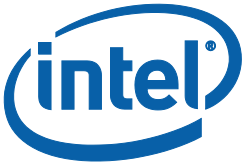


APL and GLK Secure Tokens

User Guide

Revision 1.1

November 2017

Intel Confidential

Contents

Table

§§

Revision History

Revision Number	Description	Revision Date
0.6	Initial Release	March 2015
0.8	Added instruction to update OEM KM	June 2017
1.1	Updated Globally valid description	November 2017

§§

1 Introduction

This document gives an overview of Secure Tokens for GLK platforms.

1.1 Goal

The goal of this guide is to train the user to:

1. Prepare his platform to work with Secure Tokens
2. Create Secure Tokens
3. Inject Secure Tokens to the platform
4. Clear Secure Token from platform after use.

1.2 Prerequisites

The user should download and install the following applications, included in the firmware kit:

- Intel® Platform Flash Tool (PFT)
- Mobile Signing Utility for GLK
- Intel® Flash Programming Tool (FPT)

An overview of the signing and manifesting process is described in:

- GLK Signing and Manifesting Guide

Which is included in the firmware kits.

1.3 Tools Used in this Document

The following tools are used within this document:

- Intel® Platform Plash Tool (PFT)
- Intel® Flash Programming Tool (FPT)
- Intel® Manifest Extension Utility (MEU)

1.4 Terminology

Table 1-1. Terminology

Term	Description
DnX	Download and Execute
EOM	End of Manufacture
Intel FIT	Intel® Flash Image Tool

Term	Description
IBB	Initial Boot Block
IBBL	Initial Boot Block Loader
IFWI	Integrated Firmware Image
ISH	Integrated Sensor Hub
OBB	OEM Boot Block
SUT	System Under Test
OEM KM	OEM Key Manifest

§§

2 Overview of Secure Tokens

2.1 Introduction

Secure Tokens are used in GLK platforms to allow operations otherwise blocked.

The OEM Unlock Token unlocks debug capabilities such as the following:

- North Peak debug messages
- ISH debug
- Intel® TXE debug interfaces when Secure Boot is not in legacy mode

The Lifecycle Token is used to unblock the ClearData HECI API, when the firmware setting requires a token to unblock it (the firmware settings can declare that Clear Data API is always unblocked, in which case no token is required). It is also used to all a DnX dumping of an image from a platform after End Of Manufacturing.

Tokens are digitally signed so that the target platform knows to accept them.

2.2 Preparing the Platform to Accept Secure Tokens

Secure Tokens must be digitally signed, to ensure that the target platform will authorize them.

GLK platforms are manufactured with an OEM Key Manifest as part of the IFWI image. One of the fields in the OEM Key Manifest is for the OEM Unlock Token, and another is for the Lifecycle Token. These should be populated with the hashes of the public keys, matching the private keys with which the tokens will be signed. A token whose key does not match the relevant hash in the OEM Key Manifest will be rejected by the platform.

An overview of the signing and manifesting process is described in “GLK Signing and Manifesting Guide”.

Details instruction on creating OEM key manifest are in “bring up guide”.

Both documents are included in the firmware kits.

3 Creation of Secure Tokens

3.1 Introduction

The Intel® Platform Flash Tool (PFT) includes a module which is the tool provided for Secure Token creation on GLK platforms.

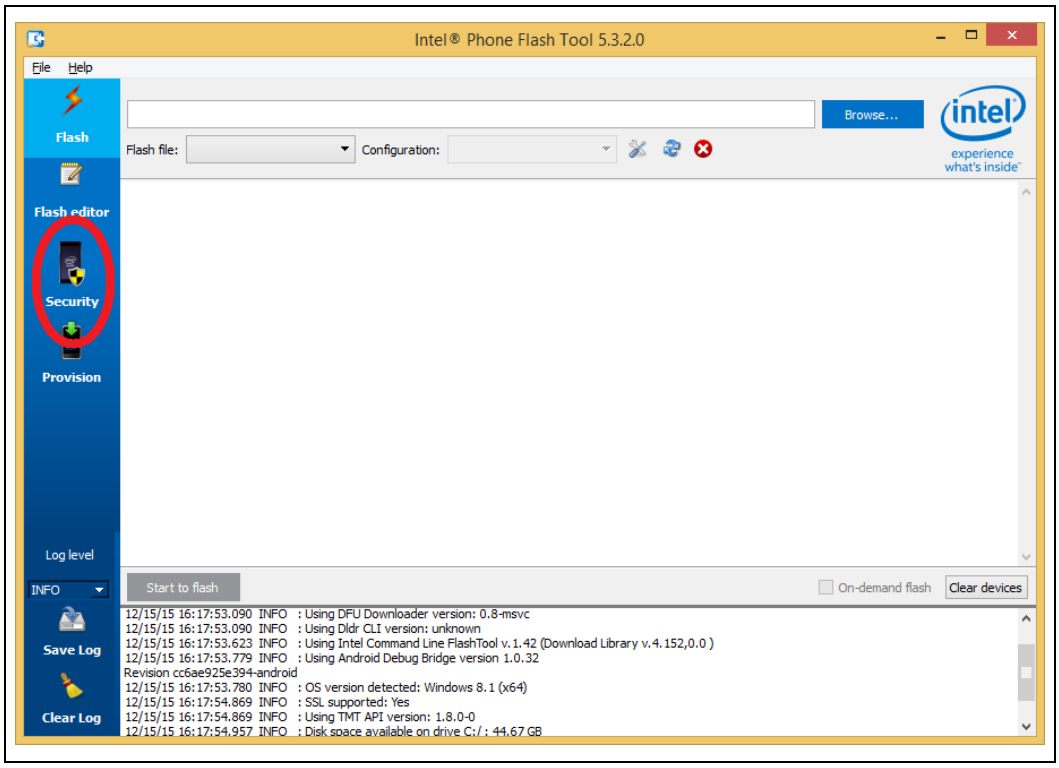
The module supports many platforms, and displays many options not supported on the GLK platforms. This guide will only cover the POR features, and show how to create Secure Tokens for GLK platforms.

3.2 Installing Intel® PFT

Install the Intel® Platform Flash Tool (PFT) and the Mobile Signing Utility for GLK, both included in the firmware kit.

3.3 Launching Intel® PFT Token Module

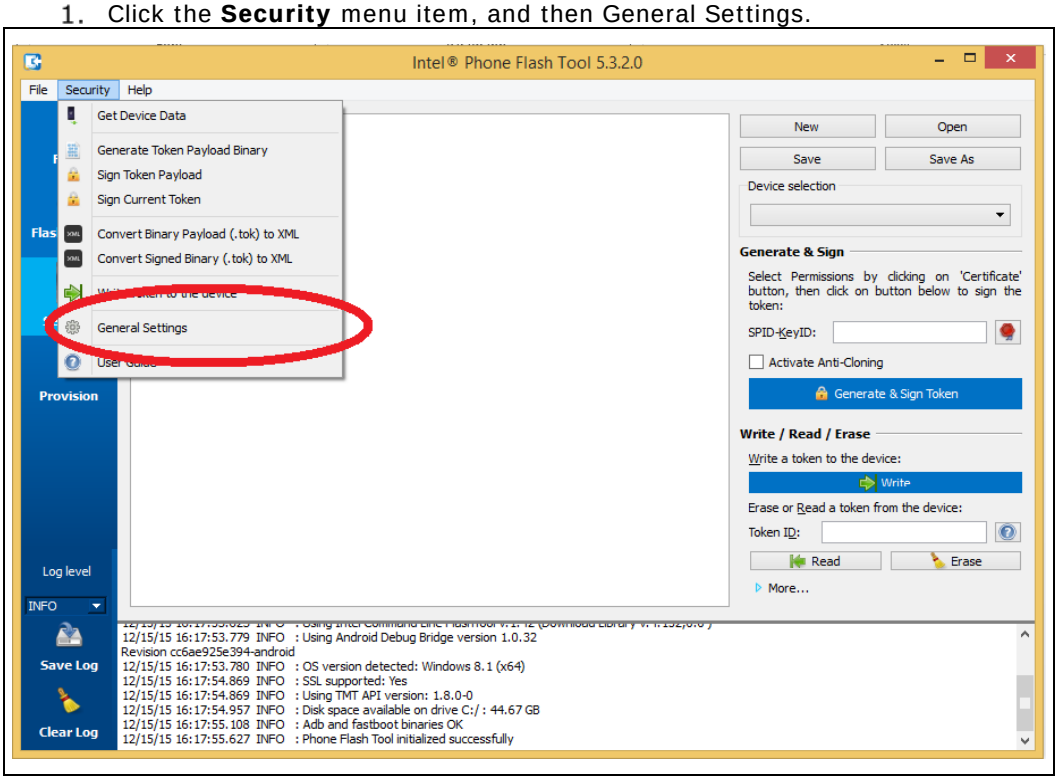
1. Open Intel® Platform Flash Tool (PFT), and click the **Security** button in the left margin.



This will change the UI to that of the token creation module.

3.4 Set General Settings

1. Click the **Security** menu item, and then General Settings.



This will open the **General Settings** Dialog.

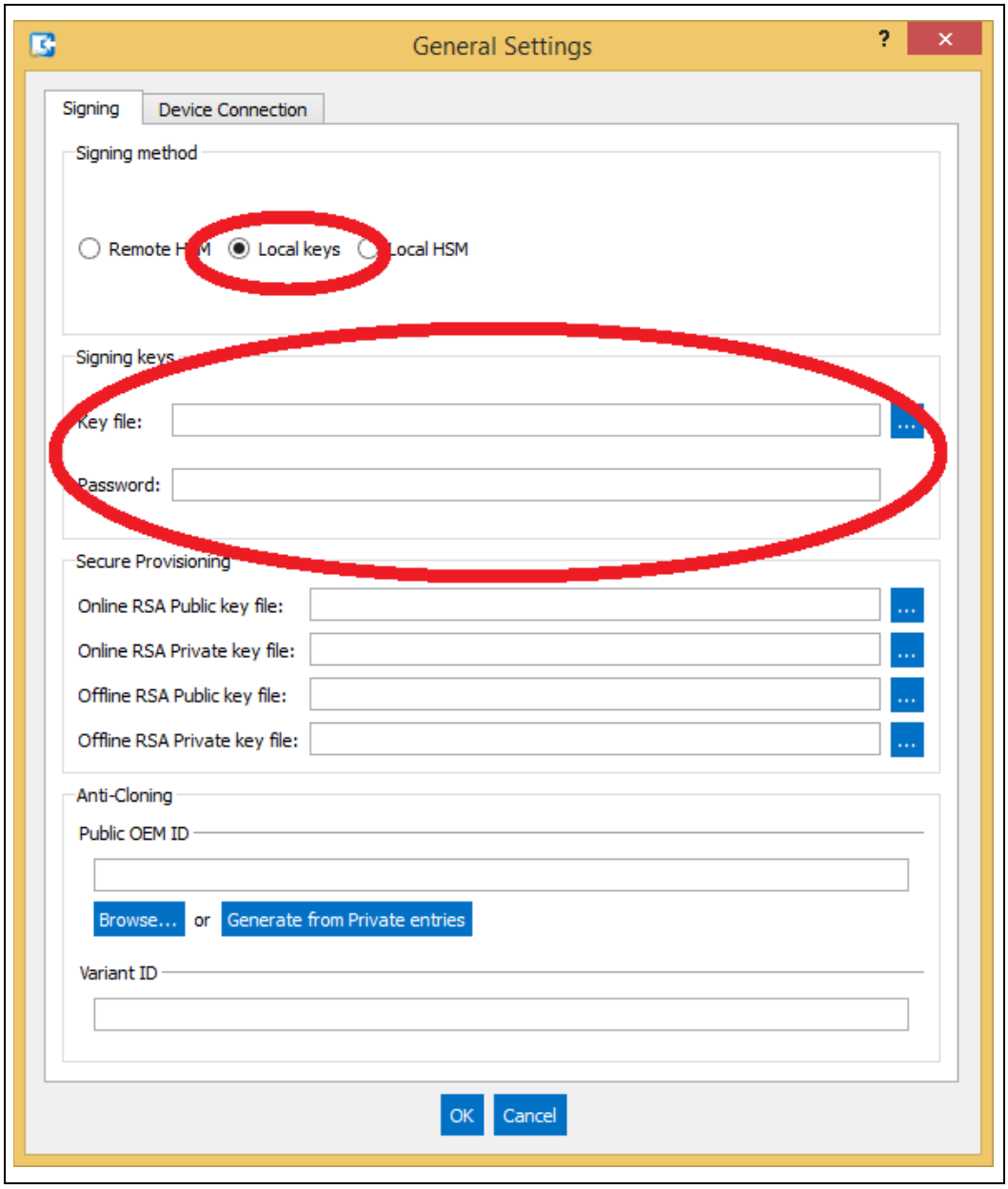
Ensure to select Local Keys as the Signing method, and then you can enter the file with the private key for signing the tokens in the Key File field.

Note: If key is not password protected, Intel® PFT does not manage to create the token. You should enter the password in the Password field.

To create a password protected private key, using OpenSSL, using for example 'foobar' as the password, run the following command from the CLI:

```
# openssl.exe genrsa -passout pass:foobar -out privkey_pwd.pem 2048
```

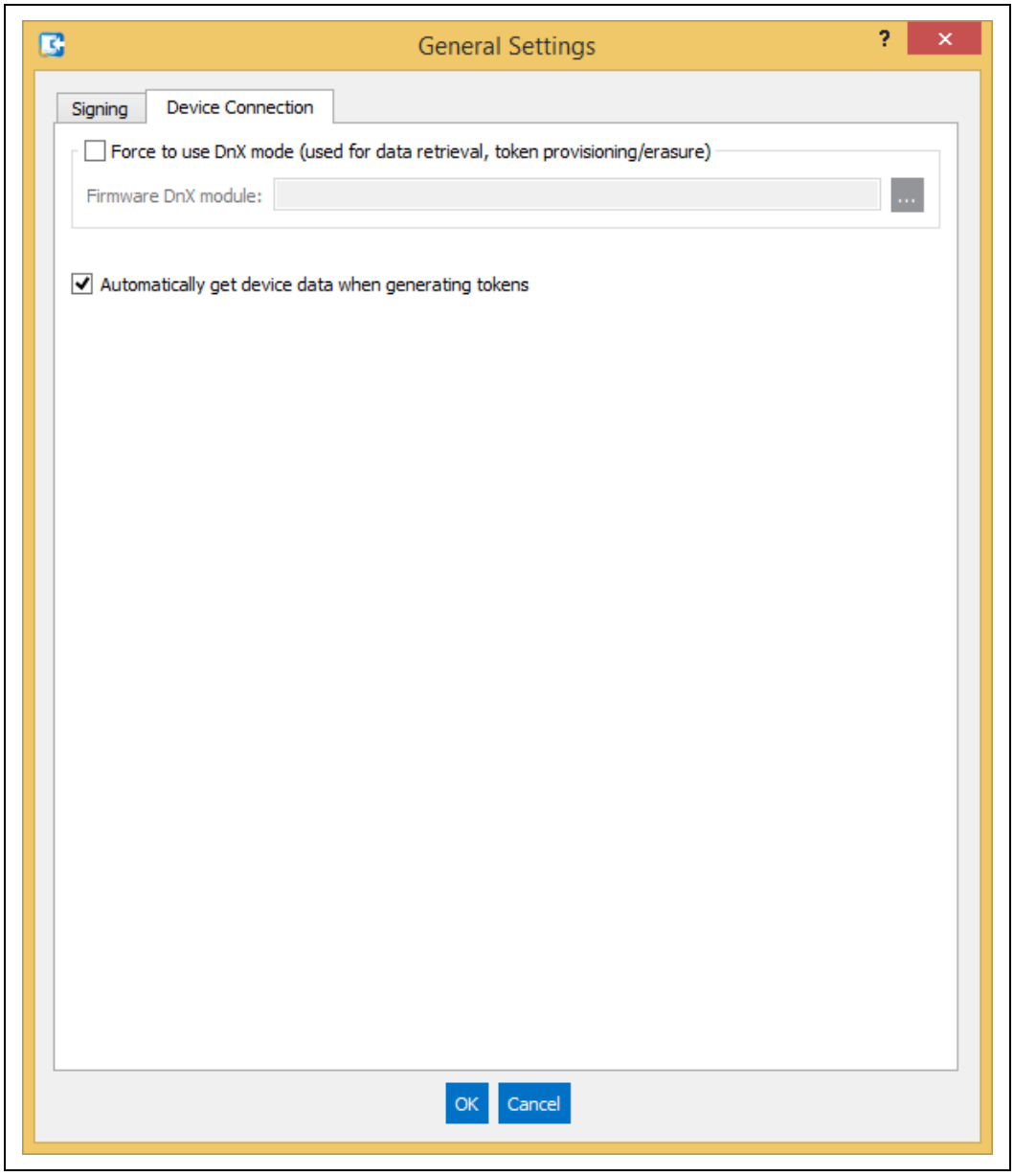
Note: Token with key hash that does not match value in OEM Key Manifest will not work. Before you continue with token creation be sure the OEM key manifest is updated. Refer Section 2.2



2. Click the **Device Connection** tab to set behaviors for creating tokens specific to particular platforms.
3. To use DnX to retrieve platform part ID information, and to inject or erase tokens, select the checkbox '**Force to use DnX mode**'.

You will then also need to enter the Firmware DnX module, which is a binary file included in the firmware kit.

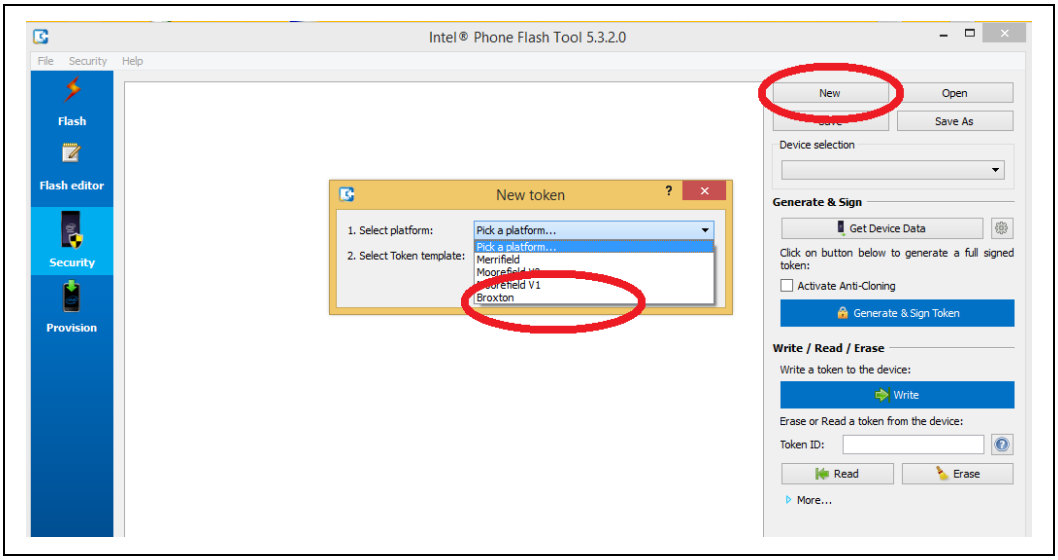
If you select the checkbox '**Automatically get device data when generating tokens**', then every token will be created with the Part ID information of the target platform, and will only be valid on that platform.

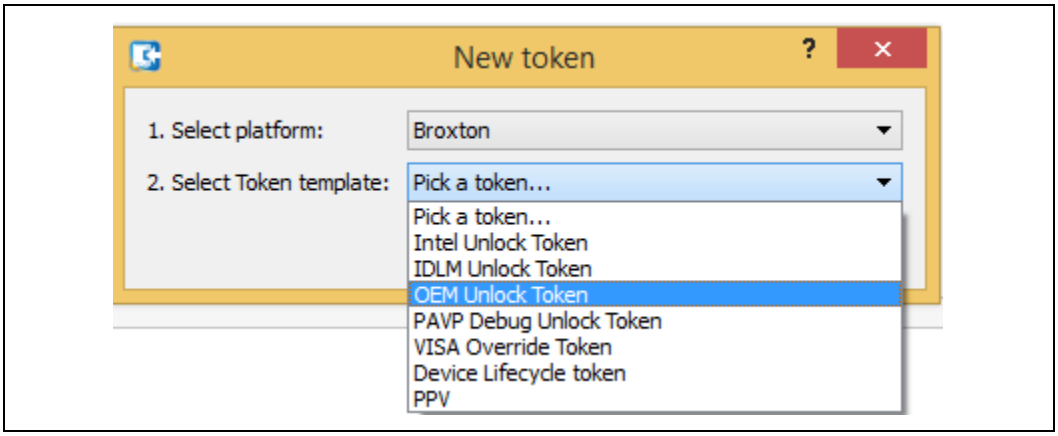


3.5 Creation the Token

1. Click the **New** button, and then select Broxton as the target platform, and OEM Unlock Token as the token template for an OEM Unlock Token, or Device Lifecycle Token to create a Device Lifecycle token.

All other options are not supported for customers.





There are multiple options that can now be set for the token (depending on which token is selected). Leave all of them with defaults, except for the following:

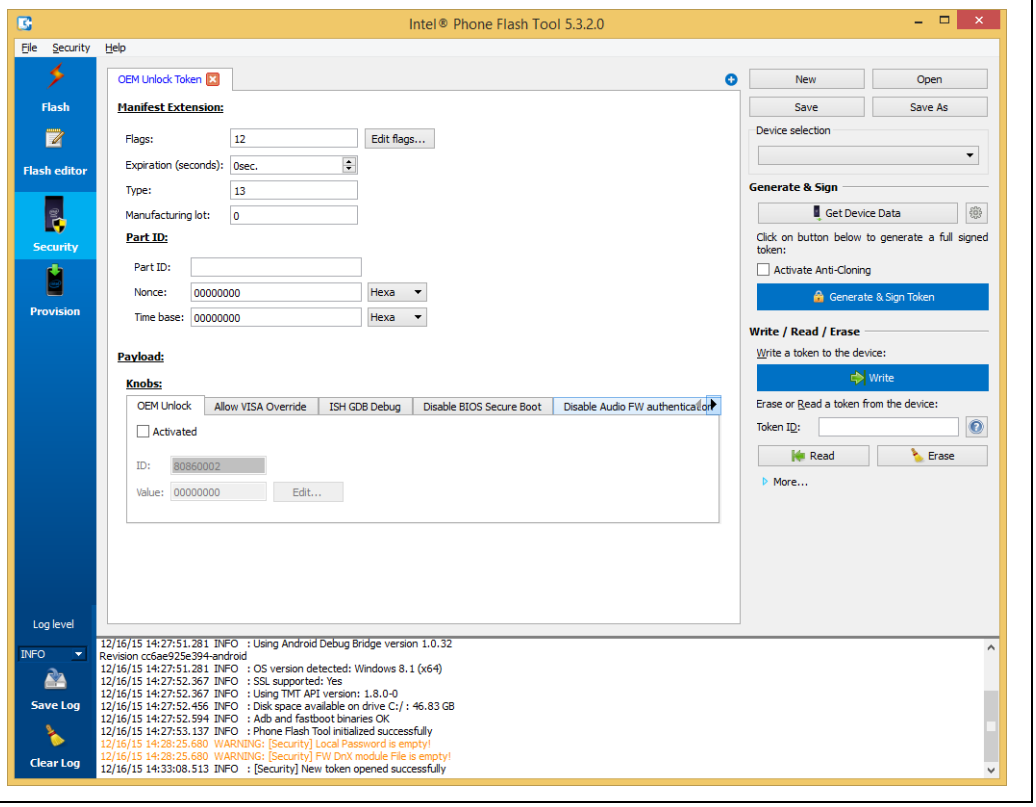
In the **Flags** section, you can set the following:

- **Globally valid**. This means that the token can be used on any platform whose token key hash matches that of the token, and is not tied to a particular platform ID. Once platform is post End of Manufacturing only part specific token is accepted i.e., only token with PID will work.
- **No Anti-replay**. Anti-Replay protection stops a token being re-used on the same device after it has been cleared. This option is only relevant for tokens tied to a particular platform ID.
- **No expiration**. This means that the token has no time limit. **Token expiration is only relevant on tokens with anti-replay**, because otherwise you can re-use the token after RTC clear.

It is recommended to use to token with time expiration and Anti-reply flag.

In the main screen you can set the following:

- **Expiration timeout** (if relevant)
- **Part ID.** This is only relevant for a token that is not Globally Valid.
- You can retrieve the Part ID data using Intel® FPT, by calling `# FPT.exe –GETPID <file>` which will retrieve the part ID into a file.
- You can open the file to copy and paste the data into the relevant fields with DnX you can set the General Setting '**Automatically get the device data when generating tokens**', and then to uncheck the **Globally valid** flag in the flags section. This will then get the Part ID data directly from the platform as the token is generated, and generate the token specifically for that platform. This requires a USB connection between the management console, and the target platform, and also that you have enabled the DnX option in the General Settings Dialog.



2. In the Payload section, you can set the 'Knobs' for the token. These define what the token allows/disables on the platform. You can check/uncheck the checkbox inside each tab to add the knob to the token, and then edit the value of the token by clicking the Edit button and selecting from the radio buttons inside. Do not attempt to change the value manually. The knobs available vary depending on the token being created. Here is an explanation of the various knobs:

Knob	Meaning
OEM Unlock	Allow an OEM (Orange) unlock
VISA override	Override default VISA signal coloring
ISH GDB Debug	Enable ISH GDB support

Knob	Meaning
Disable BIOS Secure Boot	0: Secure Boot enforced. This means BIOS BPM is validated using a production key rooted in the OEM key manifest. 1. Secure Boot uses R and D keys and policies to validate the BIOS BPM. This causes the CSE to use the R and D keys and R and D policy stitched into the image as a file key located in flash for BIOS BPM image authentication, rather than a production key rooted in the OEM key manifest. 2. Secure boot disabled. All secure boot flows are disabled. Actually, this means that the BIOS BPM is not validated.

§§

4 Injection of Token on Platform

4.1 Introduction

Tokens can be injected into a platform using the HECI interface, and tools such as FPT, or using DnX. The PFT tool, used for creating tokens, can also be directly used to inject the token using DnX, via a UI button. Some tokens can also be compiled into the firmware image, using FIT.

4.2 Injection

4.2.1 Injection using Intel® FPT

The OEM Unlock Token can be injected into a platform using Intel® FPT, running on the platform OS. The token will be read by the firmware on the next platform reset, so the machine should be rebooted after injection. It will remain there until it is erased, or the firmware is re-flashed, erasing the token. Intel recommends never releasing to customers a platform with an erased OEM Unlock Token, but to re-flash the full firmware image instead.

The Lifecycle token is used to enable the Clear Data HECI API (when the image settings determine that the API is blocked without a token). This token can be injected using Intel® FPT, running on the platform OS. The system should not be rebooted. The Clear Data HECI command should be sent immediately, before the system is rebooted. Rebooting the system will clear the token.

Note: The token is validated during the Clear Data flow, using the hash value for the token in the OEM Key Manifest.

Operation	Command Line
Lists the token ID currently on the system	Fpt.exe - LISTTOKENS
Writes the token where the filename is the token name	Fpt.exe -WRITETOKEN<file>
Delete the token for the token ID provided	FPT.exe - ERASETOKEN<pid>

Note: These APIs are unable to give any indication if the token passed validation or not.

4.2.2 Injection Using DnX

The OEM Unlock Token can be injected into a platform using DnX. This requires the management console to be connected to the target platform with a USB cable. The target machine must enter into DnX mode. Depending on OEM implementation, there

may be an explicit hardware trigger for this. Alternatively, connecting the target system to the management console should enter the target system into DnX mode until the DnX timeout is reached. The DnX APIs for tokens are only available while the target system is in DnX mode.

The token will be read and validated by the firmware on the next platform reset, so the machine should be rebooted after injection. It will remain there until it is erased, or the firmware is re-flashed, erasing the token. Intel recommends never releasing to customers a platform with an erased OEM Unlock Token, but to re-flash the full firmware image instead.

The Lifecycle token is used to enable the Clear Data APIs (when the image settings determine that the API is blocked without a token). This token is injected as part of the Set Device Lifecycle Stage API call, and the system should not be rebooted. The DnX API that injects the token also validates it, and can return a failure message if the validation fails.

Note: This validation is not done using the hash value for the token in the OEM Key Manifest, but using the OEM Key Manifest hash itself, which is stored in an FPF. The Clear Data command should be sent immediately before the DnX session is closed, which occurs when the system is rebooted. Ending the DnX session will clear the token.

The DnX token API is supported by the Intel® PFT command line:

Operation	Command Line
Read token in slot 0 to read_token.bin	`dnxFwDownloader.exe --command readtoken --fw_dnx DNXP_0x1.bin --path read_token.bin --slot 0`
Write token OEMUnlock.bin to slot 0	`dnxFwDownloader.exe --command writetoken --fw_dnx DNXP_0x1.bin --token OEMUnlock.bin --slot 0`
Erase token in slot 0	`dnxFwDownloader.exe --command erasetoken --fw_dnx DNXP_0x1.bin --slot 0`
Set Device Lifecycle stage, using token DeviceLifecycle.bin	`dnxFwDownloader.exe --command setdevicelifecyclestage --fw_dnx DNXP_0x1.bin –token DeviceLifecycle.bin`

Note: Each of the DnX commands requires the passing of the DnX binary DNXP_0x1.bin to the platform. This binary file is included in the firmware kit.

Note: The APIs are unable to give any indication if the OEM Unlock token passed validation or not. The Set Device Lifecycle Stage API does return a meaningful error code and string to indicate success or failure of the token validation. For example, if the token validation failed, the following text is displayed:

```
s\dnx_v2>dldrcli.exe --command setdevicelifecyclestage --token token.bin
CLI version: 3.0.0.1048
Starting SET DEVICE LIFECYCLE STAGE procedure
SET DEVICE LIFECYCLE STAGE procedure failed
Error code: 0x80000009
Error message: Manifest authentication failure
```

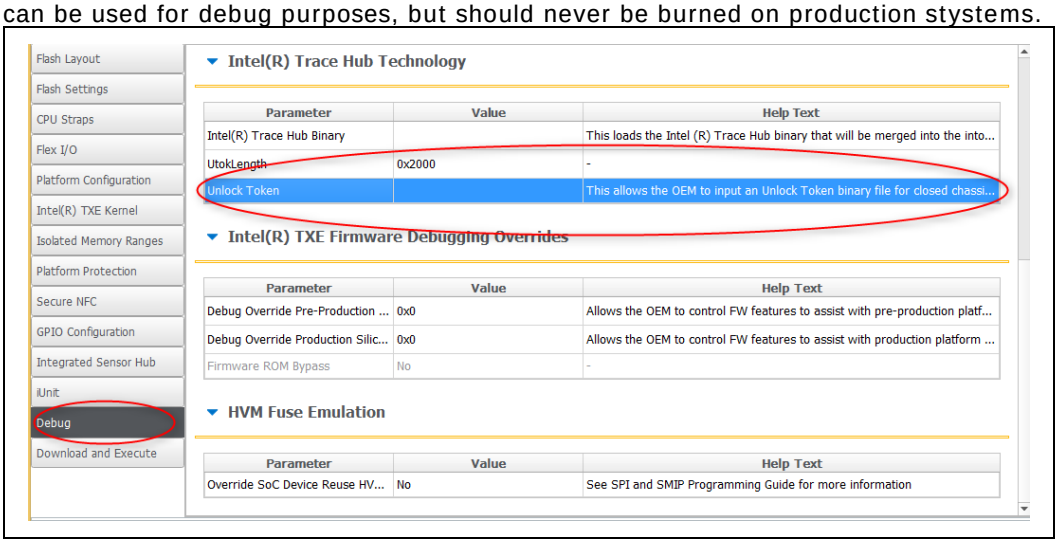
The errors returned include

Error	Description
Invalid Public Key	Pub key hash in token does not match public key hash in OEM Key Manifest
Authentication failure	The token data authentication did not match declared information in the token manifest
Invalid Unit ID	The token is designated to a different platform, with a different Part ID
Invalid Lifecycle Stage	The value set for the lifecycle stage is incorrect
Token expired	Token time has expired

4.2.3 Building a Token into the Firmware Image

The OEM Unlock Token can be compiled directly into the firmware image when it is built, using FIT.

It is entered in the Debug tab, in the Unlock Token field. An image prepared this way can be used for debug purposes, but should never be burned on production systems.



4.3 Clearing of Token

The Device Lifecycle token is removed when the system is next rebooted.

The OEM Unlock token survives a reboot, and must be erased using the FPT or DnX APIs described above. Intel recommends never releasing to customers a platform with an erased OEM Unlock Token, but to re-flash the full firmware image instead.

4.4 Debugging Secure Token Injection

The Device Lifecycle token is injected using DnX APIs. These APIs return an error code, which can indicate if a token was rejected, and why. These are explained in Section 4.2.2.

The OEM Unlock Token is only examined by firmware at system boot, and so the injection API cannot return any failure codes. In the event that the token is failing to unlock the platform, North Peak messages must be examined, as they indicate why a token was rejected.

§§