0xFFBF0040	0xFD0002FF	0xFFBF0040

MCX Nx4x 和 MCX N23x

PFR 域	一切都被禁用	一切都已启用*	由 DA 控制
DCFG_CC_SOCU_NS_PIN、DCFG_CC_SOCU_PIN	0xF81007EF	0xF81007EF	0xFFBF0040
DCFG_CC_SOCU_NS_DFLT、DCFG_CC_SOCU_DFLT	0xFFBF0040	0xF81007EF	0xFFBF0040

注意： 对于调试，仍然需要身份验证，但该域无法通过 DAC 中的 SoC 掩码禁用。

RT5xx/6xx

Fuse	一切都被禁用	一切都已启用	由 DA 控制
DCFG_CC_SOCU、 DCFG_CC_SOCU_NS	0x80FF408D	0x80FFFF20	0x00404088
DCFG_CC_SOCU_AP	0x7F00BF72	0x7F0000DF	0xFFBFBF77

RW61x

Fuse	一切都被禁用	一切都已启用*	由 DA 控制
DCFG_CC_SOCU、 DCFG_CC_SOCU_NS	0x3FFA007E	0x3FFFFF14	0x1002000F
DCFG_CC_SOCU_AP	0xC005FF81	0xC00000EB	0xEFFDFFF0

注意： 对于调试，仍然需要身份验证，但该域无法通过 DAC 中的 SoC 掩码禁用。

RT 118x 没有任何 fuse 来控制调试权限。调试取决于 LC，为 OEM_OPEN：允许所有调试，OEM_CLOSE：全部关闭但可由 DAC 启用，以及 OEM_LOCKED：全部关闭且无法启用。OEM_CLOSE 中管理调试权限的唯一方法是在 DAC 中设置 SoC。有关 SoC 掩码的示例，请参阅设备用户手册。

7.2 Signature provider workflow

本节介绍设置签名提供程序以及构建由签名提供程序签名的 image 的过程。签名提供程序服务器的示例位于 `<install_folder>/sample_data/signature_provider_examples` 中，其中一个使用 ROT ECC 密钥，另一个使用 ROT RSA 密钥。这些示例演示了 API 的完整实现，但在现实世界中，预计实现将通过与 HW HSM 模块通信或与另一台服务器的自定义 HTTPS 通信来更改。服务器的两个示例都可以按原样使用，以测试使用签名提供程序时的工具行为。每个服务器都有示例私钥并为 `public_keys_certs` 端点准备公钥响应。准备好的 ECC/RSA 公共树有 4 个 ROT 密钥/证书，每个 ROT 密钥/证书有一个 IMG 密钥/证书。这些密钥不应在最终产品中使用。

下图显示签名提供者的变体，SEC 工具将请求发送到自定义签名提供者 HTTP 服务器。此服务器应将请求传递给其中一个安全解决方案，然后将响应传递回 SEC 工具。准备好的示例仅实现自定义签名提供程序 HTTP 服务器。示例服务器正在执行应由 HSM 或外部签名提供程序完成的所有操作。由用户来实施完整的解决方案。

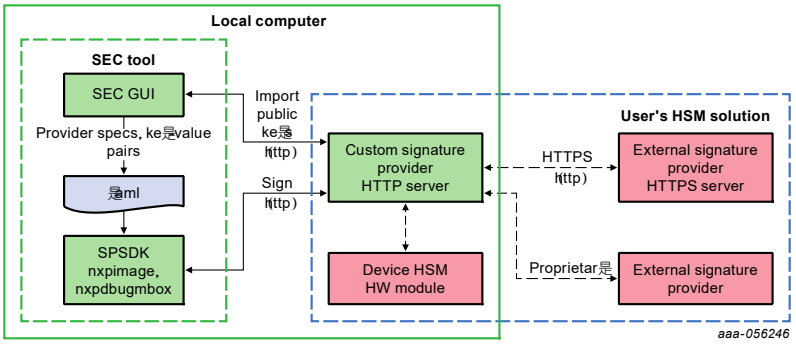

Figure 74. 签名提供程序的预期结构

7.2.1 Run the server

确保已安装 python 3.12 或更高版本。打开服务器实现所在的目录，并指定`requirements.txt`使用的 SPSDK 软件包版本（使用与**main menu > Help > About**中列出的版本相同的版本），然后运行以下命令：

```
# ensure, venv is installed
```

```
pip install virtualenv
# create virtual environment into .venv subfolder
python -m venv ".venv"
# activate it
.venv\Scripts\activate
# install all required packages into venv
pip install -r requirements.txt
# start the server
python server.py
```

服务器记录每个操作，因此可以查看执行了哪些操作。

7.2.2 Set up in the SEC Tool

要将签名提供程序示例与 SEC 工具结合使用，请按照以下步骤操作：

1. 为处理器创建/使用工作区。
2. 选中 PKI 选项卡上的 **Use sign. provider**复选框。 如果工作区中有密钥，它们会被移动到工作区中的备份子文件夹中。
3. 单击步骤 2 中的复选框旁边的 **Configure...**，打开签名提供程序对话框。
4. 检查您的签名提供程序的默认参数。如果使用 `resources\signature_provider_examples`中的签名提供程序服务器，则设置可以保留原样。
5. 在参数表中选择 prehash 选项。启用 prehash 后，仅将数据的哈希值发送到服务器进行签名。
6. 单击 **Test connection** 按钮以验证服务器是否配置正确。
7. 导入公钥有两个选项：
   - 在同一对话框中，单击 **Import public keys** 按钮以从服务器导入公钥；这是推荐的方式，但是只有当服务器实现可选 API **public_keys_certs** 时才可以使用。
   - 如果不支持 **Import public keys** 命令，另一种方法是使用 **PKI management**选项卡中的 **Import keys**。确保公钥与签名提供程序站点上使用的私钥匹配（将密钥复制到包含签名提供程序示例的文件夹中）。
8. 在 **Build**选项卡上照常选择密钥，现在 SB、MBI 和认证块的配置文件将使用签名提供程序配置。
9. 现在，签名提供程序已配置完毕。可以构建签名 image。

7.2.2.1 Signature provider and debug authentication

SEC 工具支持签名提供程序对 DA 证书进行签名。现场技术人员的 DA 密钥生成流程与没有签名提供程序的情况相同；调试密钥对的生成仅在本地受支持。唯一的区别是，在 OEM 站点上的 DAC 生成中，使用签名提供程序进行签名。这对用户来说是透明的，因为公钥是从 workspace 密钥中选择出来的。唯一的区别在于生成的配置文件，其中**sign_provider**字段是根据签名提供程序设置来设置的。

7.3 Script hooks workflow

脚本 hook 在生成、写入和制造脚本执行之前或期间执行。脚本 hook 允许在生成的脚本之外进行脚本自定义。Hook 脚本位于 “hooks” 子文件夹中，新工作区包含所选处理器的示例。脚本 hook 并非由该工具生成；它们应该完全由用户控制。如果 hook 脚本不存在，可以在图形用户界面中单击一下来创建它。

7.3.1 Build script hooks

1. Pre-build hook - 在构建脚本之前执行 `pre_build_<os-name>.bat/sh`。可以生成此脚本来对 MCUboot 应用程序 image 进行签名。此脚本没有任何参数，并且仅在执行构建脚本之前从 GUI 调用。
2. 构建上下文钩子：在构建脚本开始时调用`build_context_<os-name>.bat/sh`，允许用户添加或修改环境变量。这个hook 函数调用时不带任何参数。

3. 构建hook：build_.bat/sh 脚本在构建脚本的每个主要步骤之后都会执行。此脚本是从构建脚本中调用的，上一步的名称将作为参数传递。所有支持的步骤都在生成的示例中处理。如果某个 hook 步骤调用以失败告终，则构建脚本执行将停止并退出并显示错误。

7.3.2 Write script hooks

1. 构建上下文钩子：在构建脚本开始时调用`write_context_<os-name>.bat/sh`，允许用户添加或修改环境变量。这个hook 函数调用时不带任何参数。
2. 写 hook：在写脚本中的每个主要步骤之后调用 `write_<os-name>.bat/sh`。此脚本是从写脚本中调用的，上一步的名称将作为参数传递。所有支持的步骤都在生成的示例中处理。如果某个 hook 步骤调用以失败告终，则写脚本执行将停止并退出并显示错误。

7.3.3 Manufacturing hooks

制造 hook：在制造过程的开始和结束时，在第一个任务开始之前和最后一个任务完成之后调用 `manufacturing_<os-name>.bat/sh`。如果步骤 **started** 的制造钩子执行失败，则不会执行计划的制造步骤。对步骤 **finished** 的调用有一个附加参数 **status**，该参数可以有两个值 **ok**或 **fail**，表示制造过程的执行状态：**ok**表示所有任务都已成功完成，否则就是 **fail**。

7.3.4 Typical usage

以下是一些示例，说明如何使用 hook 脚本来自定义构建、编写或制造脚本，而无需修改从 SEC 工具生成的脚本：

- 更新构建的输入源文件。
- 修复写入脚本中的问题，或应用更新上一个操作的其他操作。
- 将制造操作与装配线同步。

7.4 Manufacturing workflow

对于制造操作，该工具提供了一个仅专注于制造的简化用户界面。

7.4.1 Create manufacturing package

OEM 可以创建可用于将文件传输到工厂的制造包 (*.zip)。可以创建制造包。

- 在 **Write image**视图上 - 用于使用当前写入脚本进行配置
- 在 **SB editor**中 - 应用 SB 文件中指定的命令

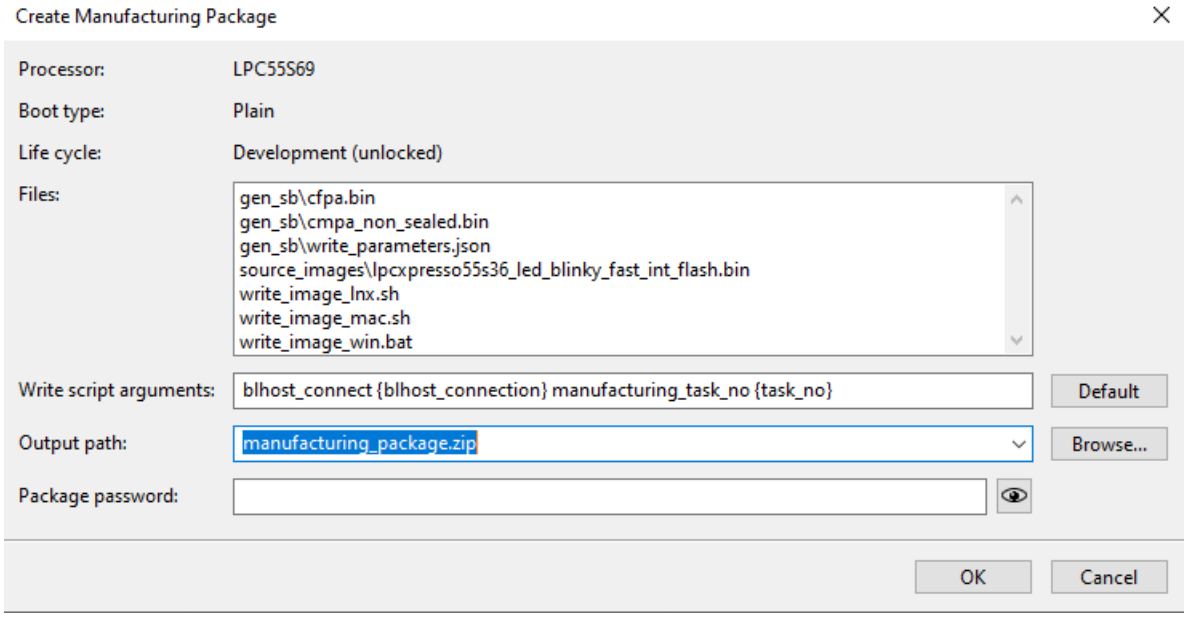

Figure 75. Create Manufacturing Package 对话框

Create Manufacturing Package 对话框允许以下操作：

- 查看 Manufacturing package 中包含的文件。
- 检查写入脚本参数，这些参数的格式与量产工具中使用的格式相同。
- 选择输出量产包文件路径。
- 设置 zip 文件的密码。使用 AES 算法对 zip 文件中的文件进行加密；但是，文件元数据（如文件名和文件大小）未加密。
- 对于 EdgeLock 2GO 配置，可以选择在制造包中指定 API 密钥，但也可以在稍后的制造过程中指定

7.4.2 Performance optimization

假设配置/写入脚本将被执行多次以解决不同的开发问题，因此该脚本主要针对此用例而设计，并且可能包含一些在供应应用于空处理器时在制造中不需要的部分。以下是通过手动修改生成的脚本来提高制造性能的一些技巧：

- 可能不需要擦除启动内存。如果处理器是新的，则闪存已被擦除。
- 可能不需要检测芯片是否已经受到保护；写入脚本中的这一部分允许更新安全处理器的应用程序（对于支持它的芯片）
- 较长的 **blhost**操作序列可能不是最佳的，因为 **blhost**初始化时间并非微不足道。将操作移至 SB 文件（请参阅 SB 编辑器）或使用 **blhost batch** 命令可能会更好（更多详情，请参阅 SPSDK 文档）。
- 如果处理器在制造期间重置，则在启动和操作系统驱动程序重新连接之前会有一段延迟时间。默认为 3 秒。时间可以根据喜好进行调整。

7.4.3 Manufacturing operations

在生产现场，可以使用 **main menu > File > Import Manufacturing Package...** 导入制造包。导入过程中，软件包中的文件会被提取到新的 workspace — 称为 **manufacturing workspace**。有关导入包的更多信息，请参阅从 ZIP 文件导入 workspace。

一旦 workspace 被创建（或重新打开），量产工具就会显示出来（更多详情，请参阅 Manufacturing Tool ），而 SEC 的其余功能则不可用。如果量产工具被关闭，则整个工具操作完成。如果您重新启动 SEC， 它会继续提供量产操作，或选择另一个 workspace：

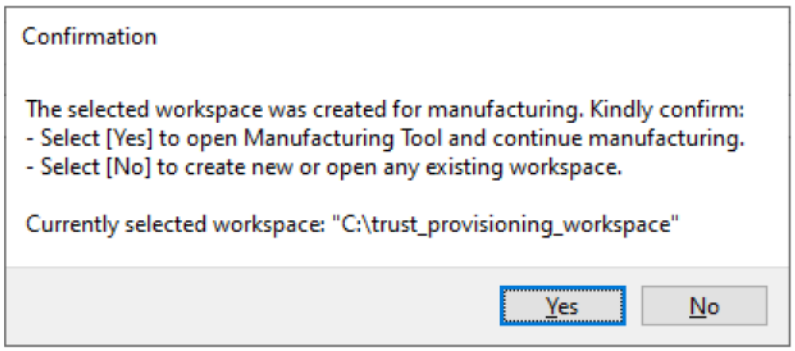

Figure 76. 确认重新打开制造

如果量产包中包含写入脚本，请检查脚本中的 `SPT_INSTALL_BIN` 环境变量是否指向计算机上的安装目录。如果没有指向，建议全局设置环境变量或手动更新写入脚本。

对于 **serial** 连接：调整波特率，默认值为115200；但是，对于几个处理器，已成功测试了高达 1000000 的波特率。

7.4.4 Steps in the manufacturing production

1. 通过 USB 或串行线或 I2C 或 SPI 连接一个或多个处理器，然后点击 Autodetect 按钮以检测串行端口名称。如果该工具检测到任何与处理器不匹配的串行设备，请禁用该设备。然后点击下方的 Test connection，检查与所有启用的处理器的连接，确保测试通过。
2. 点击 Start（开始）按钮，以启动信任配置操作。等待所有操作完成。
3. 如果报告任何问题，点击状态单元格显示日志并修复问题。
4. 继续步骤 1的操作。

成功配置的设备数量显示在 Manufacturing Tool 窗口的底部，但它不是 100% 可靠的。

7.4.5 Manufacturing logs

制造日志存储在制造工作空间的子目录 `logs/YYYY-MM-DD/`中。日志文件名为 `manufacturing_log_YYYY-MM-DD_hh-mm-ss_#.log`，其中 `#` 表示并行执行的制造任务的索引。可以使用制造窗口中的 **Export logs** 按钮导出它们。导出允许选择一天（日期）或所有日志，并将选定的日志导出到 ZIP 文件中。

7.5 MCUboot workflow

MCUboot 是一个用于 32 位微控制器的开源安全引导加载程序。更多详情，请参阅 MCUboot 文档。它已被移植到许多 NXP 处理器，并且您可以在 MCUXpresso SDK 中找到它的支持。更多详情，请参阅`boards\<board>\ota_examples\mcuboot*`。

对于大多数处理器，SEC 工具包含与 MCUXpresso SDK 中提供的项目示例兼容的默认配置。因此，有一个从 MCUXpresso SDK 为 EVK/FRDM 板构建的“示例应用程序”，默认的 imgtool 参数与 MCUXpresso SDK 匹配。

7.5.1 Steps to start MCUboot for NXP board via New Workspace

1. 打开 **New Workspace** 对话框并选择处理器。

2. 通过 UART 连接开发板，确保处理器启动处于 ISP 模式，可以使用其他连接，使用 UART 是因为 mcuboot 应用程序是使用调试控制台构建的，并打印可以使用终端验证的状态。
3. 选择 MCUboot 引导加载程序 image，系统将自动选择所选处理器的相应示例应用程序（包括引导加载程序和可执行 image）。如果没有选择任何 image，则表示所选处理器没有可用的示例应用程序。在这种情况下，请参阅以下手动配置说明。
4. 选择与示例应用程序兼容的内存设置的所需配置文件，并创建工作区。对于 MCX N 系列，必须使用 IFR。
5. MCUboot Sign Image 对话框将打开，所有必需的选项都已预先选中，因此可以保存选项并对 image 进行签名。
6. 构建 bootable image 并将其写入处理器。注意： 要验证一切是否正常工作，请参阅下面手动配置的步骤 9。

7.5.2 Steps to start MCUboot with such a processor manually

1. 为选定的处理器创建新的工作区。
2. 通过 UART 连接开发板，确保处理器启动处于 ISP 模式，然后检查连接（**main menu > Target > Connection**…）。可以使用其他连接，使用 UART 是因为 mcuboot 应用程序是使用调试控制台构建的，并打印可以使用终端验证的状态。
3. 在 **main menu > Target > Boot Memory …** 中验证所选的启动内存。对于 MCX N 系列，必须使用 IFR。
4. 打开**Build image**视图，选择“mcuboot opensource”应用程序作为 “Source executable image”。预构建的应用程序位于 `sample_data\targets\<processor>\source_images\<board>_mcuboot_opensource.s19`。您还可以从 MCUXpresso SDK 构建您自己的。如果应用程序包含外部闪存 (FCB) 的配置，该工具会检测并提供使用它。建议采纳。
5. 打开 **main menu > Tools > MCUboot > Sign Image** 并配置以下内容：
   - 需要签署的二次申请；预构建的应用程序位于 `sample_data\targets\<processor>\source_images\<board>_ota_mcuboot_basic.s19`
   - 签名密钥；它位于预构建应用程序或 MCUXpresso SDK 的同一文件夹中，位于 `boards\<board>\ota_examples\mcuboot_opensource\keys\` 文件夹中。根据平台可以选择 `sign-rsa2048-priv.pem` 或者 `sign-ecdsa-p256-priv.pem`。尚不支持受密码保护的密钥。
   - 默认情况下，imgtool 参数应与 SDK 示例匹配。不需要改变它们。
   - 点击 **Sign** 按钮对申请进行签名；解决问题（如果有）。
   - 确保选中以下复选框：
     - 第一个复选框：该工具创建预构建hook脚本，该脚本在每次构建之前对应用程序进行签名
     - 第二个复选框：该工具重新配置其他image，因此签名的image被写入目标地址；保留 “Image 1”
   - 验证应用程序的目标地址。
   - 单击 **Save & Close**关闭对话框。
6. 在 **Build** 选项卡上，仔细检查 **Build script hooks** 部分是否包含预构建脚本。
7. 打开 **Additional Images** 并检查签名的应用程序是否已正确配置为 “Image 1”.
8. 构建 bootable image 并将其写入处理器。
9. 打开终端（在 MCUXpresso IDE 中，转至 **main menu > Windows > Show View > Other > Terminal**），选择 UART 端口，确认并重置处理器。您应该收到以下文本：

```
hello sbl.
Bootloader Version 2.0.0
Primary   slot: version=1.0.0+0
Image 0 Secondary slot: Image not found
writing copy_done; fa_id=0 off=0xfffd0 (0xfffd0)
Image 0 loaded from the primary slot
Bootloader chainload address offset: 0x0
Reset_Handler address offset: 0x400
Jumping to the image
```

```
Booting the primary slot - flash remapping is disabled

***********************************
* Basic MCUBoot application example *
***********************************
Built Apr 16 YYYY 12:34:56
```

要签署并撰写申请的第二份副本，请执行以下附加步骤：

1. 如果终端仍然打开，请将其断开。将板重置为 ISP 模式。
2. 打开 **main menu > Tools > MCUboot > Sign Image...**。
3. 保持相同的输入应用程序 image。
4. 更改输出 image 名称，例如添加后缀 1_1
5. 在 imgtool 参数中，将版本更改为 1.1
6. 在 **Set additional image** 部分中更改：
   - Image 2
   - 目标地址 - 将地址增加槽大小（请参阅 imgtool 参数中的槽大小参数）
7. 保存和关闭

   注意： 预构建脚本将被覆盖，并对 Image 2 进行签名。之前签名的 Image 1 保持在磁盘上不变，并将被写入闪存的原始位置。
8. 打开 **Additional Images** 并检查是否有两个应用程序：“Image 1” 和 “Image 2”。
9. 构建并写入。
10. 连接终端，将板切换为从选定的启动存储器启动，然后重置。终端应该收到以下文本：

```
hello sbl.
Bootloader Version 2.0.0
Primary   slot: version=1.0.0+0
Secondary slot: version=1.1.0+0
writing copy_done; fa_id=1 off=0xfffd0 (0x1fffd0)
Image 0 loaded from the secondary slot
Bootloader chainload address offset: 0x100000
Reset_Handler address offset: 0x100400
Jumping to the image

Booting the secondary slot - flash remapping is enabled

***********************************
* Basic MCUBoot application example *
***********************************

Built Apr 16 YYYY 13:24:56
```

7.6 EdgeLock 2GO Trust Provisioning workflow

EdgeLock 2GO 是由 NXP 运营的完全托管云平台，提供安全配置服务，以便使用受支持的恩智浦产品轻松部署和维护物联网设备。该服务允许创建和管理安全对象，例如对称密钥、密钥对和证书，然后将它们安全地配置到您的 NXP MCU 或 MPU 中。

该工具支持以下流程：

- **EdgeLock 2GO with device ID**流程，制造工厂可在生产过程中从 EdgeLock 2GO 云服务器检索安全对象。
- **EdgeLock 2GO per product type**流程，其中 OEM 从 EdgeLock 2GO 服务器检索安全对象数据库，而制造工厂的生产无需在线访问 EdgeLock 2GO 服务器即可运行。

7.6.1 EdgeLock 2GO with device ID, high-level description

下图对应于图 EdgeLock 2GO 信任配置工作流程中所示的操作顺序。

配置流程

1. OEM 准备了一个安全的配置并在本地加以证明。然后，SEC工具提供所有安全对象，OEM通过EdgeLock 2GO web portal上传它们。
2. OEM厂商将由SEC工具生成的制造包发送到制造工厂。

EdgeLock 2GO 制造工艺及设备 **ID**

1. 该工具获取处理器 UUID
2. 该工具请求给定 UUID 的安全对象。EdgeLock 2GO 服务器验证 UUID、生成安全对象并使用 EdgeLock 2GO 信任根对其进行加密。安全对象被下载到制造机器。
3. 安全对象和 EdgeLock 2GO 配置固件已加载到目标处理器。调用固件以将安全对象安装到目标位置。应用程序 image 被应用到目标引导存储器中。

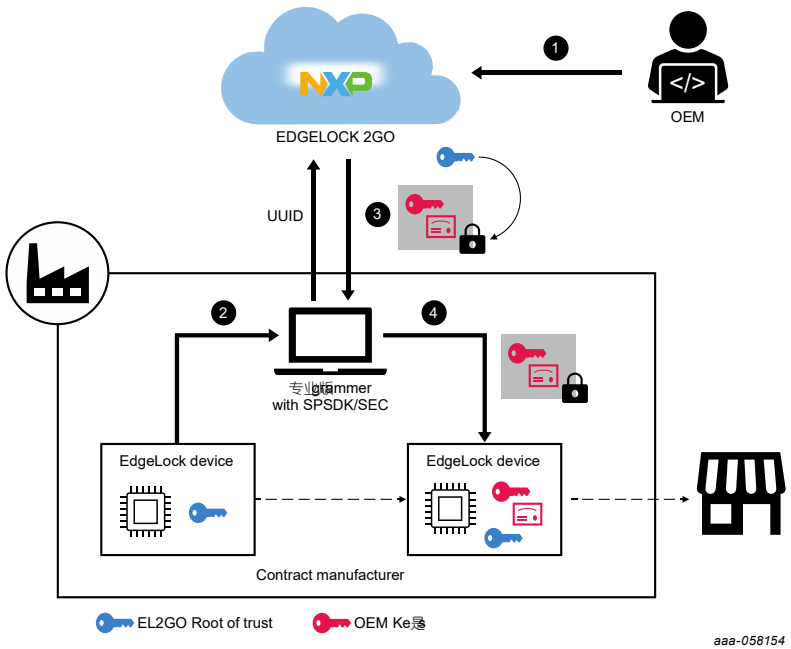

Figure 77. **EdgeLock 2GO 信任配置工作流程**

7.6.2 EdgeLock 2GO per product type, high-level description

配置流程

1. OEM 准备了一个安全的配置并在本地加以证明。然后，SEC工具提供所有安全对象，OEM通过EdgeLock 2GO web portal上传它们。
2. OEM厂商下载安全对象数据库，并将数据库和制造包提供给制造工厂。

EdgeLock 2GO 各产品类型的制造工艺

1. 该工具从数据库中获取安全对象。
2. 安全对象和 EdgeLock 2GO 配置固件已加载到目标处理器。调用固件以将安全对象安装到目标位置。应用程序 image 被应用到目标引导存储器中。

7.6.3 EdgeLock 2GO flow, step by step

要配置 EdgeLock 2GO 配置，请执行以下步骤：

1. 打开安全启动模式并验证应用程序是否正常工作。
2. 在 EdgeLock 2GO server上，为产品创建设备组并创建 API 密钥。更多详情，请参阅EdgeLock 2GO Configuration via web portal。
3. 点击 **main menu > Target > Trust Provisioning Type**，选择 **EdgeLock 2GO**。选择 **EdgeLock 2GO with device ID** 或 **EdgeLock 2GO per product type**。填写 EdgeLock2GO 参数并测试与服务器的连接。
4. 安装 GnuPG 工具并利用准备好的加密脚本。
   ##########:
   - Windows - 推荐用于 Windows 的 Kleopatra：Kleopatra for Windows (GPG4Win). 加密也可以使用“Kleopatra GUI”手动完成。更多详情，请参阅 *EdgeLock 2GO Quick Start Guide*（文档 AN13255（位于 EdgeLock 2GO server））或使用 上述方式（GnuPG 实用程序随 gpg4win 一起安装）。
   - Ubuntu：使用 Ubuntu 存储库中的 gpg 软件包。
   - macOS：推荐使用brew包系统中的gnupg；适用于 OS X 的 GnuPG 也应该可以工作（请参阅 GnuPG downloads）
5. 在构建脚本中启用 GPG 加密所需的步骤：
   - 加密是在工作区中生成的 `el2go/encrypt_el2go_files.*` 脚本中完成的。使用 **Build** 页面上的 **Update files** 按钮来更新脚本。
   - 导入 EdgeLock 2GO 公钥进行加密，请参阅生成的脚本了解如何操作
   - 生成用于签名的 RSA 密钥对 - **gpg -full-generate-key**。使用 2048 或 4096 密钥大小。
   - 设置环境变量
     – EL2GO_GPG_PASSWORD - 生成的密钥对的密码
     – EL2GO_GPG_SIGN_KEY - 使用 **gpg-list-secret-keys** 提供的密钥 ID 确定应使用哪个密钥进行签名。如果定义了 EL2GO_GPG_SIGN_KEY 环境变量，则会从构建脚本调用加密作为构建操作的一部分。
6. 确保在 **main menu > Target > Life Cycle** 中选择开发生命周期。出于开发目的，开放生命周期支持 EdgeLock 2GO 验证。这不适用于生产。
7. 在构建页面上，单击 **Build image** 来构建应用程序。确保没有错误。
8. 打开`<workspace>/el2go/publish`子文件夹并将所有安全资产上传到EdgeLock 2GO server。请按照 Configure EdgeLock 2GO through the portal中提供的说明进行操作。
   注意： EdgeLock 2GO 服务器的文件也显示在构建页面上。在生成的文件中，它们由 el2go 云图标标记。描述将显示在工具提示中。
9. 对于 EdgeLock 2GO 的每个产品流程，请参阅 Create database with secure objects for EdgeLock 2GO per product type。
10. 进入 **Write** 页面进行 Write 操作。
11. 如果一切按预期工作，请设置部署生命周期并重复步骤 7-9。
12. 创建制造包更多详情，请参阅Create manufacturing package。有关 API 密钥分配，请参阅 API key to access EdgeLock 2GO server。

有关处理器特定信息，请参阅 Processor-specific workflows。

7.6.4 API key to access EdgeLock 2GO server

- 有关开发，必须在 **Trust Provisioning Mode** 对话框中指定 API 密钥。
- 对于 **EdgeLock 2GO per product type**，生产过程中不使用API 密钥。
- 对于 **EdgeLock 2GO with device ID**，生产过程中还需要 API 密钥。
  – 对于制造包，可以选择性指定API key；如果指定，建议使用密码加密包以保护访问。

- 如果制造包中未指定 API 密钥，则可以在制造对话框中或通过环境变量 SEC_EL2GO_API_KEY 指定。
- 对于 EdgeLock 2GO 信任配置，需要在 EdgeLock 2GO 服务器上为制造 API 密钥启用以下访问：
  - View - 查看设备组
  - General - 注册设备
  - Remote trust provisioning - 查看安全对象及其中间 CA
- 对于 EdgeLock 2GO WPC 配置，需要在 EdgeLock 2GO 服务器上为制造 API 密钥启用以下访问：
  - View - 查看 Qi 相关数据
  - Manage - 将 Qi 模板分配和取消分配给设备组，以及查询批处理作业

7.6.5 EdgeLock 2GO Configuration via web portal

EdgeLock 2GO 门户网站 URL：EdgeLock 2GO web portal

更多详情，请参阅：EdgeLock 2GO Provisioning via Secure Provisioning Tool (SEC) for MCUs（文档 AN14624）。

如何创建设备组

- Devices > New Device Group
- 指定任意名称
- 硬件系列类型：MPU 和 MCU
- 硬件选项：产品名称
- 硬件类型和产品名称：按名称选择处理器

如何创建 **RKTH** 安全对象

- Secure Objects > New Secure Object
- 选择类型：OEM 固件身份验证密钥 Hash
- 指定任意名称
- 提供二进制文件：<upload the file>
- 使用策略：添加自定义策略。选择**Open** 或 **Closed/Locked** 策略。**Open** 用于开发和测试，**Closed** 和 **Closed/Locked** 用于生产。该策略应与工具中选择的生命周期保持一致。有关详情，请参阅特定于处理器的建议。

如何创建 **CUST-MK-SK** 安全对象

- Secure Objects > New Secure Object
- 选择类型: OEM FW Decryption Key
- 指定任意名称
- 提供密钥材料 (.asc)：提供签名和加密的密钥。它是由 SEC 工具通过 GnuPG 生成的。
- 提供公钥以使用它来验证加密密钥的签名。 (.asc)：您的公钥
- 使用政策：同RKTH；对于一个设备组中使用的所有安全对象，该策略必须相同。

其他安全对象

有关处理器特定信息，请参阅 Processor-specific workflows。

7.6.6 Create database with secure objects for EdgeLock 2GO per product type

数据库分为两种类型：

- 静态数据库：不包含任何处理器特定的安全对象
- 动态数据库：包含处理器特定的证书。

要创建用于“离线”配置的安全对象数据库（每个产品类型使用 EdgeLock 2GO），请按照以下步骤操作：

1. 打开 **main menu > Targets > Trust Provisioning Mode…** 并确保已选择 **EdgeLock 2GO per product type**。
2. 在**Per Product Database**面板上选定：
   - number of devices per database（每个数据库的设备数量）
   - number of databases to be generated（要生成的数据库数量）

   这些参数仅用于动态数据库。对于静态数据库，这些参数将被忽略。
3. 点击按钮开始生成。请注意，生成动态数据库可能需要相当长的时间（通常为几个小时），并且会显示一个进度条来指示预计时间。在数据库生成之前，请勿关闭窗口。
4. 点击 **Download** 按钮，从服务器下载数据库。

对于写入操作，数据库名称被硬编码为 `<workspace>/trust_provisioning/el2go_product_batch.db`。对于制造操作，可以在制造对话框中选择数据库名称和位置。

数据库不包含在制造包中，预计将单独交付。

7.7 Merge Images Tool workflow

本节描述了将两个独立image合并为一个单一image的特定用例。合并的原因可能是构建过程中需要将两个image文件一起签名。本节是为带有 Trust zone hello world SDK 示例的 MCUXpresso IDE 编写的，其中包含两个链接在一起的独立项目 `_hello_world_s`和`_hello_worlds_ns`。有必要删除项目之间的链接（MCUXpresso IDE），以生成两个独立的镜像，而不是在构建过程中通过将两个项目合并在一起创建的一个image。以 MIMXRT595 处理器为例。对于其他处理器，工作流程相同，只是起始地址不同。

7.7.1 Building the TrustZone example projects for merge

1. 通过选择非安全项目 `_hello_world_ns`转到**Project > Properties > Project References**，然后取消选中引用中的`_hello_world_s`来删除两个工程之间的链接。
2. 无需任何修改即可构建两个项目
3. 此示例使用.s19 文件，因此最终的.axf 文件需要转换。为此，请转到两个项目的 Debug 文件夹，右键单击生成的 `_hello_world_*.axf`文件，选择**Binary Utilities > Create S-Record**。

7.7.2 Merging two images into one

1. 通过选择**main menu > Tools > Merge Tool**来打开合并工具。
2. 选择 `_hello_world_s.s19`。目标地址将自动设置为 `0x18001000`（安全内存区域）
3. 对于 MIMXRT595 处理器，用户只能写入非安全地址。第一个image的给定地址位于安全内存映射中。 将第一个image的目标地址更改为 `0x08001000` （非安全内存区域）
4. 选择 `_hello_world_ns.s19`。目标地址将自动设置为 `0x08100000`（非安全内存区域）
5. 勾选 **Apply the merged images as the Source executable image on Build image view**
6. 其余部分保持不变，然后按“OK”按钮
7. 检查image是否自动应用为源可执行image，并且其起始地址也已更新

7.7.3 Signing merged image

要签署并构建合并的image，请参阅 Booting signed image using shadow registers，但使用合并image作为源可执行image。

7.8 Custom USB VID and PID devices workflow

如果您有具有自定义 VID 和 PID 的设备，则可以在 **main menu > Target > Connection** 选择自定义 VID 和 PID。选定的 VID&PID 用于##写入操作。写入脚本不支持在执行过程中更改 VID&PID。建议在写入操作之前更改 VID&PID（例如，在**Advanced mode**中使用 PFR 对话框或 OTP 对话框），重置处理器，然后使用自定义 VID&PID 启动写入操作。

对于制造操作，必须在创建制造包期间在连接配置中选择自定义 VID&PID。使用此配置，制造操作同时接受默认和自定义 VID&PID。

8 Command-line operations

SEC 工具还提供命令行操作界面，支持在自动化环境中集成或自定义 image 生成和烧写流程。生成 image、烧写 image、验证、生成密钥或检测 USB 设备等操作需要使用特定命令。

要显示可用的命令、参数和示例，请从命令提示符运行以下命令：

```
c:/nxp/SEC_Provi_25.09/bin/securep.exe -h
```

要显示特定命令的可用参数，请从命令提示符运行以下命令：

```
c:/nxp/SEC_Provi_25.09/bin/securep.exe <command> -h
```

注意： SEC 工具应用程序的位置取决于安装文件夹。

命令行中使用的所有支持的命令和参数 （如以下章节所述） 也可以在单独的配置 JSON 文件中指定。然后将此 JSON 文件作为命令行参数传递，请参阅 。 Example how to use args-file arguments.

args-file 参数

参数	描述
`--args-file ARGS_FILE`	带有 CLI 参数的 JSON 文件的路径，允许在一个文件中指定所有参数。该路径是绝对路径或相对于当前工作目录的路径。文件格式由指定 `schema/cli_args_file_schema_v?.json`.

8.1 Build

build 命令支持在 SEC 的 **Build image** 视图中执行的操作。

下面对 **build**指令参数进行介绍。

生成参数描述

参数	描述
`-h, --help`	显示帮助信息，然后退出。
`--additional-images-cfg`	JSON 的路径，其中包含用于构建的附加 image 像的配置。文件格式由模式指定 `schema/additional_images_schema_v?.json`.
`--bee-user-keys-config BEE_USER_KEYS_CONFIG.json`	带有 BEE 配置的 JSON 文件。参数文件的路径为 schema/bee_image_encryption_schema_v2.json。该参数仅适用于加密的 XIP（BEE 用户密钥）引导类 型。

参数	描述
--boot-type VALUE	安全启动类型。运行 securep -help 以查看所有支持的引导类型。
--cfpa-cfg CFPA_CFG.json	带有 USER CFPA 配置的 JSON 文件的路径。建议从 PFR 配置框中导出文件。
--cmpa-cfg CMPA_CFG.json	带有 USER CMPA 配置的 JSON 文件的路径。建议从 PFR 配置框中导出文件。
--csf-cert CSF_CERT	用于对 image 进行签名的公共 CSF 密钥文件的路径。如果未指定，则根据 HAB4 PKI 树命名约定从 imgcert 路径名派生。
--dcd DCD.bin	DCD 二进制文件路径。
--dekkey DEKKEY	32/48/64 个十六进制字符：用于 AHAB 加密的数据加密密钥。该参数适用于具有 AHAB 安全系统的处理器。
--device {name of the selected processor}	目标处理器。使用以下命令显示支持的处理器列表 securep -h
--dual-image-boot-cfg DUAL_IMAGE_BOOT_CFG.json	具有双 image 启动配置的 JSON 文件；文件格式由 schema/dual_image_boot_schema_v?.json 指定。该参数适用于支持双 image 启动的处理器
--ele-firmware ELE_FIRMWARE	EdgeLock Enclave (ELE) 固件文件的路径。该参数适用于具有 AHAB 安全系统的处理器的加密启动类型。
--firmware-version FIRMWARE_VERSION	应用程序 image 固件的版本。
--iee-config IEE_CONFIG.json	带有 IEE 配置的 JSON 文件。参数文件的路径为 schema/iee_image_encryption_schema_v?.json。该参数仅适用于 IEE 加密启动类型。
--image-version IMAGE_VERSION	可启动 image 的版本可以是 4 字节格式，例如 0xFFFE0001（低位 2 字节为真实版本号，高位 2 字节 为低位 2 字节的取反值），也可以只是真实版本号 （2 个字节）。参数仅适用于在构建选项卡上支持 image 版本的处理器。
--img-cert IMG_CERT	用于对 image 进行签名的公共 IMG 密钥文件路径。建议将该命令与已初始化密钥管理的 workspace 一 起使用。如果未在 workspace 设置指定密钥，则会 通过该命令进行导入。
--iped-cfg IPED_CFG.json	具有 PRINCE 配置的 JSON 文件；文件格式由 schema/prince_config_schema_v?.json 指定
--keyblob-keyid KEYBLOB_KEYID	32 位值：Keyblob 加密密钥标识符。该参数适用于具有 AHAB 安全系统的处理器。
--keysource \{OTP, KeyStore\}	RT5xx/6xx 安全 image 的密钥 source
--life-cycle LIFE-CYCLE-ID	请求的处理器生命周期状态。使用 securep -h 命令显示支持的处理器列表。
--otfad-config OTFAD_CONFIG.json	具有 OTFAD 配置的 JSON 文件。参数文件的路 径为 schema/otfad_image_encryption_schema_v?.json。该参数仅适用于 OTFAD 加密启动类型。
--otp-cfg OTP_CFG.json	带有 USER OTP 配置的 JSON 文件的路径。建议 从 OTP 配置框中导出文件。
--prince-cfg PRINCE_CFG.json	带有 PRINCE 配置的 JSON 文件。请参阅 SEC 安 装文件夹中的 bin/schema/prince_config_schema_v<version>.json。
--romcfg-cfg ROMCFG_CFG.json	带有 USER ROMCFG 配置的 JSON 文件的路径。建议从 IFR 配置框中导出文件。

参数	描述
--save-settings	保存 workspace 的设置。
--sbkek/--cust_mk_sk/--sb3kdk SBKEK	64 个十六进制字符：用作处理 SB2 文件的密钥加密密 钥的密钥；仅安全二进制 image 需要； 如果未指 定，将从 workspace 中获取。
--script-only	仅生成脚本，不启动。
--secret-key-type \{AES-128, AES-192, AES-256\}	HAB 加密算法，默认是基于处理器不同。
--source-image SOURCE_IMAGE	输入创建 bootable image 所用的 source image。
--start-address START_ADDRESS	输入源镜像程序起始地址。只有二进制 source image 需要使用该参数。
--target-image TARGET_IMAGE	生成 bootable image 的目标路径。
--trust-provi {disabled, device_hsm, el2go_indirect, wpc_no_provi, wpc_device_hsm}	信任配置类型。el2go_indirect 代表 Edgelock 2GO 代理流程。
--trust-zone TRUST_ZONE	disabled（如果配置不是生成 image 的一部分）或 default（如果默认启用 TrustZone 配置）或自定义 Trust Zone 配置 JSON 文件的路径。
--userkey USERKEY	适用于 RT5xx/6xx 安全 image 的密钥：对于 OTP 密钥源，它代表主密钥；对于密钥库，它表示用于签名的密钥。
-v, --verbose	增加输出信息
-w WORKSPACE, --workspace WORKSPACE	Workspace 位置，workspace 目录的路径。注意：Workspace 中的设置都会自动加载。所有命令行参数都可用于覆盖加载的设置。
--xip-enc-otpmk-config XIP_ENC_OTPMK_CONFIG.json	带有 OTPMK 配置的 XIP 加密的 JSON 文件。参数文件的路径为 schema/xip_enc_otpmk_schema_v?.json。参数仅适用于 XIP 加密（BEE OTPMK）和 XIP 加密（OTFAD OTPMK）启动类型。
--xmcd-cfg XMCD_CFG	具有 XMCD 配置（简化或完整）的 YAML 或二进制文件的路径；有关文件格式，请参阅 SPSDK 命令 nxpimage bootable-image xmcd get-templates。

引导设备参数 (互斥)

参数	描述**miao**
--boot-device VALUE	预定义的启动存储器。运行 securep --help 以查看所有支持的引导类型。
--boot-device-file BOOT_DEVICE_FILE	具有启动存储器配置的文件
--boot-device-type TYPE	启动内存类型，以下类型之一：flex-spi-nor, flex_spi_nand, ifr_memory, onchip_memory, onchip_ram, sdhc_emmc, sdhc_emmc_mpu, sdhc_sd_card, sdhc_sd_card_mpu, semc_nand, serial_downloader, xspi-nor。应用默认预定义的此类启动内存。

8.2 Write

write 命令支持 **Write image **中所描述的操作。

下面对 **write** 指令参数进行介绍：

烧写参数描述

参数	描述
-h, --help	显示帮助信息，然后退出。
--source-image SOURCE_IMAGE	要上传到目标的 source image 路径
--write-params-cfg WRITE_PARAMS_CFG.json	JSON 文件，其中包含所需的参数和所需要烧写的 fuses（或影子寄存器）。请参见安装文件夹中的schema/write_parameters_schema_v?.json。
--life-cycle LIFE-CYCLE-ID	请求的处理器生命周期状态。使用 securep -h 命令显示支持的处理器列表。
--trust-provi {disabled, device_hsm, el2go_indirect, wpc_no_provi, wpc_device_hsm}	配置信任域类型。
-v, --verbose	增加输出信息
--device {name of the selected processor}	目标处理器。使用以下命令显示支持的处理器列表 securep -h
--boot-type VALUE	安全启动类型。运行 securep --help 以查看所有支持的启动类型。
--script-only	仅生成脚本，不启动。
-w WORKSPACE, --workspace WORKSPACE	workspace 位置。**注意：**Workspace 中的设置都会自动加载。所有命令行参数都可用于覆盖加载的设置。
--debug-probe PROBE	选择调试探针。使用 `-debug-probe auto` 选择任何调试探针。使用 `-debug-probe invalid` 列出所有连接的调试探针。

有关启动设备参数，请参阅 引导设备参数 (互斥)

连接参数（互斥）

参数	描述
--usb VID PID	通过 vid/pid 表示的 USB HID 连接到设备。USB HID 连接为默认设置。vid/pid 可以指定为十进制形式（例如，123）或十六进制形式（例如，0xbeef）。
--uart UART	通过 UART 连接到设备。指明 COM 端口（参见 --baud-rate argument）。示例： --uart COM3
--i2c address speed_kHz	通过 USB 连接 I2C 设备。以 kHz 为单位指定 I2C 设备地址和时钟。如果未指定 --sio-device 参数，则会自动选择 SIO 设备。示例： —i2c 0x10 400
--spi speed_kHz polarity phase	通过 USB 桥接器通过 SPI 连接到目标。以 kHz、极性（SPI CPOL 选项） 和相位 （SPI CPHA 选项）为单位指定 SPI 时钟。如果未指定 -sio-device 参数，则会自动选择 SIO 设备。示例： --sp i 1000 1 1
--baud-rate BAUD_RATE	通过指定波特率的 UART 连接到设备。-uart 参数也必须指定。示例： --baud-rate 9600
--sio-device SIO_DEVICE	通过指定的 SIO 通过 USB-SIO（I2C 或 SPI）连接到设备。--i2c 或 --spi 参数也必须指定。示例： --sio-device HID\VID_1FC9&PID_0090&MI_03\7&96E050B&0&0000

注意： 要连接到开发板，必须指定 USB 或串行端口。如果未指定任何内容，则 USB 会自动检测。

8.3 Generate keys

Generate 命令支持 **Generate Keys** 描述的操作。与 GUI 相比，命令行功能受到限制。

下面对 **generate** 指令参数进行介绍：

生成密钥参数

参数	描述
`-h, --help`	显示帮助信息，然后退出。
`--keys-cfg KEYS_CFG.json`	带有密钥配置的 JSON 文件。
`--device {name of the selected processor}`	目标处理器。使用以下命令显示支持的处理器列表 `securep -h`
`--boot-type VALUE`	安全启动类型。运行 `securep --help` 以查看所有支持的启动类型。
`--script-only`	仅生成脚本，不启动。
`-w WORKSPACE, --workspace WORKSPACE`	workspace 位置。注意： Workspace 中的设置都会自动加载。所有命令行参数都可用于覆盖加载的设置。

有关启动设备参数，请参阅 引导设备参数 (互斥)

8.4 Manufacture

Manufacture 命令允许并行运行选定的脚本多次，每次用于不同的连接。**manufacture** 命令可以使用以下参数：

Manufacture 参数

参数	描述
`-h, --help`	显示帮助信息，然后退出。
`--script_path SCRIPT_PATH`	执行的脚本的路径。
`--script_params SCRIPT_PARAMS`	脚本的参数。更多详情，请参阅 量产工具。
`--connections CONNECTIONS {CONNECTIONS ...}`	制造过程中使用的所有连接设备的列表，格式为 `-p <port\>,<baud\>` 或 `-u <usb-path\>` 或 `-l usb,<usb-path\>,spi\[,<port\>,<pin\>,<speed\_kHz\>,<polarity\>,<phase\>\]` 或 `-l usb,<usb-path\>,i2c\[,<address\>,<speed\_kHz\>\]`。要查找所有可用的 USB/USB-SIO 连接，请自动使用 `-u <autodetect-all-USBs\>` 或 `-l usb,<autodetect-all-USBSIOs\>,spi\[,<port\>,<pin\>,<speed\_kHz\>,<polarity\>,<phase\>\]` 或 `-l usb,<autodetect-all-USBSIOs\>,i2c\[,<address\>,<speed\_kHz\>\]`。自动检测选项不能与其他选项一起使用。SPSDK 文档中描述了 SIO 操作的参数和默认值。

8.5 Devices info

使用 **devices-info** 命令，您可以获得有关支持的处理器及其支持的启动设备的信息。

devices-info命令有两种模式：

设备信息命令模式

模式	说明
processors	有关支持的处理器及其支持的启动设备的信息。这是默认模式。
boot-devices	有关支持的启动设备内存的信息。

以下参数可用于 **devices-info** 命令：

Devices-info 设备信息特定参数

参数	描述
-h, --help	显示帮助信息，然后退出。
--device DEVICE	在处理器称中搜索的设备名称子字符串。对于引导设备，需要处理器的全名来显示支持的内存。如果未提供，则返回未筛选的数据。
--format \{json,txt\}	输出的格式。文件输出默认为 json，STDOUT 为 txt。有关处理器 json信息，请参阅 schema/devices_info_schema_v?.json。有关启动设备的 json信息，请参阅 schema/boot_devices_info_schema_v?.json。
--output OUTPUT	存储输出的文件的路径。如果未提供，输出将打印到 STDOUT。

8.6 Test Connection

test-connection命令执行处理器特定的连接测试。必须在工作区中或通过命令行选项指定设备。如果在工作区或选项中未指定连接，则使用默认连接。如果连接的设备匹配且处于 ISP 模式，则连接测试成功。

支持的参数

参数	描述
-h, --help	显示帮助信息，然后退出。
-v, --verbose	增加输出信息
-w WORKSPACE, --workspace WORKSPACE	workspace 位置。
--device {name of the selected processor}	目标处理器。使用以下命令显示支持的处理器列表 securep -h

连接参数（互斥）

参数	描述
--usb VID PID	通过 vid/pid 表示的 USB HID 连接到设备。USB HID 连接为默认设置。vid/pid 可以指定为十进制形式（例如，123）或十六进制形式（例如，0xbeef）。
--uart UART	通过 UART 连接到设备。指明 COM 端口（参见 --baud-rate argument）。示例： --uart COM3
--i2c address speed_kHz	通过 USB 连接 I2C 设备。以 kHz 为单位指定 I2C 设备地址和时钟。如果未指定 --sio-device 参数，则会自动选择 SIO 设备。示例： —i2c 0x10 400

参数	描述
`--spi speed_kHz polarity phase`	通过 USB 桥接器通过 SPI 连接到目标。以 kHz、极性（SPI CPOL 选项） 和相位 （SPI CPHA 选项）为单位指定 SPI 时钟。如果未指定 -sio-device 参数，则会自动选择 SIO 设备。示例： `--sp i 1000 1 1`
`--baud-rate BAUD_RATE`	通过指定波特率的 UART 连接到设备。-uart 参数也必须指定。示例： `--baud-rate 9600`
`--sio-device SIO_DEVICE`	通过指定的 SIO 通过 USB-SIO（I2C 或 SPI）连接到设备。`--i2c` 或 `--spi` 参数也必须指定。示例： `--sio-device HID\VID_1FC9&PID_0090&MI_03\7&96E050B&0&0000`

8.7 Detect USB devices

detect命令可以识别通过 USB 连接的任何处理器。要成功进行检测，处理器必须处于 ISP 模式。

8.8 Start flashloader

在 RT1xxx 处理器上启动 flashloader。 以下参数可用于**start-flashloader**命令：

start-flashloader 命令的特定参数

参数	描述
`-h, --help`	显示帮助信息，然后退出。
`-v, --verbose`	增加输出信息
`-w WORKSPACE, --workspace WORKSPACE`	workspace 位置。
`--script-only`	仅生成脚本，不启动。

连接参数（互斥）

参数	描述
`--usb VID PID`	通过 vid/pid 表示的 USB HID 连接到设备。USB HID 连接为默认设置。vid/pid 可以指定为十进制形式（例如，`123`）或十六进制形式（例如，`0xbeef`）。
`--uart UART`	通过 UART 连接到设备。指明 COM 端口（参见 `--baud-rate argument`）。示例： `--uart COM3`
`--i2c address speed_kHz`	通过 USB 连接 I2C 设备。以 kHz 为单位指定 I2C 设备地址和时钟。如果未指定 `--sio-device` 参数，则会自动选择 SIO 设备。示例： —i2c 0x10 400
`--spi speed_kHz polarity phase`	通过 USB 桥接器通过 SPI 连接到目标。以 kHz、极性（SPI CPOL 选项） 和相位 （SPI CPHA 选项）为单位指定 SPI 时钟。如果未指定 -sio-device 参数，则会自动选择 SIO 设备。示例： `--sp i 1000 1 1`
`--baud-rate BAUD_RATE`	通过指定波特率的 UART 连接到设备。-uart 参数也必须指定。示例： `--baud-rate 9600`
`--sio-device SIO_DEVICE`	通过指定的 SIO 通过 USB-SIO（I2C 或 SPI）连接到设备。`--i2c` 或 `--spi` 参数也必须指定。示例： `--sio-device HID\VID_1FC9&PID_0090&MI_03\7&96E050B&0&0000`

8.9 Apply settings

apply-settings命令支持两种主要用例

- 修改现有workspace
- 创建新workspace

要修改现有workspace，请指定工作区的路径和要更改的选项。可以使用通用选项来实现此目的。

要创建新的workspace，至少需要指定启动设备和设备选项。

apply-settings 命令参数

参数	描述
`-h, --help`	显示帮助信息，然后退出。
`-v, --verbose`	增加输出信息
`-w WORKSPACE, --workspace WORKSPACE`	workspace 位置。
`--life-cycle LIFE-CYCLE-ID`	请求的处理器生命周期状态。使用 `securep -h` 命令显示支持的处理器列表。
`--trust-provi {disabled, device_hsm, el2go_indirect, wpc_no_provi, wpc_device_hsm}`	配置信任域类型。
`--device {name of the selected processor}`	目标处理器。使用以下命令显示支持的处理器列表 `securep -h`
`--boot-type VALUE`	安全启动类型。运行 `securep --help` 以查看所有支持的启动类型。
`--boot-device VALUE`	预定义的启动存储器。运行 `securep --help` 以查看所有支持的引导类型。
`--boot-device-file BOOT_DEVICE_FILE`	具有启动存储器配置的文件
`--boot-device-type {flex-spi-nor, flex_spi_nand, onchip_memory, sdhc_sd_card, semc_nand}`	启动存储器类型。设置此类型的默认预定义引导存储器。

连接参数（互斥）

参数	描述
`--usb VID PID`	通过 vid/pid 表示的 USB HID 连接到设备。USB HID 连接为默认设置。vid/pid 可以指定为十进制形式（例如，`123`）或十六进制形式（例如，`0xbeef`）。
`--uart UART`	通过 UART 连接到设备。指明 COM 端口（参见 `--baud-rate argument`）。示例： `--uart COM3`
`--i2c address speed_kHz`	通过 USB 连接 I2C 设备。以 kHz 为单位指定 I2C 设备地址和时钟。如果未指定 `--sio-device` 参数，则会自动选择 SIO 设备。示例： —i2c 0x10 400
`--spi speed_kHz polarity phase`	通过 USB 桥接器通过 SPI 连接到目标。以 kHz、极性（SPI CPOL 选项） 和相位 （SPI CPHA 选项）为单位指定 SPI 时钟。如果未指定 -sio-device 参数，则会自动选择 SIO 设备。示例： `--sp i 1000 1 1`
`--baud-rate BAUD_RATE`	通过指定波特率的 UART 连接到设备。-uart 参数也必须指定。示例： `--baud-rate 9600`

参数	描述
--sio-device SIO_DEVICE	通过指定的 SIO 通过 USB-SIO（I2C 或 SPI）连接到设备。--i2c 或 --spi 参数也必须指定。示例：--sio-device HID\VID_1FC9&PID_0090&MI_03\7&96E050B&0&0000

8.10 Environment variables in filepath-based arguments

SEC 工具接受指定路径的所有参数中的环境变量，例如：

```
securep.exe -w /workspaces/mcuxprovi --device MIMX9596 --boot-device-type
onchip_ram --boot-type unsigned build --additional-images
additional_images_cfg.json --ele-firmware
"${MCUX_SDK_16}\firmware\edgelock\mx95a0-ahab-container.img" --save-settings
```

8.11 Command-line examples

8.11.1 Example: How to build and write an image for configuration stored in the workspace folder

在此示例中，假设 GUI 已准备好 workspace 内的完整配置（生成密钥、配置构建 image、 配置写入 image）。

```
securep.exe -w /workspaces/mcuxprovi build
```

```
securep.exe -w /workspaces/mcuxprovi write
```

更详尽的例子请使用以下命令查看：

```
securep.exe print-cli-examples
```

8.11.2 Example how to use args-file arguments

在以下示例中，CLI 参数将转换为 JSON 文件。

构建命令行中的参数：

```
securep.exe -w /workspaces/mcuxprovi --device MIMX9596 --boot-device-type
onchip_ram --boot-type unsigned build --additional-images
additional_images_cfg.json --ele-firmware mx95a0-ahab-container.img --save-
settings
```

上述构建参数的 **args-file** 参数的用法：

```
securep.exe --args-file args_file_build.json
```

args_file_build.json 内容（仅列出第一个附加 image）：

```
{
  "cli_args": {
    "-w": "/workspaces/mcuxprovi",
    "--device": "MIMX9596",
    "--boot-device-type": "onchip_ram",
    "--boot-type": "unsigned",
```

```
    "build": [],
    "--additional-images": {
      "images": [
        {
          "entry_type": "oei_ddr",
          "container_set": "#1",
          "extra_settings": {
            "lpddr_imem_path": "${ENV_VAR_DDR}/lpddr5_imem_v202311.bin",
            "lpddr_imem_qb_path": "${ENV_VAR_DDR}/lpddr5_imem_qb_v202311.bin",
            "lpddr_dmem_path": "${ENV_VAR_DDR}/lpddr5_dmem_v202311.bin",
            "lpddr_dmem_qb_path": "${ENV_VAR_DDR}/lpddr5_dmem_qb_v202311.bin",
            "oei_ddr_path": "source_images/oei-m33-ddr.bin"
          }
        }
      ]
    },
    "--ele-firmware": "source_images/mx95a0-ahab-container.img",
    "--save-settings": []
  }
}
```

用于写入的命令行中的参数：

```
securep.exe -w /workspaces/mcuxprovi --device MIMX9596 --boot-device-type
onchip_ram --boot-type unsigned write --source-image bootable_images/flash.bin
```

上述写入参数的 **args-file** 参数的用法：

```
securep.exe -w /workspaces/mcuxprovi --args-file args_file_write.json
```

args_file_write.json 内容：

```
{
  "cli_args": {
    "--device": "MIMX9596",
    "--boot-device-type": "onchip_ram",
    "--boot-type": "unsigned",
    "write": [],
    "--source-image": "bootable_images/flash.bin"
  }
}
```

直接在命令行中传递的参数与 args 文件中的参数组合在一起。不能在两个地方都指定参数。

8.12 Command-line tools

SEC 工具使用以下命令行工具产生密钥并生成/烧写 image：

openssl： 密钥生成

spsdk： Secure Provisioning SDK，有关更多详细信息，请参见 main menu > Help > SPSDK Online Documentation。以下工具作为 SPSDK 的一部分提供：

- **blhost**： 替换旧的 blhost 工具
- **dk6prog**：用于读取和编程 DK6 目标设备闪存的工具。
- **el2go-host**： 管理 EdgeLock 2GO 配置操作
- **lpcprog**：用于与 LPC8xx 目标上的引导加载程序通信的实用程序。

- **nxpcrypto**： 使用密钥和证书的操作
- **nxpdebugmbox**： 调试邮箱和凭证文件生成工具
- **nxpdevhsm**： 该应用程序旨在为 OEM 初始配置设备创建 SB3 配置文件。
- **nxpdevscan**： 用于检测通过 USB、UART、I2C 和 SPI 连接到主机的 NXP 设备的实用程序。
- **nxpdice**：专为涵盖 DICE 相关操作而设计的应用程序。
- **nxpele**： 用于与目标上的 EdgeLock Enclave 进行通信的实用程序。
- **nxpfuses**：NXP Fuse 工具。
- **nxpimage**： 生成可启动 image 和 SB 文件。
- **nxpmemcfg**： 用于内存配置操作的实用程序集合。
- **nxpshe**： NXP 用于处理 SHE（安全硬件扩展）的工具。
- **nxpuuu**： 用于 i.MX MPU image 部署的应用程序。它基于 libUUU（通用更新实用程序）。
- **nxpwpc**： 涵盖 WPC 操作的实用程序。
- **pfr**： 生成受保护的 flash 区域文件（cmpa/cfpa）和 IFR。
- **sdphost**： 替换旧版 sdphost 实用程序
- **sdpshost**： 使用 SDPS 协议 (i.MX8/9) 与 i.MX 目标上的 ROM 通信的实用程序。
- **shadowregs**： shadow registers（影子寄存器）配置工具

imgtool： MCUboot 的 image 签名和密钥管理

9 Troubleshooting

本章包含已知问题和建议的解决方案。对于最后一刻的问题，请参阅作为 SEC 工具的一部分并在网络上提供的 *Secure Provisioning Tool Release Notes*（文档 MCUXSPTRN）。

9.1 General

- 应用程序必须安装在用户具有写入权限的位置。
- 默认情况下，安全配置工具不会配置处理器中所有可用的安全功能。仅配置所选启动类型所需的功能。请在 OTP/PFR 中配置其余的功能。
- 如果该工具以选项 `-v`（verbose 模式）启动，它将提供可用于分析和修复问题的其他详细信息（日志）。
  如何在Windows系统中操作：
  - 点击Windows开始按钮，运行 “Command Prompt”。
  - 使用`cd c:\nxp\SEC_Provi_25.09\bin`命令切换当前目录
  - 使用命令 `securep.exe -v`

9.2 Windows

- 在 Windows 平台上，首先确保在 PATH 中找到 Windows FIND utility（GNU FIND utils 可能会破坏此功能）。

9.3 Linux

- 在 Linux 平台上，USB 和/或串行设备文件必须是当前用户可读写的。要解决此问题，请将 `resources/udev/99-secure-provisioning.rules` 安装到 `/etc/udev/rules.d/99-secure-provisioning.rules`。电脑上可能存在其他优先级更高的规则并造成冲突。在这种情况下，请更新冲突的规则，或制作此规则文件，以便按预期顺序应用它们。

- Ubuntu 22 and USB2Serial CP210x：Ubuntu 22上有一个会造成冲突的 `brltty`包，它会造成一些 CP210x USB 转串口的问题。卸载“brltty”软件包可以解决这个问题。更多详情，请参阅 Ubuntu 错误报告 - BRLTTY 问题。
- SEC 工具与 Xorg 显示服务器配合良好。Wayland 在 Ubuntu 中默认，会导致各种 UI 故障，这就是应用程序快捷方式包含使用 Xorg (x11) 后端的配置的原因。如果 SEC GUI 在 Wayland 显示服务器下手动执行，请确保使用正确的环境变量执行，例如：
  `UBUNTU_MENUPROXY=0 GDK_BACKEND=x11 /opt/nxp/SEC_Provi_25.09/bin/securep.`
- 用于 i.MX 9x 的 nxpuuu 实用程序可能会在日志视图中产生大量转义字符。要减少转义字符的数量，请使用以下命令：

```
cd /opt/nxp/SEC_Provi_25.09/bin/tools/spsdk/
sudo sh -c "nxpuuu -udev >> /etc/udev/rules.d/70-nxpuuu.rules"
sudo udevadm control --reload
```

- P&E Micro Debug Probe 被正确检测到，并提供 PEMicro 和 pyOCD 接口。PEMicro 接口无法使用 - 必须选择并使用 pyOCD 接口。

9.4 macOS

- 具有无效输入的GUI控件会用红色背景标记。有时候修复该值可能无法立即正确地更改背景颜色，必须将焦点更改为另一个字段才能正确地重新绘制。
- PEmicro 调试探针目前尚不支持 mac 操作系统 Aarch64。
- 如果将多个 USB 转 SPI/I2C 转换器（MCU-Link PRO 板）连接到计算机，则 SPI 和 I2C 通信将不可靠。
- 如果一个环境变量（例如，对于 EL2GO）需要传播到通过启动器执行的 SEC，必须使用 `launchctl` 命令设置它，例如：
  `launchctl setenv EL2GO_GPG_SIGN_KEY MY_KEY_ID`
  对于永久环境变量，请参阅mac操作系统环境同步解决方案中描述的解决方案。

9.4.1 macOS i.MX 9x nxpuuu

由于 libUUU 和 macOS 上的底层 libusb 库存在已知限制（此限制目前尚无真正的解决方案，有关更多详细信息，请参阅libusb mac操作系统问题），因此，当未使用 sudo 执行时，使用 nxpuuu 写入固件会失败。固件的构建工作正常。

要使nxpuuu在macOS的无头模式下工作，请使用 `visudo` 命令配置sudoers文件。您可以授予用户无需提供密码即可执行任何命令的权限，或将其限制为单个应用程序，如下所述。将 `an_user` 替换为相应的用户名。

打开终端应用程序并执行：

```
sudo visudo
```

在文件末尾的”User specification”部分，添加以下行。确保绝对路径与已安装的 SEC 工具匹配。确保命令 路径中的空格使用反斜杠转义：

```
an_user ALL=(ALL) NOPASSWD: /Applications/SEC_Provi_25.09/Secure\ Provisioning\
 25.09.app/Contents/Frameworks/tools/spsdk/nxpuuu
```

然后使用`sudo`写入脚本，它允许无密码提示执行。

9.5 i.MX RT5xx/6xx

- 如果板子未复位且 fuse 中未配置复位引脚，则重复写入 QSPI flash 可能会失败。更多详情，请参阅 RT5xx/6xx 设备工作流程/启动 image 章节中的 RT5xx/6xx 设备工作流程。 特定处理器的工作流程.
- 如果使用了影子（shadow）寄存器，在应用新的 image 或者 fuse 配置之前，必须对芯片进行硬复位，因为影子（shadow）寄存器只能设置到不安全的处理器中。

9.6 i.MX RT1024

- 由于 flashloader 的局限，MIMXRT1024-EVK 板不支持将 SD 卡作为启动存储设备。

9.7 i.MX RT118x

- 对于 i.MX RT118x，附加image可能无法与通过 UART 的串行下载器一起使用。

9.8 MC56F818xx and MWCT2xD2

- 在安全启动的构建过程中，安全二进制文件的创建可能偶尔失败。此问题影响设备配置 SBx 文件和应用程序 SBx 文件。 构建脚本可能会因以下错误 `SpsdkNoDeviceFoundError`而终止。此问题似乎是由 ISP 连接的短暂中断引起的。重新启动 **Build image** 操作以继续。

9.9 Debug probes

- 检测 PEmicro 探针以及在检测到 PEmicro 后检测其他探针存在以下局限性：
  1. 首次尝试检测前必须连接 PEmicro，如果未连接将无法检测，必须重新启动 SEC 工具才能检测成功。
  2. 当检测到 PEmicro 时，如果不重新启动 SEC 工具，就不可能检测到任何其他 pyOCD 探测器。此外，如果一旦检测到 PEmicro，即使不再连接，它也将始终被列为可用探测器。
- 调试探测器驱动程序对USB驱动程序的依赖性在ReleaseNotes.txt的“系统要求”中进行了说明。
- 系统路径中未找到 LinkServer 二进制文件。在 Linux 系统中，可以通过创建指向 LinkServer 可执行文件的符号链接来解决此问题。要使系统范围内可用，请将链接放在 /usr/bin/ 中；要使用户特定访问权限，请使用 ~/.local/bin/。

9.10 Debug authentication

如果打开调试端口失败，通常不会有错误消息（出于安全原因，处理器仅返回一般错误代码）。要使其正常工作，请检查以下项目：

- 处理器不应处于 ISP 模式。
- 应保护处理器。
- 如果调试端口已打开，操作可能会失败。
- 调试探针连接测试失败后，可能需要重置处理器才能打开调试端口。在 KW45xx 和 K32E1xx 系列上这是必需的。

9.11 Zephyr projects

并非每个 Zephyr 示例项目都可以按原样使用。需要额外的工作流程或项目变更。 例如，对于 MCX N 设备，QSPI 示例默认需要内部闪存中的（MCUBoot）引导加载程序。如果没有引导加载程序，则默认起始地址不可用于创建可引导 image。有关详情，请参阅给定单板的 Zephyr 文档。必须使用内部 flash 中的引导加载程

序，或者必须更改起始地址。对于提到的 MCX N 设备，可以在`build/zephyr/.config`文件中更改起始地址：

```
CONFIG_FLASH_BASE_ADDRESS=0x90000000
```

必须改为：

```
CONFIG_FLASH_BASE_ADDRESS=0x90001000
```

10 References

10.1 User Guides

10.1.1 Secure Provisioning Tool User Guide (Chinese version)

Secure Provisioning Tool User Guide (Document MCUXSPTUG_ZH)

10.1.2 Secure Provisioning Too Quick Start Guide

https://docs.mcuxpresso.nxp.com/secure/latest/quick_start_guide.html

10.1.3 User Guide for MCUXpresso Config Tools

User Guide for MCUXpresso Config Tools (Desktop) (document GSMCUXCTUG)

10.2 Release Notes

10.2.1 Secure Provisioning Tool Release Notes

https://docs.mcuxpresso.nxp.com/secure/latest/release_notes.html

Secure Provisioning Tool Release Notes (document MCUXSPTRN)

10.2.2 i.MX Linux Release Notes

i.MX Linux Release Notes (document RN00210)

10.3 NXP SW resources

10.3.1 Secure Provisioning Tool home page

https://www.nxp.com/mcuxpresso/secure

10.3.2 MCU-Link Pro product page

http://www.nxp.com/pages/:MCU-LINK-PRO

10.3.3 LPC-Link2 product page

https://www.nxp.com/design/microcontrollers-developer-resources/lpc-link2:OM13054

10.3.4 MCUXpresso SDK Builder on NXP.com

http://mcuxpresso.nxp.com

10.3.5 EdgeLock 2GO web portal

https://edgelock2go.com

10.3.6 EdgeLock 2GO application note

AN14624 https://docs.nxp.com/bundle/AN14624/page/topics/introduction.html

10.3.7 ELE firmware

https://www.nxp.com/lgfiles/NMG/MAD/YOCTO/firmware-ele-imx-0.1.3-4e6938c.bin

10.3.8 Embedded Linux for i.MX Applications processors

https://www.nxp.com/design/design-center/software/embedded-software/i-mx-software/embedded-linux-for-i-mx-applications-processors:IMXLINUX

10.3.9 NXP i.MX main repository

https://github.com/nxp-imx

10.3.10 OEI DDR firmware

https://github.com/nxp-imx/imx-oei

10.3.11 CM33 System Manager repository

https://github.com/nxp-imx/imx-sm

10.3.12 U-Boot SPL, U-Boot repository

https://github.com/nxp-imx/uboot-imx

10.3.13 ARM Trusted Firmware

https://github.com/nxp-imx/imx-atf

10.3.14 TEE Binary repository

https://github.com/nxp-imx/imx-optee-os

10.4 Third party resources

10.4.1 PKCS#1 wikipedia article

https://en.wikipedia.org/wiki/PKCS_1

10.4.2 Security archive

RSA PKCS#1 security archive https://web.archive.org/web/20051029040347/http://rsasecurity.com/rsalabs/node.asp?id=2125

10.4.3 MCUboot documentation

https://docs.mcuboot.com/

10.4.4 pyOCD official website

http://pyocd.io/

10.4.5 Kleopatra for Windows (GPG4Win)

https://www.gpg4win.org/get-gpg4win.html

10.4.6 GnuPG downloads

https://gnupg.org/download/

10.4.7 Ubuntu bug report - BRLTTY issue

https://bugs.launchpad.net/ubuntu/+source/brltty/+bug/1970408

10.4.8 libusb macOS issue

https://github.com/libusb/libusb/issues/1014

10.4.9 macOS environment sync solution

https://github.com/ersiner/osx-env-sync

11 Revision history

修订记录

文档ID	发布日期	描述
MCUXSPTUG_25.09 v.18	2025 年 10 月 10 日	针对 MCUXpresso Secure Provisioning Tools 25.09 的更新。
MCUXSPTUG_25.03 v.16	2025 年 6 月 30 日	针对 MCUXpresso Secure Provisioning Tools 25.06 的更新。
MCUXSPTUG_25.03 v.16	2025 年 4 月 7 日	针对 MCUXpresso Secure Provisioning Tools 25.03 的更新。
MCUXSPTUG_10.0 v.15	2024 年 11 月 13 日	针对 MCUXpresso Secure Provisioning Tools v10 的更新。
MCUXSPTUG_9.0 v.14	2024 年 7 月 31 日	针对 MCUXpresso Secure Provisioning Tools v9 的更新。
MCUXSPTUG_8.0 v.13	2024 年 1 月 19 日	针对 MCUXpresso Secure Provisioning Tools v8 的更新。增加了 RT118x 设备

文档ID	发布日期	描述
		工作流程、RW61x 设备工作流程、和 MCX Nx4x/N23x 设备工作流程。
MCUXSPTUG_7.0 v.12	2023 年 7 月 12 日	针对 MCUXpresso Secure Provisioning Tools v7 的更新：较小的更新
MCUXSPTUG_6.0 v.11	2023 年3月15日	针对 MCUXpresso Secure Provisioning Tools v6 的更新：较小的更新
MCUXSPTUG_5.0 v.10	2023 年1月13日	已添加：支持 LPC55S36，“LPC55S3x 设备工作流”部分；特性章节已修改。
MCUXSPTUG_4.1 v.9	2022 年 9 月 26 日	已添加：支持 LPC55Sxx 和 LPC55xx 系列（LPC553x、LPC552x、LPC551x 和 LPC550x）、调试身份认证、Flash 编程器、和启动设备配置。
MCUXSPTUG_4.1 v.8	2022 年 6 月 24 日	针对 MCUXpresso Secure Provisioning Tools v4.1 的更新
MCUXSPTUG_4.0 v.7	2022 年 5 月 09 日	术语“Java 智能卡”替换为“智能卡”。
MCUXSPTUG_4.0 v.6	2022 年 4 月 22 日	针对 MCUXpresso Secure Provisioning Tools v4 的更新。添加了智能卡可信配置流程。添加了有关可启动 image 作为构建源的信息，更新了截图。
MCUXSPTUG_3.0 v.5	2021 年9 月 30 日	在文件的所有流程图中，首字母缩写“SPT”被替换为“SEC”。
MCUXSPTUG_3.0 v.4	2021 年 7 月 28 日	OTP/PFR 配置返工、针对 OTFAD 启动类型的新工作流/流程图、重组、新设备信息、细微更改
MCUXSPTUG_3.0 v.3	2021 年 4 月 20 日	针对 MCUXpresso Secure Provisioning Tools v3 的更新
MCUXSPTUG_2.1 v.2	2020 年 10 月 14 日	针对 MCUXpresso Secure Provisioning Tools v2.1 的更新
MCUXSPTUG_2.0 v.1	2020 年 8 月 25 日	针对 MCUXpresso Secure Provisioning Tools v2 的更新
MCUXSPTUG_1 v.0	2020 年 1 月 8 日	初始版本

12 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES

OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents