

Raptor Lake DnX (Download and Execute)

User Guide

October 2021

Revision 0.8

Intel Confidential



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

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

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document 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.

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

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: <http://www.intel.com/design/literature.htm>.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

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

Copyright © 2021, Intel Corporation. All rights reserved.



Contents

1	Introduction.....	6
	1.1 Terminology.....	6
2	Intel® Download and Execute (Intel® DnX).....	7
	2.1 Introduction.....	7
	2.2 Use Cases.....	8
	2.3 Triggers.....	9
3	Intel® DnX Requirements.....	10
	3.1 Tools and Files.....	10
	3.2 How to enable Intel® DnX on the Platform.....	12
	3.2.1 During Manufacturing or Before End-of-Manufacturing..	12
	3.2.2 End-of-Manufacturing.....	12
	3.3 How to build an Intel® DnX Image for Manufacturing/Refurbish use cases.....	16
	3.3.1 Preparing the Target Platform to receive an Intel® DnX- enabled image.....	16
4	Usage.....	21
	4.1 Host and Target Setup.....	22
	4.1.1 Intel® Platform Flash Tool (PFT) Overview.....	22
	4.2 Image Recovery/Programming.....	25
	4.2.1 Using Intel® PFT GUI.....	25
	4.2.2 Using Command Line.....	30
	4.3 OEM Debug Tokens.....	33
	4.3.1 Using Intel® PFT GUI.....	33
	4.3.2 Using Command Line.....	36
	4.3.3 Common error messages.....	38
5	Opening Intel® DnX capabilities post EOM.....	41
	5.1 Flow to open prohibited post EOM Intel® DnX capabilities.....	42



5.2	Preparing the OEM token for Intel® DnX capabilities using Intel® PFT GUI.....	42
5.3	Preparing the OEM token for Intel® DnX capabilities using Intel® PFT CLI.....	42
5.4	Downloading OEM Key Manifest and Set Capabilities	43
6	References	45



Revision History

Revision Number	Description	Revision Date
0.5	<ul style="list-style-type: none">• Initial release	May 2021
0.8	<ul style="list-style-type: none">• Updated revision to 0.8 for Alpha	October 2021

§ §



1 Introduction

The purpose of the document is to provide guidance on Intel® Download and Execute (Intel® DnX) feature, usage of this feature and how it gets enabled using Intel® CSE components as well as Intel® Platform Flash Tool (PFT).

1.1 Terminology

Acronym or Term	Definition
Intel® CSE	Intel® Converged Security Engine
Intel® DnX	Intel® Download and Execute
Intel® FIT	Intel® Flash Image Tool
FW	Firmware
Intel® PFT	Intel® Platform Flash Tool
Target	System Under Debug or Target Platform
OEM	Original Equipment Manufacturer
Intel® FPT	Intel® Flash Programming Tool



2 Intel® Download and Execute (Intel® DnX)

2.1 Introduction

“DnX” is Intel’s proprietary solution to download FW module to a target machine from a host machine by means of USB cable and execute it. Intel® DnX flows are executed over fixed USB 2.0 port. On SPI platforms, this capability allows to access boot media for IFWI write as well as signed Token injection for debug unlock after the platform have completed manufacturing.

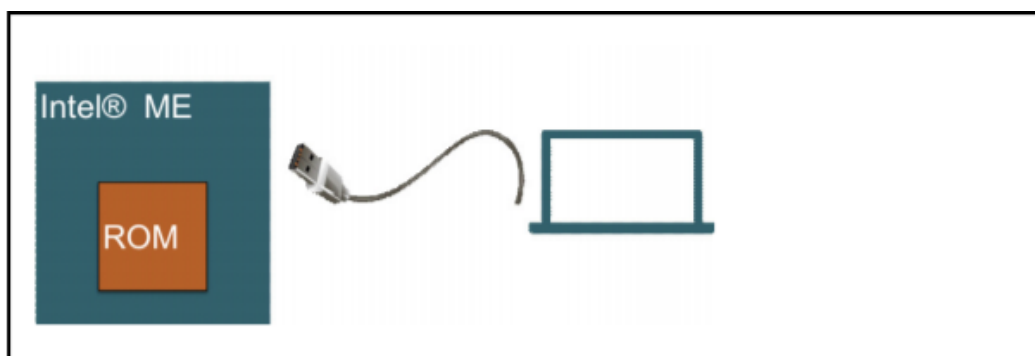


Figure 1

There are many advantages in using Intel® DnX, one of which is the capability to flash IFWI without opening the chassis to physically reach the flash device. This is especially important in the manufacturing line once the chassis is closed, in debug labs and in post manufacturing scenarios. Other methods such as FWUpdate and Capsule flow can be used on subsequent upgrades/downgrades if the platform can boot to an operating system. Intel® DnX is a capability in Intel® CSE ROM, where during the boot ROM can configure the USB port #0 on the PCH to connect to a remote computer to download Intel® DnX module which is signed by Intel. This module initiates rest of the Intel® CSE and sets up an environment to accept IFWI binary or secure token from a remote computer. The flow is explained below:

Flashing IFWI into NVM

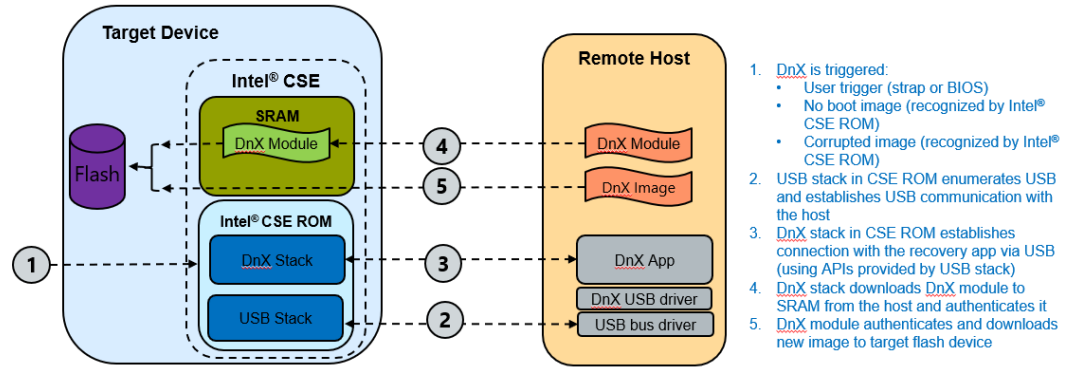


Figure 2

Injecting Secure Token for Debug

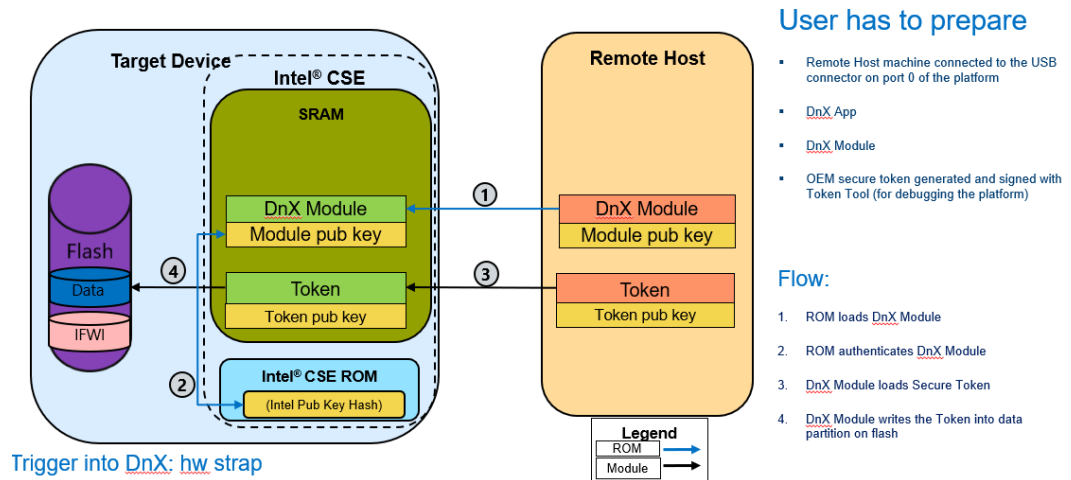


Figure 3

2.2

Use Cases

Below use-cases are supported on RAPTOR LAKE platform.

Scenario	Use Case
Manufacturing and refurbishing	<ul style="list-style-type: none"> • Full IFWI.bin programming into SPI boot media • Read content of SPI boot media



Scenario	Use Case
Debug	<ul style="list-style-type: none">Write/Read/Erased signed debug tokens into the platform

2.3

Triggers

Following methods can be used to enter Intel® DnX on the platform.

Method	Detail	Use Case
Pin Strap	<p>GPIO: GP_C08 (signal name DNX_FORCE_RELOAD) can be used to trigger Intel® DnX on the platform.</p> <p>Important : For details on Hardware trigger method, please refer to “Raptor Lake Lake External Design Specification (EDS), Volume 1 (RDC# 601458)”. Also check with your EC Customer Enablement representative on EC Firmware requirements.</p>	<p>For Injecting OEM debug tokens for non-booting scenarios due to Boot Guard failures</p> <p>OR</p> <p>For Image Recovery/Update</p>
Intel® CSE or BIOS Error Handling Flow	If Intel® CSE or BIOS reach a critical error which prevent platform from booting, it can be programmed to enter Intel® DnX flow. (e.g. Failure to authenticate BIOS signature, CSE detect FW corruption, etc.)	For Image Recovery/Update
Empty Flash Device	When CSE ROM detects an empty flash device on the platform, it will enter Intel® DnX mode	For Image Recovery/Programming



3 Intel® DnX Requirements

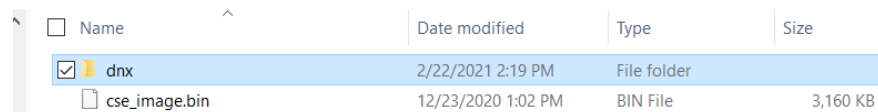
To use Intel® DnX technology on a target platform, the following are the requirements

1. Set the Intel® DnX fuse – covered in Chapter 3
2. Method to Enter Intel® DnX mode – covered in Section 2.3
3. Operations in Intel® DnX mode,
 - a. Perform Image recovery or programming of boot media
 - b. Interact with target to inject OEM debug token to enable debug features.Both a and b are covered in Chapter 4 and 5

3.1 Tools and Files

Following tools and files are applicable for Intel® DnX:

- **Intel® PFT** (Platform Flash Tool) – Intel implementation of Intel® DnX tool running on remote host computer. Intel® DnX module, config.xml and IFWI.bin are inserted to the target machine via this tool. Will be included in the Intel® CSE Kit for RAPTOR LAKE platform.
- **Intel® DnX Module** - binary file signed by Intel. This file has the Intel® DnX logic Intel® CSE ROM will run. Will be included in the Intel® CSE kit for RAPTOR LAKE Platform in the “DnX” folder



<input type="checkbox"/> Name	Date modified	Type	Size
<input checked="" type="checkbox"/> dnx	2/22/2021 2:19 PM	File folder	
<input type="checkbox"/> cse_image.bin	12/23/2020 1:02 PM	BIN File	3,160 KB

Figure 4

Please link this file to the Intel® PFT tool as shown in Chapter 4

- **Intel® FIT** – can be used to create Intel® DnX based IFWI image. This is covered in Section # 3.2 of this document.
- **Intel® FPT** - can be used to configure Intel® DnX fuse and close manufacturing on the platform. Will be included in the Intel® CSE kit for RAPTOR LAKE Platform. Please note, this setting is a must to have Intel® DnX enabled in your platforms.



Intel® FIT and Intel® FPT are released in the CSME firmware kits under System Tools.

_15.40.10.2204 > Tools > System_Tools

<input type="checkbox"/> Name	Date modified	Type
Documentation	2/22/2021 2:19 PM	File folder
<input checked="" type="checkbox"/> FIT	2/22/2021 2:19 PM	File folder
<input checked="" type="checkbox"/> FPT	2/22/2021 2:19 PM	File folder
FWUpdate	2/22/2021 2:19 PM	File folder
FWUpdate_RS	2/22/2021 2:19 PM	File folder
ICC Tools	2/22/2021 2:19 PM	File folder
MEInfo	2/22/2021 2:19 PM	File folder
MEManuf	2/22/2021 2:19 PM	File folder
MEU	2/22/2021 2:19 PM	File folder

Figure 5

The Intel® DnX module can be found in CSME kit in the “dnx” folder



3.2 How to enable Intel® DnX on the Platform

3.2.1 During Manufacturing or Before End-of-Manufacturing

Intel® DnX is enabled in the Image by default and all Intel® DnX operations such as Image Recovery/Update or OEM Debug token injection using Intel® DnX are available to use.

3.2.2 End-of-Manufacturing

If Intel® DnX is disabled in the fuse at the time of End-Of-Manufacturing, NO Intel® DnX operations are available after End-Of-Manufacturing. This includes OEM Debug token injection and Image Recovery/Programming operations.

3.2.2.1 Setting Intel® DnX Fuse

3.2.2.1.1 Enabling Intel® DnX operations after End-Of-Manufacturing

In-order for the Target to enter Intel® DnX Mode **after** End-of-Manufacturing, it is important that the Intel® DnX fuse set to YES in the End-Of-Manufacturing process. This can be done by using the Intel® FIT's configuration as below.

The screenshot shows the Intel Hash Image Tool interface. The top menu bar includes File, Build, and Help. Below the menu bar, there are icons for file operations and a dropdown menu showing 'Intel(R) ElkhartLake Chipset'. To the right of the dropdown are 'EHL No Emulation' and 'Target Type SPI'.

The left sidebar contains a list of settings categories: Flash Layout, Flash Settings, Intel(R) ME Kernel, Platform Protection, Integrated Clock Controller, Networking & Connectivity, Internal PCH Buses, Power, Debug, Compute Die Straps, Flex I/O, GPIO, Download and Execute (highlighted), and FW Update Image Build.

The main content area is divided into two sections: 'DnX Image' and 'DnX Fuses'.

DnX Image

Parameter	Value	
Platform ID	0x0	DnX Image attribute. Ignored before FPFs lock. After FPFs lock, D
BuildEnabled	No	Should Intel FIT build a DnX image
OutputFileName	\$DestDir\dnx.bin	-
DnX image private sign...		The path to the private key to use to sign the DnX image. This settin

DnX Fuses

Parameter	Value	
DnX Enabled	Yes	DnX permanent enable/disable FPF
OEM Platform ID	0x0	This setting allows OEMs to configure a Unique Platform ID into the



Figure 6

Thus,

1. Intel® FIT Tool-> Download and Execute-> DnxEnabled – set to 'YES'
2. Then perform End-Of-Manufacturing using Intel® FPT

Intel® DnX technology is now enabled via the fuses and available to use.

Note : Certain Image recovery/Programming operations are prohibited after End-Of-Manufacturing and can Only be enabled via OEM Debug token as shown in the table below. OEM Debug Tokens are covered in the Intel® Secure Debug Token Guide available in the CSME FW kits. Also refer to Section 5 for details in this document.

Use Case	Intel® DnX Operation	Pre-End-Of-Manufacturing	Post End-Of-Manufacturing with FIT fuse DnxEnabled = 1	Can OEM Debug token enable ?
OEM Debug Token	Get Part ID for tokens	Allowed	Allowed	N/A
OEM Debug Token	Write/Inject token	Allowed	Allowed	N/A
OEM Debug Token	Read token	Allowed	Allowed	N/A
OEM Debug Token	Erase token	Allowed	Allowed	N/A
Image Recovery or Programming	Read Boot Media Content	Allowed	Prohibited	Yes
Image Recovery or Programming	Provision/Write Firmware image	Allowed	Prohibited	Yes
Image Recovery or Programming	Get NV Store Info	Allowed	Prohibited	Yes
OEM Debug Token, Image Recovery or Programming	Start Over to reset the platform	Allowed	Allowed	N/A
OEM Debug Token, Image Recovery or Programming	Ping the device for its ID	Allowed	Allowed	N/A
Image Recovery or Programming	Download Recovery Module	Allowed	Allowed	N/A



Use Case	Intel® DnX Operation	Pre-End-Of-Manufacturing	Post End-Of-Manufacturing with FIT fuse DnXEnabled = 1	Can OEM Debug token enable ?
Image Recovery or Programming	Open Capabilities which are prohibited	Allowed	Allowed	N/A

After image has been provisioned, the flash layout is unknown until the NVM device initialization is performed. Therefore, until a global reset occurs, operations that require knowledge of the layout (read/write/erase token) will return an error: 0x80000058 (ENVN_REQUIRES_REINIT).

3.2.2.2 Disabling Intel® DnX operations after End-Of-manufacturing

If the Intel® DnX Fuse is disabled in the Intel® FIT after End-of-Manufacturing, NO Intel® DnX operations are available.

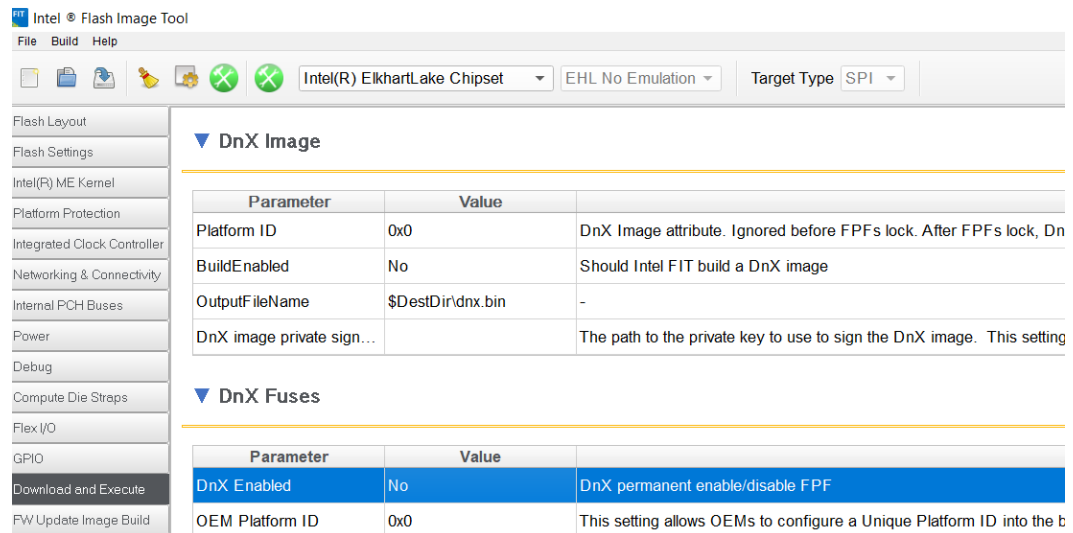


Figure 7

Thus,

1. Intel® FIT Tool-> Download and Execute-> DnxEnabled – set to 'NO'
2. Then perform End-Of-Manufacturing using Intel® FPT

All DnX operations are DISABLED as shown below after End-Of-Manufacturing.



Use Case	Intel® DnX Operation	Pre-End-Of-Manufacturing	Post End-Of-Manufacturing with DnXEnabled = 0	Can OEM Debug token enable ?
OEM Debug Token	Get Part ID for tokens	Allowed	Disabled	N/A
OEM Debug Token	Write/Inject token	Allowed	Disabled	N/A
OEM Debug Token	Read token	Allowed	Disabled	N/A
OEM Debug Token	Erase token	Allowed	Disabled	N/A
Image Recovery or Programming	Read Boot Media Content	Allowed	Disabled	N/A
Image Recovery or Programming	Provision/Write Firmware image	Allowed	Disabled	N/A
Image Recovery or Programming	Get NV Store Info	Allowed	Disabled	N/A
OEM Debug Token, Image Recovery or Programming	Start Over to reset the platform	Allowed	Disabled	N/A
OEM Debug Token, Image Recovery or Programming	Ping the device for its ID	Allowed	Disabled	N/A
Image Recovery or Programming	Download Recovery Module	Allowed	Disabled	N/A
Image Recovery or Programming	Open Capabilities which are prohibited	Allowed	Disabled	N/A

3.2.2.2.1 Firmware Settings - Important Note for Raptor lake

Intel® FIT Tool-> Download and Execute-> DnxEnabled is set to 'NO' by DEFAULT in CSE Firmware Kits.

Note : If Customer wants to have Intel® DnX operation enabled after End-Of-Manufacturing, they MUST set it to 'YES' as described in section 3.1.2.1.1



To keep Intel® DnX operations always available before End-Of-Manufacturing, please always keep this setting as 'YES'. Failing to do so will disable Intel® DnX fuse on their platform and no Intel® DnX operations will be available. There are no functional changes or restrictions for Intel® DnX related to this change, the intent is to reduce the attack surface of platforms that are being delivered with default values by closing interfaces that aren't in use.

Customers that use Intel® DnX in manufacturing line **should** have "DnX enabled" set to YES in their configuration along with settings in Section 3.3 – and then everything works as usual.

3.3 How to build an Intel® DnX Image for Manufacturing/Refurbish use cases

If customer is using Intel® DnX technology for Manufacturing or Refurbishing use case of programming an image on the SPI or reading back an existing image or even writing a new recovery/updated image on the SPI, then the Firmware image (IFWI) needs to be built with this information.

If customer is using Intel® DnX technology only for OEM Debug tokens, this section can be skipped.

3.3.1 Preparing the Target Platform to receive an Intel® DnX-enabled image

An Intel® DnX-enabled image is an image which will be programmed into the SPI using Intel® DnX technology. The image must be signed with the appropriate Private key owned by the OEM. This section lists some of the key points to note when building an Intel® DnX-enabled image.

3.3.1.1 OEM Key Manifest

Raptor Lake 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 Intel® DnX image. This should be populated with the hash of the public key, matching the private key with which the Intel® DnX-image will be signed. If there's a mismatch of the keys, the Intel® DnX-image will not be accepted by the target.

The string to look for Intel® DnX in the OEM Key Manifest is "OEMDnxIfwiManifest" as shown below :



```
<versionnotrix value="0x0000" help_text="Indicates the notrix number in the version n
<VersionBuild value="0x0000" help_text="Indicates the build number in the version num
<KeyManifestEntries>
  <KeyManifestEntry>
    <Usage value="OemDnxIfwiManifest" value_list="BootPolicyManifest,,iUnitBootLo
    <HashBinary value="C:\Users\...\Desktop\...\OpenSSL-Win32\bin\pubkey3khash.bi
  </KeyManifestEntry>
  <KeyManifestEntry>
    <Usage value = "IshManifest" value_..
    <Hash binary value = "hash to sign Ish"
  /
</KeyManifestEntries>
</KeyManifest>
```

Figure 8

An overview of the signing and manifesting process – how to generate a signing key and how to fill in the OEM Key Manifest is described in “Raptor Lake Signing and Manifesting Guide” included in the CSME Firmware kit.

This part of the process is like any other OEM owned and signed component such as OEM token or ISH. The OEM owns the security of the private keys for any OEM owned components.

3.3.1.2 Tool Settings – Intel® FIT

Use the Intel® FIT to create an Intel® DnX-enabled image.

3.3.1.2.1 Platform Protection

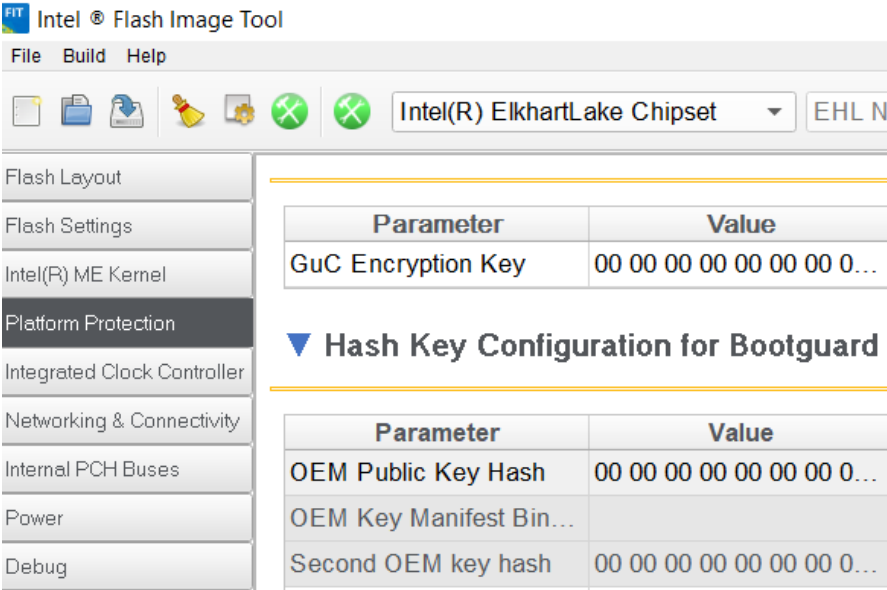


Figure 9

- Intel® FIT Tool-> Platform Protection-> Hash Key Configuration For Boot Guard -> OemPublicKeyHash – add public key has of the private key which will be used to sign Intel® DnX IFWI image
- Intel® FIT Tool-> Platform Protection-> Hash Key Configuration For Boot Guard -> OEM Key Manifest Binary – path to OEM key Manifest

3.3.1.2.2

Download and Execute

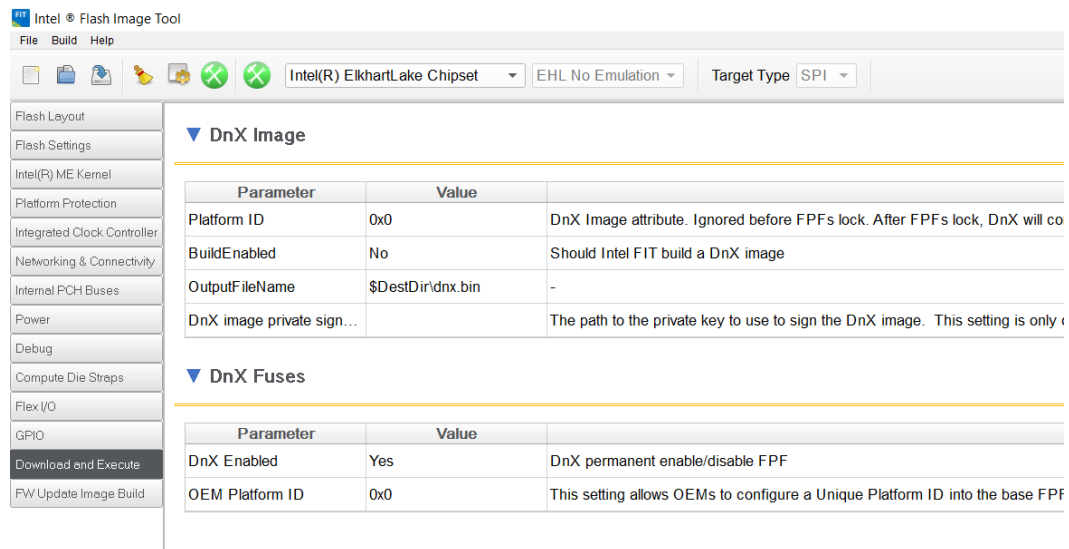


Figure 10

Please update below parameters under Intel® FIT tool to create IFWI image for Intel® DnX flash process.

- Intel® FIT Tool-> Download and Execute-> DnxEnabled – set to ‘Yes’
- Intel® FIT Tool-> Download and Execute-> BuildEnabled – set to ‘Yes’
- If Customer is using Intel® FIT tool to sign this Intel® DnX-image, Intel® FIT Tool-> Download and Execute-> SigningKey – path to private key to be used. Check settings in Section 3.3.1.2.3 as well in this case for signing tool details.
- Intel® FIT Tool-> Download and Execute->Outputfilename – Path and name of the Intel® DnX image
- Intel® FIT Tool-> Download and Execute-> Platform ID- Platform ID that Intel® DnX uses to verify image is suitable for the platform; before EOM flow, this value can be left to default.



- Intel® FIT Tool-> Download and Execute->OEM ID that Intel® DnX uses to verify image is suitable for the platform; before EOM flow, this value can be left to default.

3.3.1.2.3 Build Settings

The Build Settings is located on the top as shown with 1 in the figure below

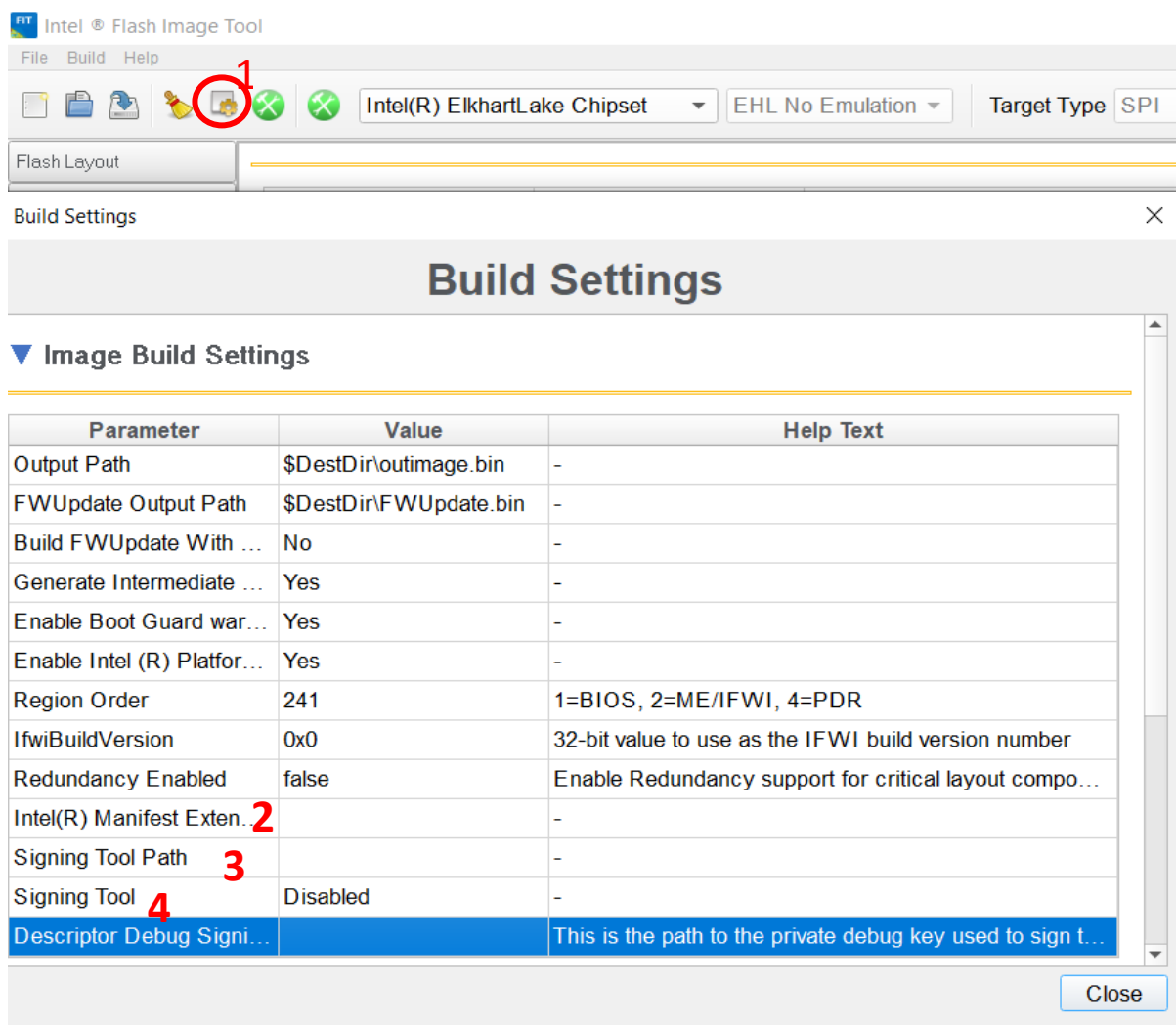


Figure 11

- Intel® FIT Tool -> Build->Build Settings-> Intel® Manifest Extension Utility Path – path to Raptor Lake Lake Intel® MEU tool; available within Intel® CSE Kit. Shown in #2
- If customer is using Intel tool to sign this Intel® DnX image,



Intel® FIT Tool -> Build-> Build Settings-> SigningToolPath – path to Open SSL tool (from SSL installation directory) as shown in #3 above

- Select “Signing Tool” as OpenSSL if OEM is using Intel tool to sign the Intel® DnX image. As shown in #4 above. Check section 3.3.1.2.1 as well for Private key setting.

Note, customer can use their own tool to sign the Intel® DnX image as well. In which case, select #4 in above figure as “Disabled”. This will not sign the Intel® DnX-image.



4 Usage

This chapter describes the set up for Intel® DnX and usage in both GUI and command line operations.

The main operations using Intel® DnX are –

1. OEM Debug tokens

Intel® DnX technology is used to write the OEM debug token to the target platform.

The OEM debug token allows to re-enable debug features on the platform without having to open the chassis. These include enabling CSME traces, enabling Debug Interfaces like USB2.Dbc, passing BIOS payload to customer BIOS and OEM unlock for IPs that are customer-debuggable.

In-order to also update the SPI after End-Of-Manufacturing, an OEM Debug token to re-enable these Capabilities using the “Set Capabilities” knob of the OEM token is required.

Full details of OEM Debug token are captured in the Secure Token’s Guide released in the CSME kit. In this document, we only cover the below using Intel® DnX technology.

- a. Read Part ID
- b. Inject the token
- c. Erase the token after use
- d. Read the token, if needed.

2. Image Recovery/Programming

During Manufacturing or Post End-of-Manufacturing

- a. Program image on blank SPI
- b. Re-flash new/recovery image in case of corrupted image
- c. Re-flash recovery image on an End-Of-Manufacturing-enabled system for Refurbish

4.1 Host and Target Setup

DnX Test Setup

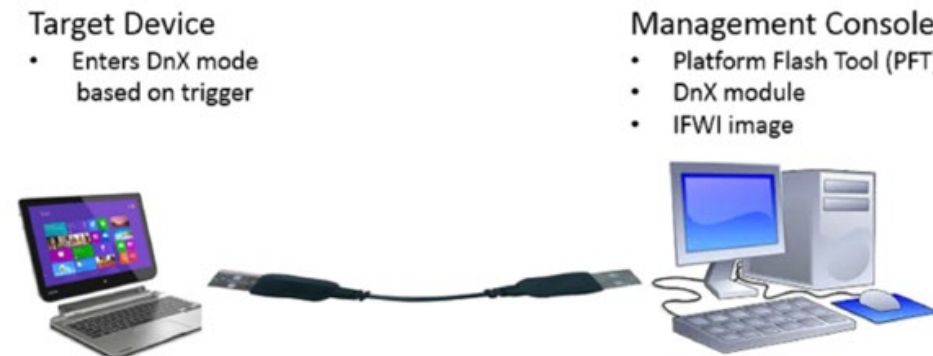


Figure 12

Requirement	Usage
Management Console / Host	A host that can be used to execute the Intel® DnX flows
Hardware Connection	USB cable connection between the Remote Host to the system under test
Intel® Platform Flash Tool (PFT)	Tool supporting Intel® DnX flows running on the Remote Host
Intel® DnX module binary	Provided in the Intel® CSE FW kit. This binary is provided as an input to the Intel® PFT.
Intel® DnX based IFWI image	IFWI image to be flashed on the target system (Can be created by Intel® FIT tool provided in the Intel® CSE)
OEM signed Debug token	Token binary to be flashed on the target system (Can be created by Intel® PFT tool provided in the Intel® CSE)

4.1.1 Intel® Platform Flash Tool (PFT) Overview

Intel® Platform Flash Tool supports GUI (Graphic User Interface) as well as CLI (Command Line Interface) and runs on the Host.

This tool supports Intel® DnX flows and consumes Intel® DnX related input files like: Intel® DnX module, secure token file to be flashed on the target system.



Please install this tool on Host system before executing Intel® DnX flows.

4.1.1.1

GUI

PFT tool is available within Intel® CSE FW Kit->Tools->DnX Tools. See example below

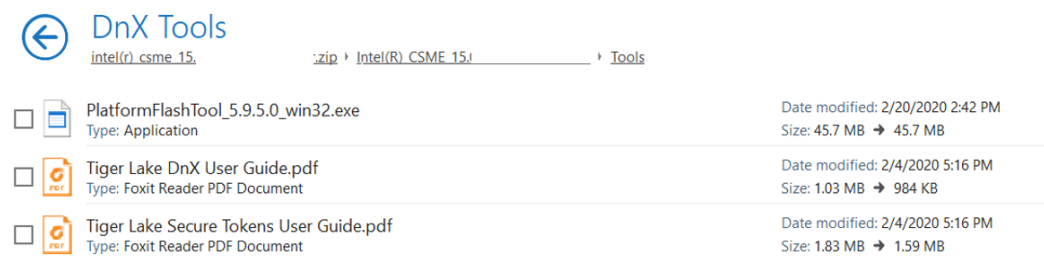


Figure 13

Run the installation package. Setup wizard will start. Click “Next” to complete the installation.

This installation process installs Tools as well as necessary USB drivers along with it as well.



Figure 14

Once PFT tool is installed on the Host:

- Make sure the target system is connected to the Host using USB cable.

- Make sure all input files required for Intel® DnX operation (e.g. Intel® DnX module, token) are available on the Host.
- Ensure that the Intel® DnX module from the CSME Firmware kit is linked to the Intel® PFT tool as shown below

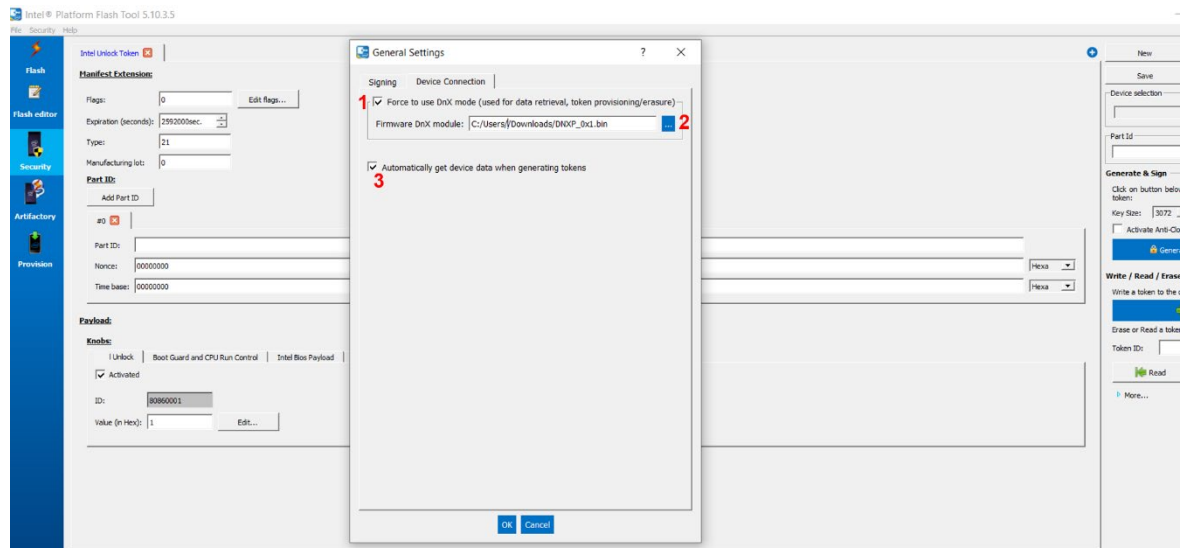


Figure 15

- Make sure the target system is in Intel® DnX mode by checking the “Device Manager” of the Host under “Universal Serial Bus device” should show Intel® DnX device

4.1.1.2

CLI - Intel® DnX Firmware downloader

Intel® DnX Firmware Downloader is the name of the executable that provides command line interface for Intel® DnX to interact with Intel® CSE firmware and perform different Intel® DnX operations. This is also installed by the Intel® PFT at the path –

“C:\Program Files (x86)\Intel\Platform Flash Tool”

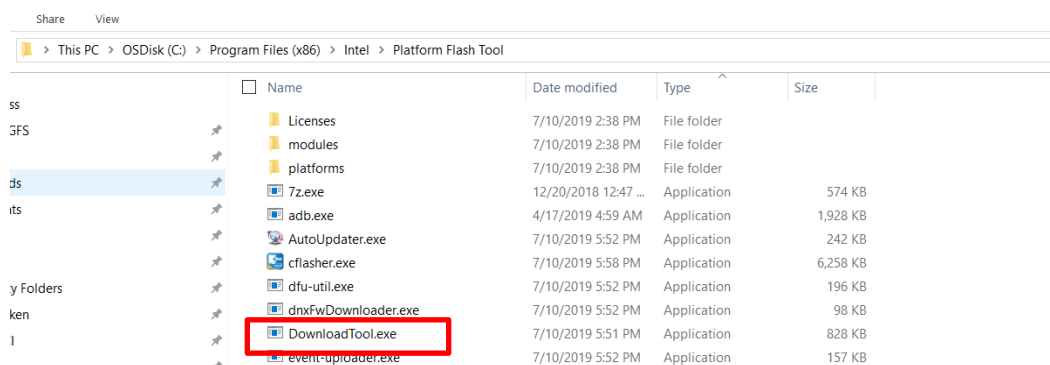


Figure 16

This tool supports serial number argument, however, does not provide USB port hence less convenient for setups with multiple targets connected to one remote host at the same time.

Note: target machine must already be in Intel® DnX mode when running Intel® DnX operations (e.g. jumper, virgin storage)

4.2 Image Recovery/Programming

4.2.1 Using Intel® PFT GUI

4.2.1.1 Executing json file containing Intel® DnX commands

In-order to execute json file with Intel® DnX commands:

1. Choose “Flash” tab on the left panel of PFT
2. Use “Browse” button to load desired json file
3. Choose *.json file which has set of attributes defined to perform Intel® DnX use cases.
4. Click on “Start Flash” Tab.

Note: User is expected to update this sample file with appropriate naming and path details of Intel® DnX module and Intel® DnX IFWI binary.

4.2.1.1.1 Creating json file for flashing Intel® CSE IFWI through Intel® DnX

In-order to create json file supporting IFWI flash command via Intel® DnX from scratch, follow those steps:

Open PFT GUI

1. Choose “**Flash editor**” tab on the left panel of PFT

2. Click **“Reset Default”** button if the flash editor contains any data
3. Set **“Initial board state”** to **“DNX FW”**
4. Set **“Predefined sequence”** to **“Flash FW with DnX FW Downloader”** and click **“Add”**
5. After clicking **“Add”**, a new Window will open:
 - a. In the **“Firmware DnX”** field, select the path to the Intel® DnX module. The Intel® DnX module is part of the IFWI kit, provided by Intel (Usually named **“dnxp_0x1.bin”**)
 - b. In the **“Firmware Image”** field, select the path to the IFWI image you would like to flash
 - c. Set Reset flags according to desired type of reset (see **“Reset Target Platform”** section of this document for reference)
6. Click **“Ok”** and then **“Save file”**
7. After saving the file, you can load it in later time to flash the image or, you can click the **“Start to flash”** button to flash the image right away

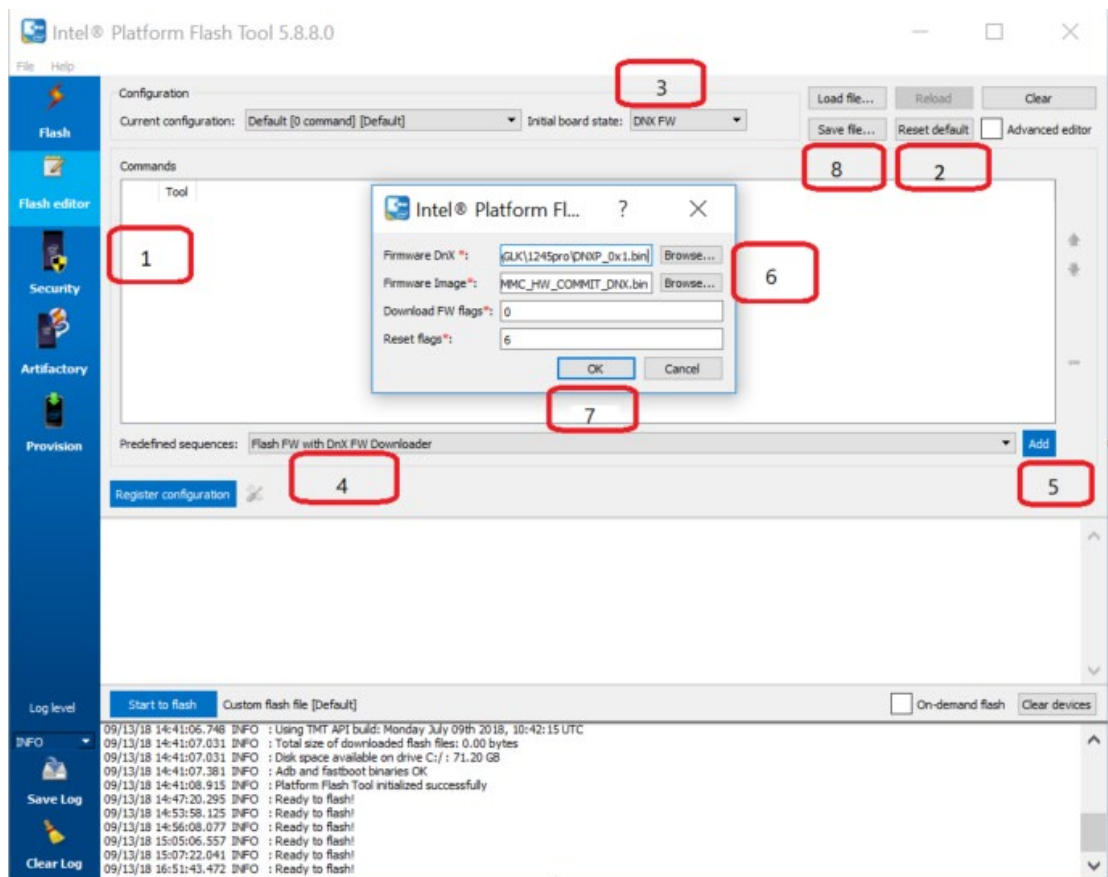




Figure 17

4.2.1.1.2 Executing different Intel® DnX operations defined in one json file

Operations can be run in GUI one-by-one by changing “**Configuration**” option for each action in the drop-down menu. Also, can define one configuration that will run number of operations one after another.

In the example below there are two Intel® DnX operations defined in one json: “dnxFwDownloader” (IFWI write) and “startover” (reset). Those operations will run under configuration name “default”, while “downloadfwos” command will also run under configuration “ifwi_flash”.

```
{
  "command" : "downloadfwos",
  "description" : "Flashing IFWI",
  "flags" : "${flags_downloadfwos}",
  "fw_dnx" : "${fw_dnx_file}",
  "fw_image" : "${fw_image_file}",
  "mandatory" : true,
  "restrict" : [
    "ifwi_flash",
    "default"
  ],
  "retry" : 2,
  "timeout" : 60000,
  "tool" : "dnxFwDownloader"
},
{
  "command" : "startover",
  "description" : "Start Over",
  "flags" : "${flags_downloadfwos}",
  "mandatory" : true,
  "restrict" : [
    "startover",
    "default"
  ],
}
```

Below example demonstrates json with two Intel® DnX commands: Device ID and IFWI write

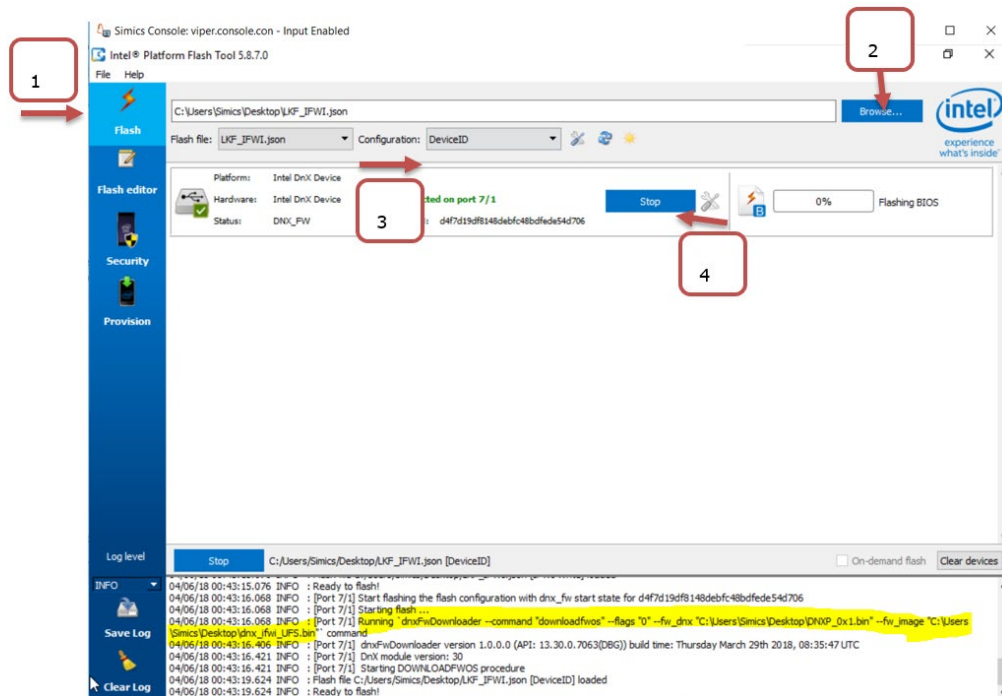


Figure 18

1. Select **“Flash”** tab on the left panel of the GUI.
2. Select the json file using the **“Browse”** button. The json file validity is then checked, and the flash operation can be started only if the selected flash file is valid. The details of the loaded flash file are printed in the log area in the DEBUG log level.
3. Once *.json file is selected and it is loaded successfully, under **“Configuration”** drop down menu, there are 2 options – **“Device ID”** and **“IFWI write”**.
4. Select **“Device ID”** option.
5. Once configuration option is selected, click on **“Start Flash”** Tab.
6. On success, Host and Target communication is established and deviceid is presented.

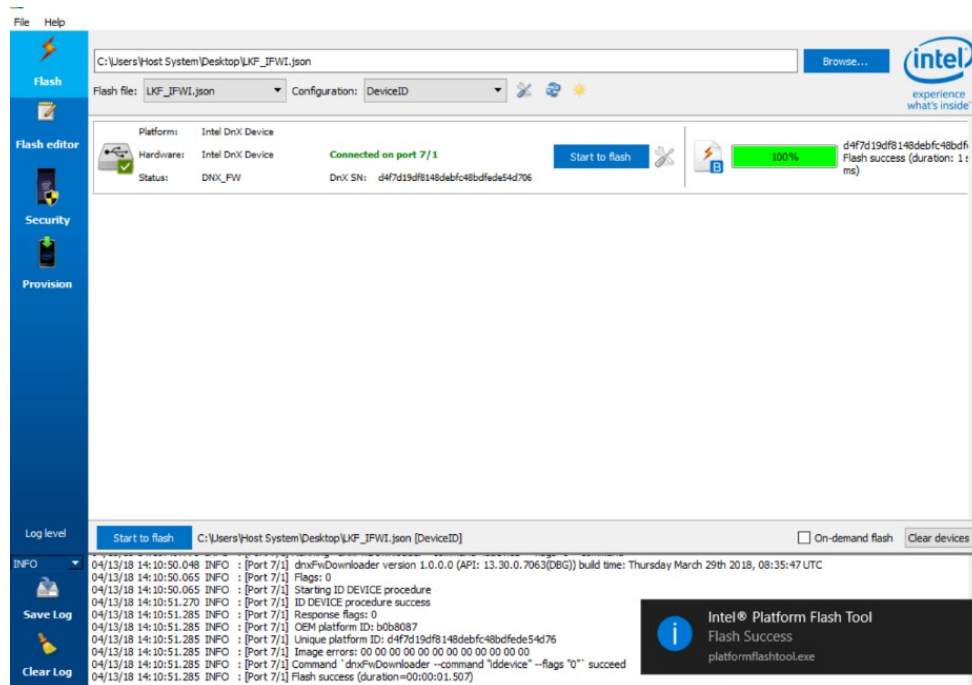


Figure 19

7. Now from configuration drop down menu, select “IFWI Write” option.
8. Click on “Start Flash” Tab.
9. This will initiate the IFWI flash process via Intel® DnX. (Can also add a command to reset the target platform after successful IFWI flashing).

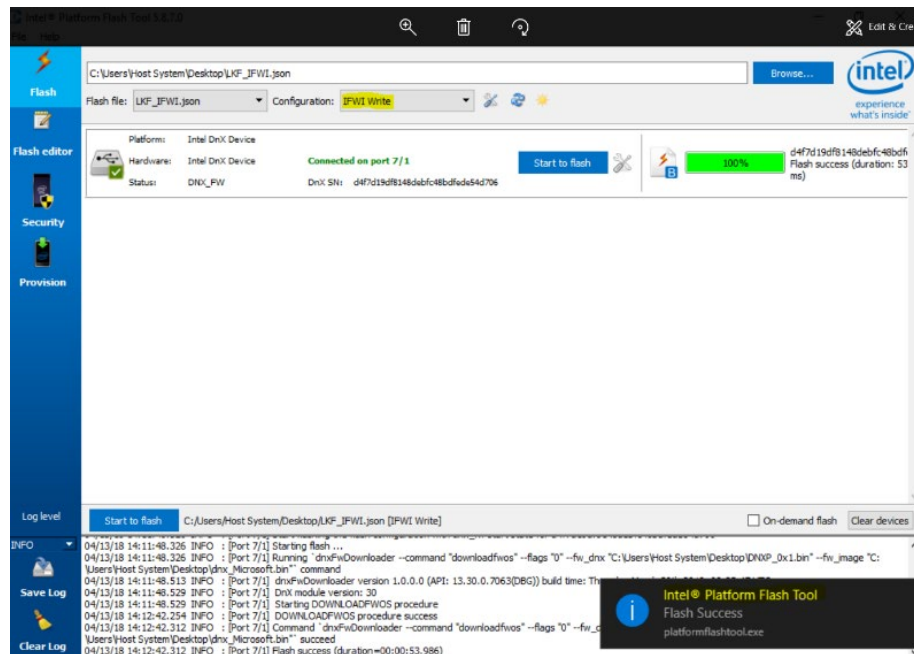


Figure 20

4.2.2 Using Command Line

In general, Intel® DnX Firmware Downloader exe is used in the following syntax -

Usage:

```
dnxFwDownloader --command <command> <command-options>
```

Help Menu

dnxFwDownloader.exe --help command lists available options/commands supported with this embedded tool.

4.2.2.1 Get storage device general info

In-order to get storage related information such as OEM PLAT ID (from IFPs), Platform Unique ID, Intel® DnX Trigger, Image Error Values, '*iddevice*' command shall be used.

Sample:

```
dnxFwDownloader.exe --command iddevice
```



4.2.2.2 Get device detailed info

In-order to get storage related information such as number of SPI components and capacity of each, '**getcardinfo**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command getcardinfo --fw_dnx
DNXP_0x1.bin --device spi --idx 0
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--device	Boot device type (spi)
--idx	device index[Optional]

4.2.2.3 Flash IFWI via Intel® DnX

In-order to flash IFWI image, '**downloadfwos**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command downloadfwos --fw_dnx
DNXP_0x1.bin --fw_image IFWI.bin --flags 0
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--fw_image	path to the firmware image
--flags	firmware download command flags[Optional]

4.2.2.4 Reset Target Platform

In-order to reset the platform, '**startover**' command shall be used.

Sample:

```
dnxFwdownloader.exe --command startover --flags 9
```

Where:



Option	Description
--flags	<p>Firmware reset command flags. In the command line appear in decimal display.</p> <p>Comprises of following info (in binary):</p> <p>Bit [1:0]: RESET_TYPE*</p> <ul style="list-style-type: none">• 00: Reset Intel® DnX protocol (no Intel® CSE /device reset) by cancelling currently active command (if any) and wait for the next command• 01: Global reset• 10: Not supported• 11: Not supported <p>Bit [3:2]: POST_RESET_STEPS</p> <ul style="list-style-type: none">• 00: After reset, take normal boot path (including honoring the Intel® DnX triggers etc.)• 01: After reset, enter OS INTEL® DNX flow• 10: After reset, ignore optional Intel® DnX triggers such as HW strap etc. and perform a full host boot• 11: Reserved

4.2.2.5 Read Boot media

In-order to read content from SPI, '**readbootmedia**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command readbootmedia --fw_dnx  
DNXP_0x1.bin -path dump.bin --device spi --idx 0 --start 0  
--blocks 4096
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--path	path to output file to dump the content



--device	Boot device type (spi)
--idx	device index[Optional]
--blocks	Number of blocks to read where each block size: 1 block = 1kByte E.g. to read 32MB: 32MB = 32768kB (in binary) = 32768 blocks

Returned data is in the 'raw' as read from the media and is not processed at all by Intel® DnX module (i.e. no decryption etc. is performed, rather all data is returned as stored on the media)

Make sure that output file location can be accessed for write, otherwise operation will fail.

4.2.2.6 Open Intel® DnX capabilities post EOM

4.3 OEM Debug Tokens

4.3.1 Using Intel® PFT GUI

Make sure the Host and Target set up is correct as described in Section 4.1

4.3.1.1 Get Part ID

Once Intel® DnX is loaded on the tool as shown in Section 4.1, simply click on "Get Device Data" to auto populate Part ID. No further step for Part ID is needed.

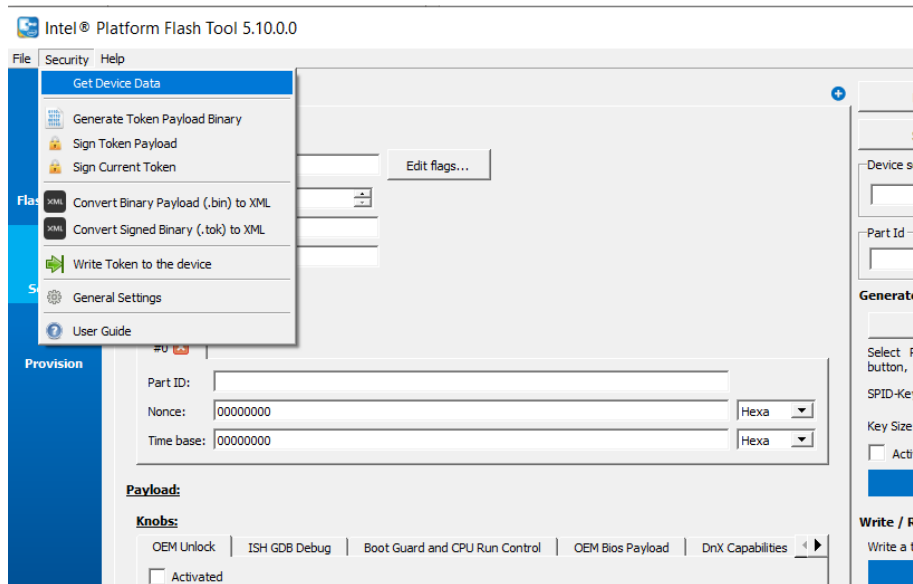


Figure 21

4.3.1.2 Inject/Write Token

The tool has a “Write” button as shown below to write the token to the system.

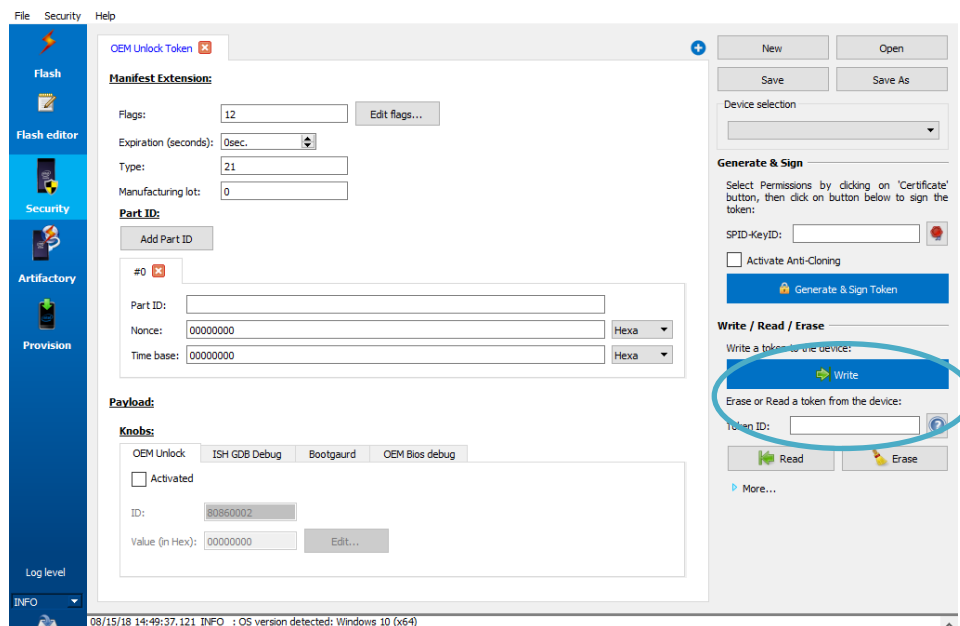


Figure 22



4.3.1.3 Erase Token

Using the “Erase” button on the UI, this tool can clear the token from the device. This requires host to be connected to the target platform with a USB cable.

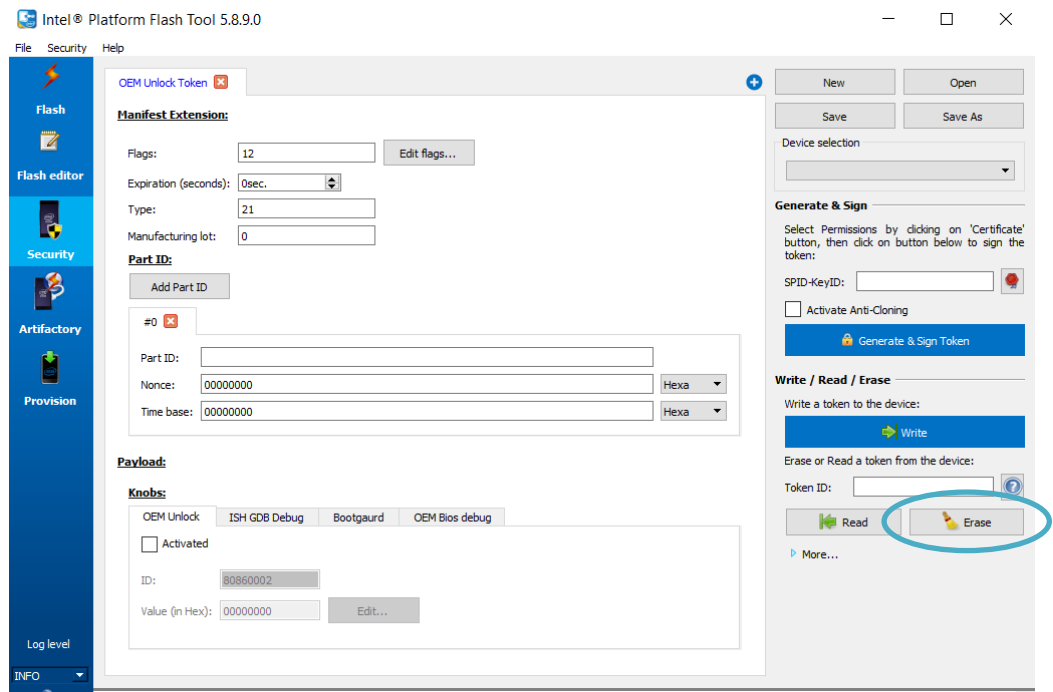


Figure 23

4.3.1.4 Read Token

Note that the device selected to read the token from should show in the “Device selection” box

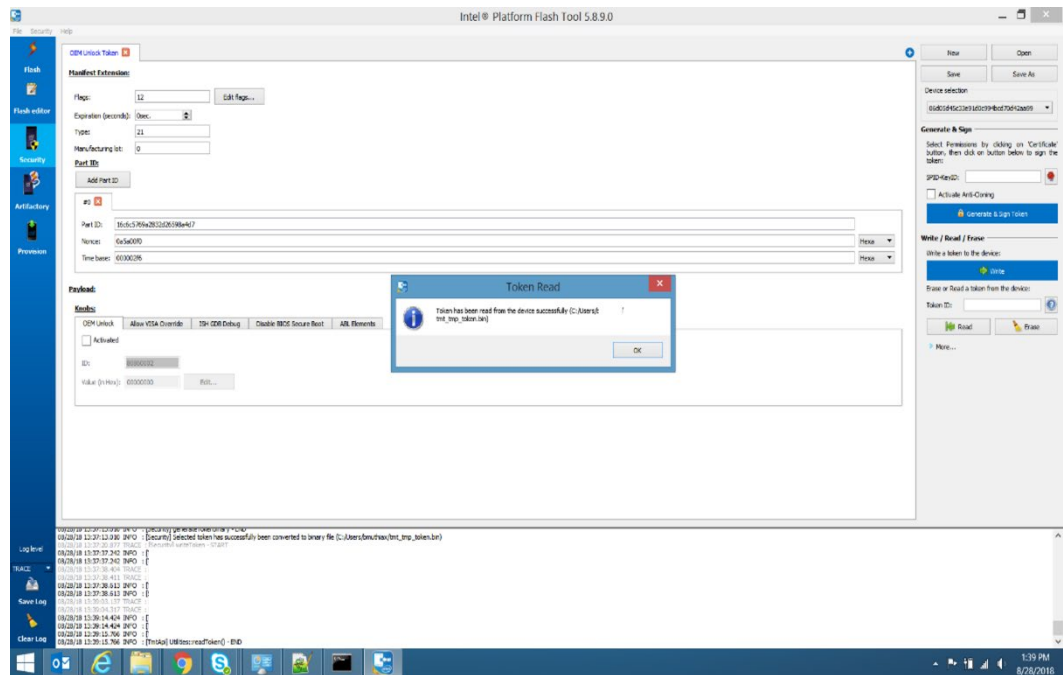


Figure 24

4.3.2 Using Command Line

Make sure the Host and Target set up for Command Line is correct as mentioned in Section 4.1

4.3.2.1 Read Part ID

In-order to get part ID specific to this token, '**gettokenpid**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command gettokenpid --fw_dnx
DNXP_0x1.bin --flags 0
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--flags	Slot number for anti-replay protection of corresponding token: <ul style="list-style-type: none"> 0: No AR protection needed. Nonce is stored in the temp storage in SRAM



	<ul style="list-style-type: none"> 1: Nonce generated is stored in first Nonce slot
--	--

4.3.2.2 Write Token

In-order to write token, '**writetoken**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command writetoken --fw_dnx
DNXP_0x1.bin --token token_to_write.bin --slot 0
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--token	path to the token
--slot	Slot Index of the token

4.3.2.3 Erase Token

In-order to erase token, '**erasetoken**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command erasetoken --fw_dnx
DNXP_0x1.bin --slot 0
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--slot	Slot Index of the token

4.3.2.4 Read Token

In-order to read token, '**readtoken**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command readtoken --fw_dnx
DNXP_0x1.bin --path read_token.bin --slot 0
```



Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--path	path to output file to dump the content of the token
--slot	Slot Index of the token

4.3.3 Common error messages

Error Code	Detail
0x30003 - Device not found	<p>If the INTEL® DNX device isn't available in the device manager: Make sure you are following the correct steps to enter INTEL® DNX mode, power cycle your platform since INTEL® DNX has a five minutes timeout (you might need to disconnect the USB cable when power cycling), check your USB cable is ok and is connected to the correct ports, check if any reworks are needed on your platform.</p> <p>If the INTEL® DNX device is available in the device manager, then the API/Tool version and the INTEL® DNX driver version used are incompatible.</p>
0x20000007 - Internal system error	This error is caused when using a non-matching INTEL® DNX SW and INTEL® DNX driver. Make sure you use the latest INTEL® DNX driver and INTEL® DNX SW.
0x30000 - basic_ios::clear	This means there's either a syntax error or the files given to the command don't exist or are incorrect.
Error code: 0x80000000 - Media not present	Boot device chosen by the hard strap is different from the device issued in the command. INTEL® DNX uses the boot device



Error Code	Detail
	chosen by the hard strap, make sure strap and command match.
0x80000008 - Invalid public key	This means your INTEL® DNX module (DNXP_0x1.bin) is signed with the wrong key. Usually this happens when trying to load the debug signed INTEL® DNX module on production platforms. You need to use a production signed INTEL® DNX module on production platforms.
Error code: 0x80000035 - Image and descriptor mismatch	This happens when trying to flash a SPI IFWI that has different regions than the IFWI already flashed on the device. This flow isn't supported. SPI INTEL® DNX only supports flashing IFWIs that have the exact same regions as the already flashed IFWI.
Error code: 0x80000039 - Invalid regions	This means your INTEL® DNX IFWI doesn't match the NVM used. Usually this happens when hard straps are set to one NVM (e.g. SPI) and trying to flash an IFWI that is for a different NVM (e.g. UFS).
Error code: 0x8000003a - Unsupported storage device	Flash initialization failed. Check that your flash is connected properly to your platform, all the hard straps are set correctly, and you have all the needed reworks.
Error code: 0x80000043 - Invalid image layout	This means that the INTEL® DNX IFWI you are trying to flash has an invalid layout. This happens when using an INTEL® DNX module that supports a new INTEL® DNX IFWI layout and trying to flash an INTEL® DNX IFWI with an older layout. If you see this error - match the versions of Intel® FIT used to generate



Error Code	Detail
	the INTEL® DNX IFWI and the INTEL® DNX module.
Error code: 0x80000045 - No DNX ifwi key in oem key manifest	This means that the INTEL® DNX IFWI created doesn't support INTEL® DNX since it misses the INTEL® DNX IFWI usage in the OEM key manifest. The INTEL® DNX IFWI usage needs to be added to the OEM key manifest in the IFWI.
0x80000058	Global reset is required to initialize the NVM before operations requiring knowledge of the layout (read/write/erase token) can be executed
0x80000001	Executed command is not supported on this boot media



5 Opening Intel® DnX capabilities post EOM

As mentioned in Chapter 3, Section 3.22 that some Image recovery/update operations are prohibited once the Intel® DnX Fuse is set at the End-Of-Manufacturing.

This section describes how to re-enable those features using OEM debug token only. Full details on how to create and sign an OEM debug token are covered in Secure Tokens Guide released in the CSME Firmware kit.

Table below lists all Intel® DnX capabilities and how they are affected by OEM:

Operation	SOC_ConfigLock is not set	SOC_Config Lock is set	Can be re-enabled through OEM debug token
START-OVER	Allowed	Allowed	N/A
ID Device (PING)	Allowed	Allowed	N/A
Download Recovery Module	Allowed	Allowed	N/A
Download OEM Key Manifest	Allowed	Allowed	N/A
Get NV Store Info	Allowed	Prohibited	Yes
Provision FW Image	Allowed	Prohibited	Yes
Set Capabilities	Allowed	Allowed	N/A
Get Token Part ID	Allowed	Allowed	N/A
Read Token	Allowed	Allowed	N/A
Write Token	Allowed	Allowed	N/A
Erase Token	Allowed	Allowed	N/A
Read Boot Media Content	Allowed	Prohibited	Yes

5.1 Flow to open prohibited post EOM Intel® DnX capabilities

1. Use Intel® PFT to generate new OEM secure token with “DnX Capabilities” knob. Set value of this knob according to the desired Intel® DnX capability
2. As mentioned in Section 3.3, prepare signed OEM KM containing:
 - a. Public key hash of private key used for signing OEM secure token
 - b. Public key hash of private key used for signing Intel® DnX Image (if IFWI flashing will be done)
3. Use PFT to run Intel® DnX command to download OEM KM to the SRAM with the Set Capabilities command

5.2 Preparing the OEM token for Intel® DnX capabilities using Intel® PFT GUI

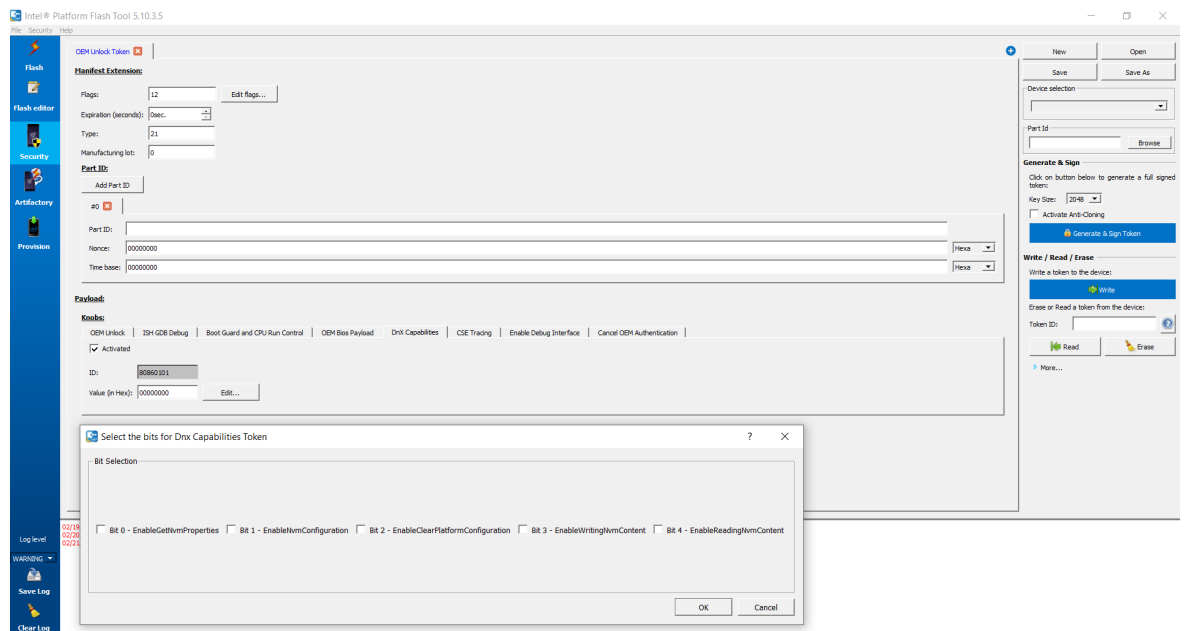


Figure 25

5.3 Preparing the OEM token for Intel® DnX capabilities using Intel® PFT CLI

Example of how to enable Intel® DnX capabilities knob inside OEM secure token using xml inside PFT installation folder (oem_unlock_token_template_project.xml)



```

<?xml version="1.0" encoding="UTF-8">
<tokens version="1" format="1" view_filter="simple">
  <token name="OEM Unlock Token" platform="Lakefield" token id="3">
    <!-- Set the desired expiration time for token to be valid-->
    <manifest_extension type="21" length="0" version="1" number_ids="1" payload_version="1" flags="0" expiration_seconds="3600" manufacturing_lot="0">
      <!-- Enter part_id, nonce and time_base of your device-->
      <part_id nonce="00000000" part_id="0x" time_base="00000000"/>
    </manifest_extension>
    <payload nb_knobs="4">
      <knob name="OEM Unlock" id="0x80860002" data="0x00000001" activated="1"/>
      <knob name="Bootguard" id="0x80860030" data="0x00000002" activated="1"/>
      <knob name="OEM Bios debug" id="0x80860051" data="0x00000001" activated="1"/>
      <!-- "activated" points to which knob is activated
      <"data" depends on the list below. The data is in hex>
      <Bitmap options>
        <EnableGetNvmProperties           - bit 0>
        <EnableNvmConfiguration          - bit 1>
        <EnableClearPlatformConfiguration - bit 2>
        <EnableWritingNvmContent          - bit 3>
        <EnableReadingNvmContent          - bit 4-->
      <knob name="DnX Capabilities" id="0x80860101" data="0x00000001f" activated="1"/>
    </payload>
  </token>
</tokens>

```

Figure 26

In this example data="0x0000001f" means that bits[4..0] = '0b11111', hence **all** Intel® DnX capabilities are open for the Intel® DnX session where above token is valid.

Note: Intel® DnX capabilities that appear under bit[1] and bit[2] in DnX knob are relevant for block boot media only, hence are not supported for SPI boot media and can be ignored.

5.4 Downloading OEM Key Manifest and Set Capabilities

In-order to download OEM Key Manifest as part of the flow to open prohibited post EOM Intel® DnX capabilities, '**downloadoemkeymanifest**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command downloadoemkeymanifest --key
OEM_KM.bin --fw_dnx DNXP_0x1.bin
```

Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--key	Path to the OEM Key Manifest binary

In-order to set Intel® DnX capabilities enabled in OEM secure token as part of the flow to open prohibited post EOM Intel® DnX capabilities, '**setcapabilities**' command shall be used.

Sample:

```
dnxFwDownloader.exe --command setcapabilities --
capabilities OEM_UnlockToken.tok --fw_dnx DNXP_0x1.bin
```



Where:

Option	Description
--fw_dnx	path to the Intel® DnX module binary
--capabilities	<p>Path to the OEM Secure Token with DnX capabilities knob enabled and set to the desired value:</p> <p>DnX capabilities knob options:</p> <p>Bit 0 - EnableGetNvmProperties</p> <p>Bit 1 – Not supported for SPI boot media</p> <p>Bit 2 – Not supported for SPI boot media</p> <p>Bit 3 - EnableWritingNvmContent</p> <p>Bit 4 - EnableReadingNvmContent</p>



6 *References*

Document	Document No. / Location
Intel® Signing and Manifesting Guide	CSME FW kit
OxM Debug Tokens Guide	CSME FW kit