

# **APL and GLK Secure Tokens**

**User Guide** 

**Revision 1.1** 

November 2017

**Intel Confidential** 



You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at Intel.com, or from the OEM or retailer.

No computer system can be absolutely secure. Intel does not assume any liability for lost or stolen data or systems or any damages resulting from such losses.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or visit www.intel.com/design/literature.htm.

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel, the Intel logo, and the trademarks of Intel Corporation in the U.S. and/or other countries.

\*Other names and brands may be claimed as the property of others.

Copyright © 2017, Intel Corporation. All rights reserved.



# **Contents**

1	Introdu	uction	5
	1.1 1.2	Goal Prerequisites	5 5
	1.3 1.4	Tools Used in this Document Terminology	5 5
2	Overvie	ew of Secure Tokens	7
	2.1	Introduction	7
	2.2	Preparing the Platform to Accept Secure Tokens	7
3	Creatio	n of Secure Tokens	8
	3.1	Introduction	8
	3.2	Installing Intel® PFT	8
	3.3	Launching Intel <sup>®</sup> PFT Token Module	8
	3.4	Set General Settings	9
	3.5	Creation the Token 1	11
4	Injectio	on of Token on Platform 1	15
	4.1	Introduction 1	15
	4.2	Injection 1	15
		4.2.1 Injection using Intel <sup>®</sup> FPT 1	15
		4.2.2 Injection Using DnX 1	15
		4.2.3 Building a Token into the Firmware Image 1	17
	4.3	Clearing of Token 1	17
	4.4	Debugging Secure Token Injection 1	8

#### Table

Table 1-1.	Terminology					5
------------	-------------	--	--	--	--	---



Revision Number	Description	<b>Revision Date</b>
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<sup>®</sup> Platform Flash Tool (PFT)
- Mobile Signing Utility for GLK
- Intel<sup>®</sup> 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<sup>®</sup> Platform Plash Tool (PFT)
- Intel<sup>®</sup> Flash Programming Tool (FPT)
- Intel<sup>®</sup> Manifest Extension Utility (MEU)

#### 1.4 Terminology

#### Table 1-1. Terminology

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



Term	Description
IBB	Initial Boot Block
IBBL	Initial Boot Block Loader
IFWI	Integrated Firmware Image
ISH	Integrated Sensor Hub
ОВВ	OEM Boot Block
SUT	System Under Test
ОЕМ КМ	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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> PFT

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

### 3.3 Launching Intel<sup>®</sup> PFT Token Module

1. Open Intel<sup>®</sup> 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<sup>®</sup> 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 <u>Section 2.2</u>



General Settings ? ×
Signing Device Connection
Signing method
◯ Remote HIM
Signing keys
Key file:
Password:
Secure Provisioning
Online RSA Public key file:
Online RSA Private key file:
Offline RSA Public key file:
Offline RSA Private key file:
Anti-Cloning
Public OEM ID
Browse or Generate from Private entries
Variant ID
OK Cancel

- 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.



B		General S	ettings		? ×
Signing	Device Connection				
For	e to use DnX mode (use	d for data retrieval, tok	en provisioning/erasure)	)	
Firmwa	re DnX module:				
✓ Autor	natically get device data	when generating token	S		
		OK C	ancal		
		OK C			

### 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.





G	New token	?	×
1. Select platform:	Broxton		•
2. Select Token template:	Pick a token		-
	Pick a token		
	IDLM Unlock Token		- 11
	OEM Unlock Token		
	PAVP Debug Unlock Token		
	VISA Override Token		
	Device Lifecycle token		
	PPV		

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<sup>®</sup> 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.

<b></b>			Intel® Phone	Flash Tool 5.3.2.0		- • • • •
<u>File Security</u>	Help					
1	OEM Unlock Token 🔀			0	New	Open
Flash	Manifest Extension:				Save	Save As
					Device selection	
	Flags:	12	Edit flags			•
Flash editor	Expiration (seconds):	Osec.	÷			
	Type:	13			Generate & Sign	
Š.	Manufacturing lot:	0			📱 Get Devi	ice Data
Security	Part ID:				Click on button below token:	to generate a full signe
	Part ID:				Activate Anti-Clonin	g
	Nonce: 000000	000	Hexa 💌		🔒 Genera	te & Sign Token
Provision	Time base: 000000	000	Hexa 💌		Write / Dead / Eraco	
	Daudaada				Write a token to the de	vice:
	Payload:					Write
	Knobs:				Frase or Read a token i	from the device
	OEM Unlock Allow VISA Override ISH GDB Debug Disable BIOS Secure Boot Disable Audio FW authentical di				Token ID:	i on the device.
	Activated				lite David	
	ID: 80860002				Mead	🍾 Erase
	Value: 00000000	Edit			More	
Log lovel						
Logievei	12/16/15 14:27:51.281 IN	FO : Using Android	Debug Bridge version 1.0.32			
IFO 👻	Revision cc6ae925e394-an 12/16/15 14:27:51.281 IN	droid FO : OS version de	tected: Windows 8.1 (x64)			
	12/16/15 14:27:52.367 IN 12/16/15 14:27:52.367 IN	FO : SSL supported FO : Using TMT API	: Yes version: 1.8.0-0			
Save Log	12/16/15 14:27:52.456 IN 12/16/15 14:27:52.594 IN	FO : Disk space ava FO : Adb and fastb	ilable on drive C:/ : 46.83 GB oot binaries OK			
\$	12/16/15 14:27:53.137 IN 12/16/15 14:28:25.680 W	FO : Phone Flash To ARNING: [Security]	ool initialized successfully ocal Password is empty!			
	12/16/15 14:28:25.680 W	ARNING: [Security] F	W DnX module File is empty!			

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.
	<ol> <li>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.</li> <li>Secure boot disabled. All secure boot flows are disabled. Actually, this means that the BIOS BPM is not validated.</li> </ol>



# 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<sup>®</sup> FPT

The OEM Unlock Token can be injected into a platform using Intel<sup>®</sup> 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<sup>®</sup> 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></file>
Delete the token for the token ID provided	FPT.exe - ERASETOKEN <pid></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.

Operation	Command Line
Read token in slot 0 to read_token.bin	dnxFwDownloader.execommand readtokenfw_dnx DNXP_0x1.binpath read_token.binslot 0
Write token OEMUnlock.bin to slot 0	dnxFwDownloader.execommand writetokenfw_dnx DNXP_0x1.bintoken OEMUnlock.binslot 0
Erase token in slot 0	dnxFwDownloader.execommand erasetokenfw_dnx DNXP_0x1.binslot 0
Set Device Lifecycle stage, using token DeviceLifecycle.bin	dnxFwDownloader.execommand setdevicelifecyclestage fw_dnx DNXP_0x1.bin -token DeviceLifecycle.bin

The DnX token API is supported by the Intel<sup>®</sup> PFT command line:

*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:





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 I mage

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 stystems.

cayout		echnology	
Settings			
Straps	Parameter	Value	Help Text
	Intel(R) Trace Hub Binary		This loads the Intel (R) Trace Hub binary that will be merged into the into
0	UtokLength	0x2000	-
m Configuration	Unlock Token		This allows the OEM to input an Unlock Token binary file for closed chassi.
() TXE Kernel			
d Memory Ranges	<ul> <li>Intel(R) TXE Firmwar</li> </ul>	e Debuaalna Overri	les
d Memory Ranges	<ul> <li>Intel(R) TXE Firmwar</li> </ul>	e Debugging Overri	des
d Memory Ranges m Protection	Intel(R) TXE Firmwar	e Debugging Overrie	Help Toxt
ed Memory Ranges m Protection	Intel(R) TXE Firmwar      Parameter      Debug Override Pre-Production	e Debugging Overrie Value	Help Text Allows the OEM to control FW features to assist with pre-production platf
d Memory Ranges m Protection PNFC Configuration	Intel(R) TXE Firmwai      Parameter      Debug Override Pre-Production      Debug Override Production Silic	value Value 0x0	Help Text Allows the OEM to control FW features to assist with pre-production platf Allows the OEM to control FW features to assist with production platform
d Memory Ranges m Protection e NFC Configuration ated Sensor Hub	Intel(R) TXE Firmwai     Parameter     Debug Override Pre-Production     Debug Override Production Silic     Firmware ROM Bypass	e Debugging Overrie Value 0x0 0x0 No	Help Text Allows the OEM to control FW features to assist with pre-production platf Allows the OEM to control FW features to assist with production platform
d Memory Ranges m Protection e NFC Configuration ated Sensor Hub	Intel(R) TXE Firmwai     Parameter     Debug Override Pre-Production     Debug Override Production Silic     Firmware ROM Bypass	value Value 0x0 0x0 No	Help Text Allows the OEM to control FW features to assist with pre-production platf Allows the OEM to control FW features to assist with production platform
d Memory Ranges m Protection e NFC Configuration ated Sensor Hub	Intel(R) TXE Firmwai     Parameter     Debug Override Pre-Production     Debug Override Production Silic     Firmware ROM Bypass     HVM Fuse Emulation	value Value 0x0 0x0 No	Help Text Allows the OEM to control FW features to assist with pre-production platf Allows the OEM to control FW features to assist with production platform
d Memory Ranges m Protection e NFC Configuration ated Sensor Hub	Intel(R) TXE Firmwar     Parameter     Debug Override Pre-Production     Debug Override Production Silic     Firmware ROM Bypass     HVM Fuse Emulation     Parameter	e Debugging Overrie Value 0x0 0x0 No Value	Help Text Allows the OEM to control FW features to assist with pre-production platf Allows the OEM to control FW features to assist with production platform Help Text

### 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 <u>Section 4.2.2</u>.

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.