

DnX (Download and Execute)

User Guide

April 2020

Revision 1.1

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 © 2020, Intel Corporation. All rights reserved.



Contents

1	Introduction.....	5
	1.1 Terminology.....	5
2	Download and Execute (DnX)	6
	2.1 Introduction	6
	2.2 Use Cases	7
	2.3 Triggers	8
	2.4 DnX flow.....	9
	2.5 Tools	9
3	High-Level Setup Detail for DnX	10
	3.1 Setup Requirements.....	10
4	Opening DnX capabilities post EOM.....	12
5	Intel® Platform Flash Tool (PFT) Overview	16
	5.1 Installation Details	16
	5.2 Usage.....	16
	5.2.1 PFT Command Line Tools for DnX	17
	5.2.2 Using GUI interface	23
	5.2.3 Common error messages.....	29



Revision History

Revision Number	Description	Revision Date
0.7	<ul style="list-style-type: none">Initial release	July 2019
0.71	<ul style="list-style-type: none">Updates for SPI based platform	Aug 2019
1.0	<ul style="list-style-type: none">Removed capabilities not supported for SPI	Nov 2019
1.1	<ul style="list-style-type: none">Adding IFWI flashing for SPI	April 2020

§ §



1 Introduction

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

1.1 Terminology

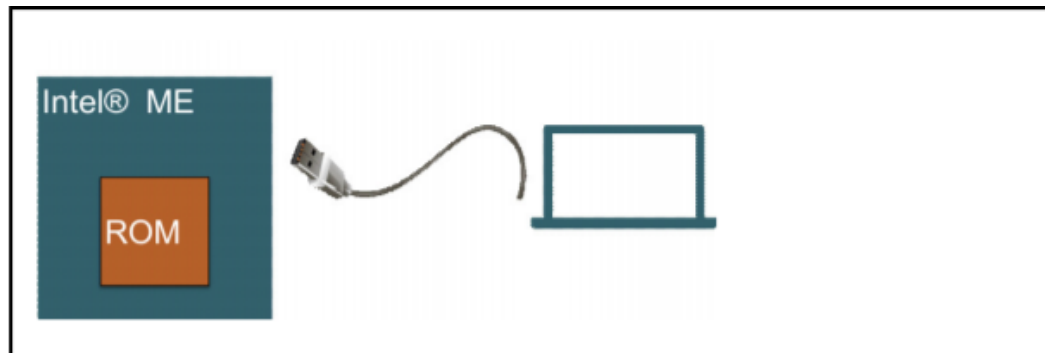
Table 1: Terminology

Acronym or Term	Definition
Intel® CSME	Intel® Converged Security Management Engine
DnX	Download and Execute
Intel® FIT	Intel® Flash Image Tool
FW	Firmware
Intel® PFT	Intel® Platform Flash Tool

2 Download and Execute (DnX)

2.1 Introduction

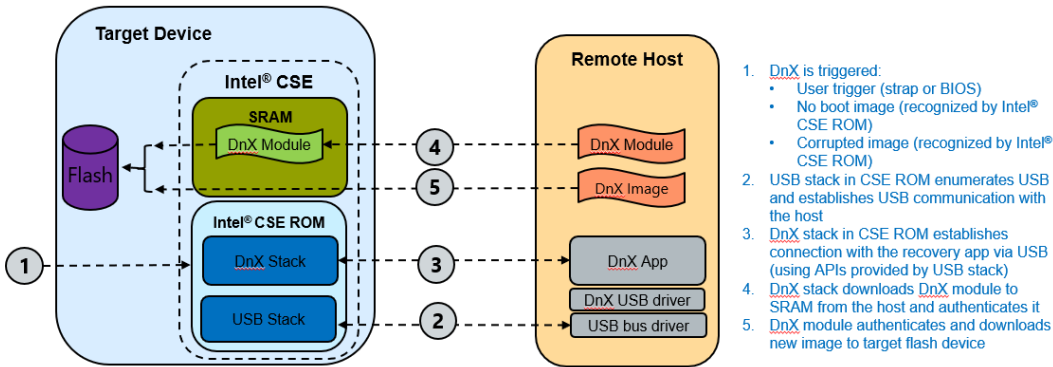
DnX is Intel's proprietary solution to download FW module to a target machine from a host machine by means of USB cable and execute it. DnX flows are executed over fixed USB 2.0 port. On SPI platforms, this capability allows to access boot media for IFWI write as well as signed Token injection for debug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



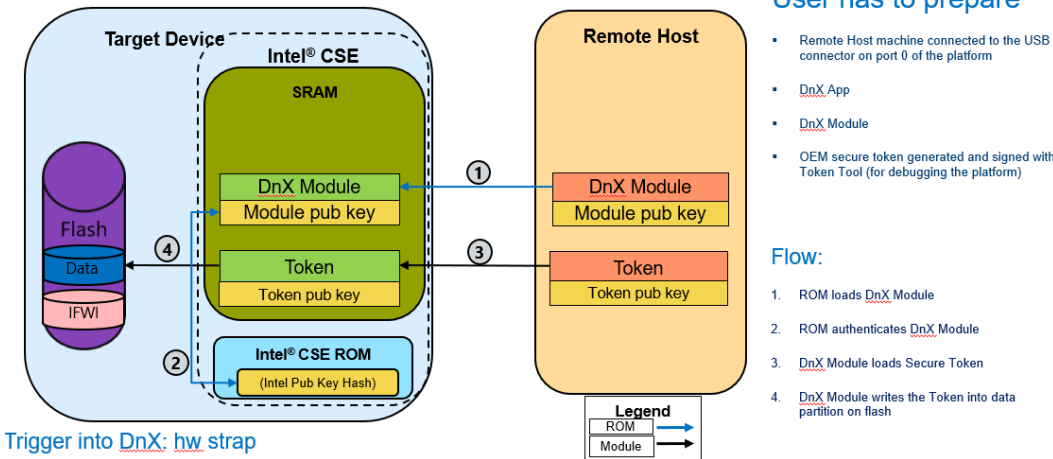
There are many advantages in using DnX, one of which is the capability to flash IFWI without opening the chassis to physically reach the flash device. This is especially important in the manufacturing line once the chassis is closed, in debug labs and in post manufacturing scenarios. Other methods such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



Injecting Secure Token for Debug



2.2 Use Cases

Below use-cases are supported on Intel® CSME 15.0.

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
Debug	<ul style="list-style-type: none">• Write/Read/Erse signed debug tokens into the platform



2.3 Triggers

Following methods can be used to trigger DnX on the platform

Method	Detail
Pin Strap	<p>TGL and RKL signal name:</p> <p>GPPC_E_22 which has 2 functions:</p> <ul style="list-style-type: none">• DDPA_CTRLCLK• DNX_FORCE_RELOAD <p>It can be used to initiate DnX trigger: '0': No DnX; '1': Enter DnX mode. Sampled by CSME ROM at Global Reset exit.</p> <p>To trigger DnX, should do Global Reset and assert the GPIO pin. Then the ROM will sample the GPIO pin and boot in DnX Mode.</p> <p>The pin can be used later, in normal boot, for another function: DDPA_CTRLCLK.</p>
Intel® CSME or BIOS Error Handling Flow	<p>If Intel® CSME or BIOS reach a critical error which prevent platform from booting, it can be programmed to enter DnX flow. (e.g Failure to authenticate BIOS signature, CSME detect FW corruption, etc.)</p>
Empty Flash Device	<p>When CSME ROM detects an empty flash device on the platform, it will enter DnX mode</p>
BIOS HECI Call	<p>BIOS can make a HECI call to Intel® CSME during boot to enter DnX after the reset</p>



2.4 DnX flow

1. DNX Trigger:
 - a. ROM Detect empty Flash
 - b. Corrupted Flash
 - c. User Trigger via BIOS or Strap
2. ROM enumerates USB and establish USB Comm. with the Host
3. ROM DNX Logic establishes connection with the recovery app via USB
4. ROM DNX logic downloads DNX module from recovery app to SRAM and authenticates it
5. DNX Module performs DnX operation requested by user

2.5 Tools

Following tools are applicable for DnX:

- **Intel® PFT** (Platform Flash Tool) – Intel implementation of DnX tool running on remote host computer. DnX module, config.xml and IFWI.bin are inserted to the target machine via this tool. Will be included in the Intel® CSME kit.
- **DnX Module** - binary file signed by Intel. This file has Intel® CSME 15.0 the DnX logic Intel® CSME ROM will run. Will be included in the Intel® CSME 15.0 kit.
- **Intel® FIT** – can be used to create DnX based IFWI image. For more detail on how to create IFWI image for DnX, please refer to Intel® Bring up Guide in the Intel® CSME 15.0 kit.
- **Intel® FPT** - can be used to configure DnX fuse and close manufacturing on the platform. Will be included in the Intel® CSME 15.0 kit.

3 High-Level Setup Detail for DnX

3.1 Setup Requirements

In order to complete DnX flow, the following setup is required

DnX Test Setup

Target Device

- Enters DnX mode based on trigger

Management Console

- Platform Flash Tool (PFT)
- DnX module
- IFWI image



Requirements:

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



Requirement	Usage
OEM signed secure token	Token binary to be flashed on the target system (Can be created by Intel® PFT tool provided in the Intel® CSME)



4 Opening DnX capabilities post EOM

Table below lists all 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 DnX 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

Note: 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).

Some capabilities, like secure token handling operations, are always available. While other DnX capabilities, allowing access the boot media, are prohibited once platform



goes through EOM (End of Manufacturing). In order to enable them, OEM authorization is required.

This authorization is provided by generating an OEM secure token with “DnX capabilities” knob enabled. Value set in this knob will specify which of the DnX capabilities will be allowed for the specific session where token is valid.

Flow to open prohibited post EOM DnX capabilities:

1. Use PFT to generate new OEM secure token with active “DnX Capabilities” knob. Set value of this knob according to the desired DnX capability (see Figure 4.1 and 4.2 for example). To see more details on how to generate a token refer to “Secure Tokens Guide”
2. Sign OEM secure token. To see more details on how to sign a token refer to “Signing and Manifesting Guide”.
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 DnX Image (if IFIW flashing will be done)
4. Use PFT to run DnX command to download OEM KM to the SRAM (need to provide OEM KM and DnX Module as input). See command example in “Open DnX capabilities post EOM” section below.
5. Use PFT to run DnX command to set capabilities defined in OEM secure token (need to provide OEM secure token and DnX Module as input). See command example in “Open DnX capabilities post EOM” section below.

Example of how to enable DnX capabilities knob inside OEM secure token using PFT GUI:

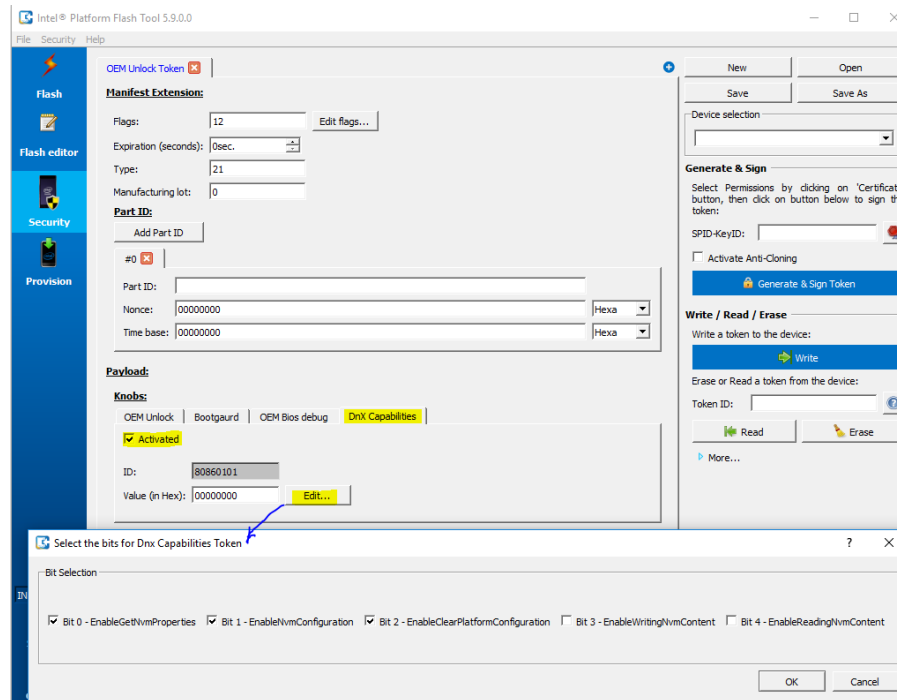


Figure 4.1

Example of how to enable 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="Lakesfield" 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="0x00000001" activated="1"/>
    </payload>
  </token>
</tokens>
```

Figure 4.2



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

Note: 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.

Note: In order to provide debug support for OEM platforms post EOM, Intel has the capability to open DnX for specific platform.



5 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 Remote Host.

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

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

5.1 Installation Details

PFT tool is available within Intel® CSME FW Kit->Tools->DnX Tools. 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.



5.2 Usage

Once PFT tool is installed on the Remote Host:

- Make sure the target system is connected to the Remote Host using USB cable.
- Make sure all input files required for DnX operation (e.g. DnX module, token) are available on the Remote Host.
- Make sure the target system is in DnX mode.



5.2.1 PFT Command Line Tools for DnX

DnX Firmware Downloader

This command line tool provides means to interact with Intel® CSME firmware and perform different DnX operations.

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 DnX mode when running DnX operations (e.g jumper, virgin storage)

5.2.1.1 DnX Firmware Downloader

Usage:

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

Help Menu

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

5.2.1.1.1 Get storage device general info

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

Sample:

`dnxFwDownloader.exe --command iddevice`

5.2.1.1.2 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 DnX module binary
--device	Boot device type (spi)
--idx	device index[Optional]

5.2.1.1.3 Flash IFWI via 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 DnX module binary
--fw_image	path to the firmware image
--flags	firmware download command flags[Optional]

5.2.1.1.4 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 DnX protocol (no Intel® CSME /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 DnX triggers etc.) • 01: After reset, enter OS DNX flow • 10: After reset, ignore optional DnX triggers such as HW strap etc. and perform a full host boot • 11: Reserved
---------	--

5.2.1.1.5 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 DnX module binary
--path	path to output file to dump the content
--device	Boot device type (spi)
--idx	device index[Optional]
--start	address to start to read from (in Bytes).
--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 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.

5.2.1.1.6 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 DnX module binary



--path	path to output file to dump the content of the token
--slot	Slot Index of the token

5.2.1.1.7 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 DnX module binary
--token	path to the token
--slot	Slot Index of the token

5.2.1.1.8 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 DnX module binary
--slot	Slot Index of the token

5.2.1.1.9 Get Token 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 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• 1: Nonce generated is stored in first Nonce slot

5.2.1.1.10 Open DnX capabilities post EOM

In order to download OEM KM as part of the flow to open prohibited post EOM 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 DnX module binary
--key	Path to the OEM Key Manifest binary

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

Sample:


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

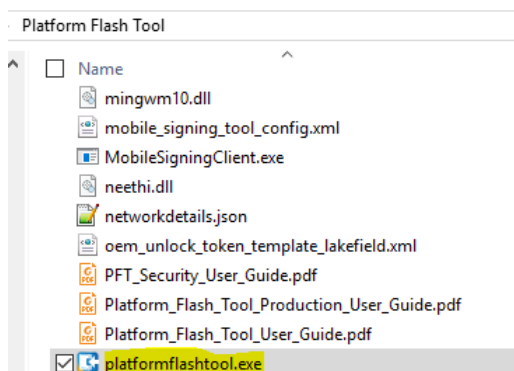


Where:

Option	Description
--fw_dnx	path to the 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>

5.2.2 Using GUI interface

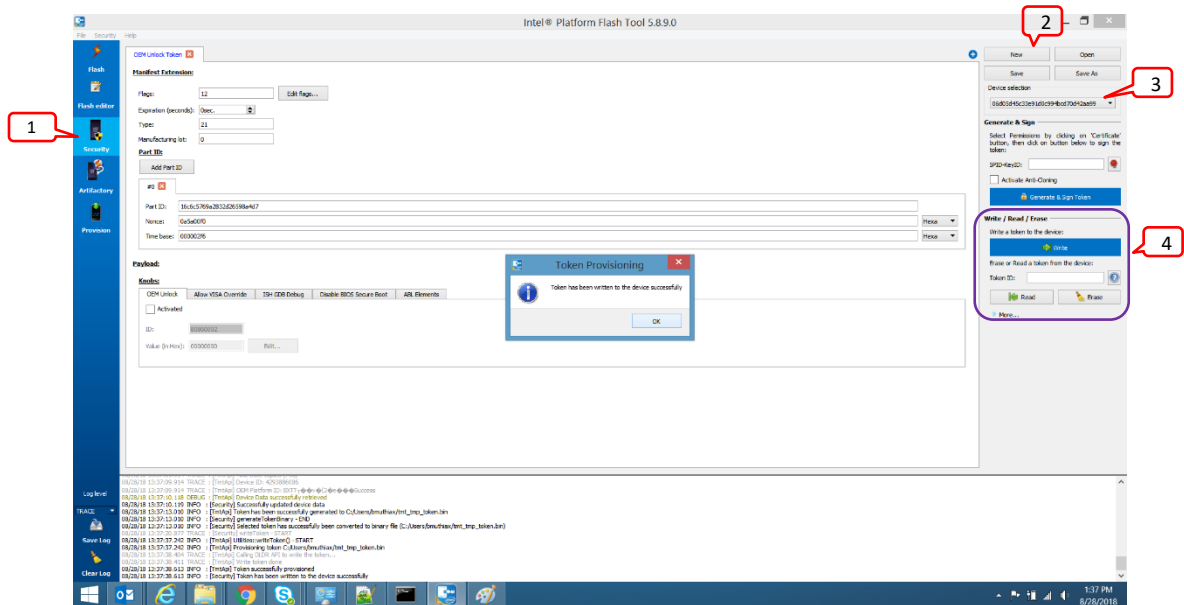
- To launch GUI interface, click on the Desktop icon  which launches GUI interface or open it from Intel® PFT installation folder.



- Go to the Security tab on the left.
- To inject a token
 - Generate and Sign a new token or brows for an existing token to write into the device. For guidance on how to generate and sign secure token, see “Secure Token Guide”



- Select a device
- press button “Write” in “Write / Read / Erase” section of the GUI
- To Read or Erase a token
 - Select a device
 - press button “Read” / “Erase” in “Write / Read / Erase” section of the GUI



5.2.2.1 Executing json file containing DnX commands

In order to execute json file with 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 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 DnX module and DNX IFWI binary.

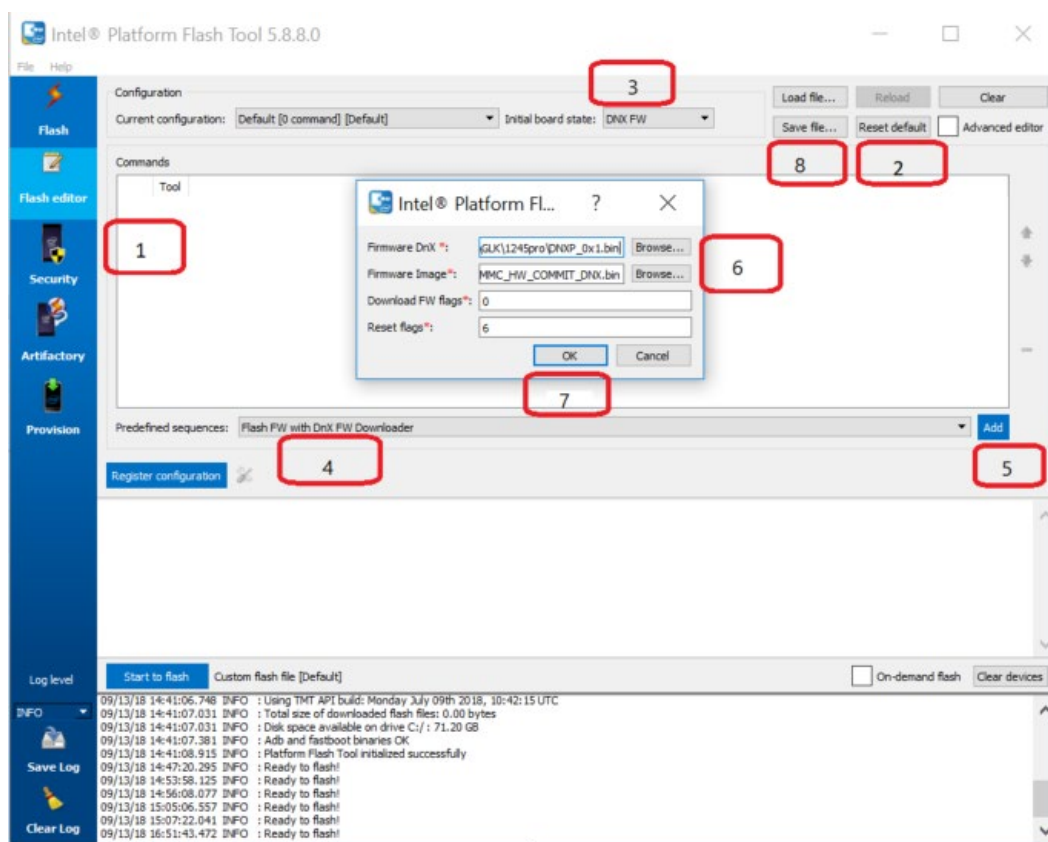
5.2.2.1.1 Creating json file for flashing Intel® CSME IFWI thru DnX

In order to create json file supporting IFWI flash command via 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 DnX module. The 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





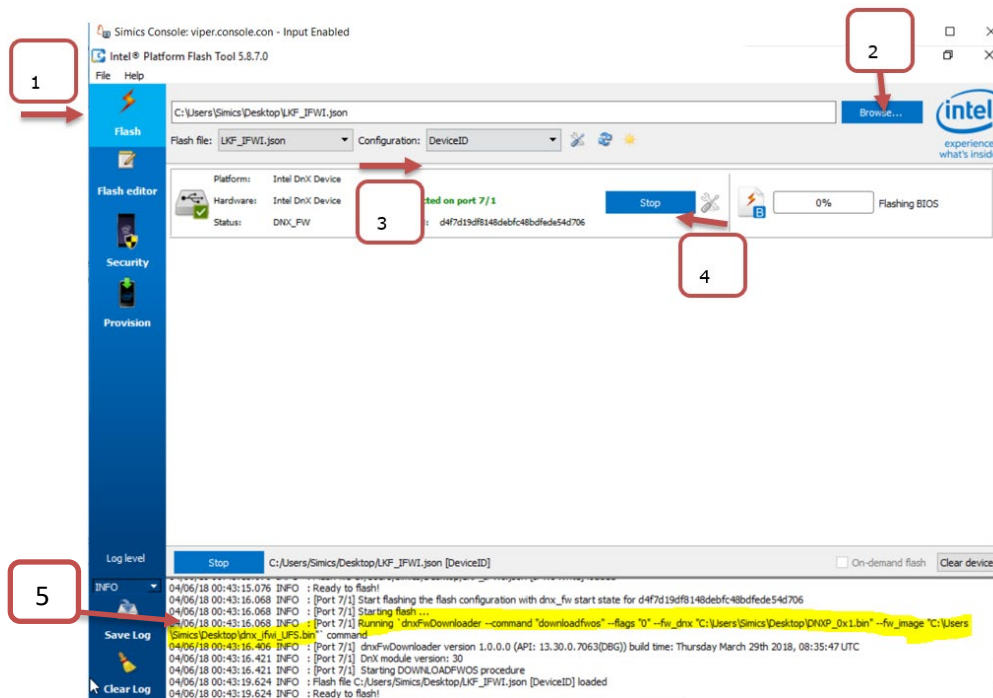
5.2.2.1.2 Executing different DnX operations defined in one json file

It is possible to support number of DnX operations in one json file, each operation will appear as “command” when reviewing json file in text editor (see example of such json below). 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.

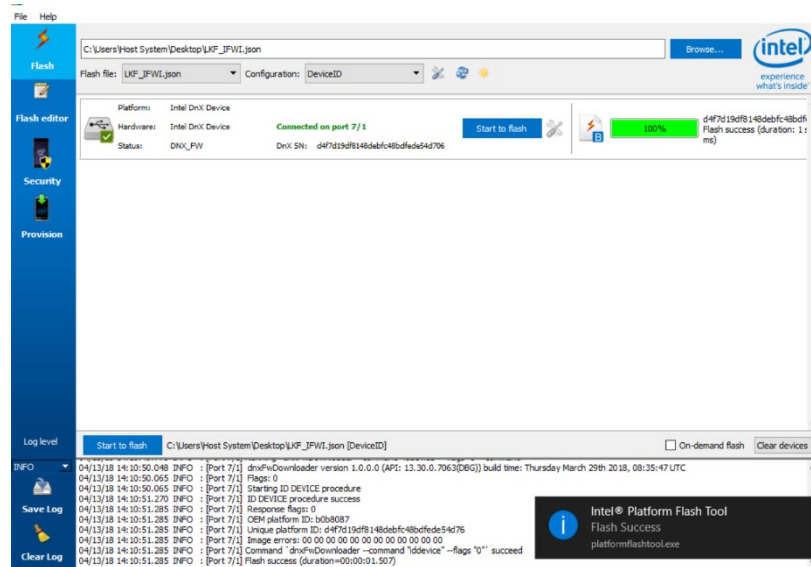
In the example below there are two DnX operations defined in one json: “dnxFwDownloader” (IFWI write) and “startover” (reset). Those operations will run under configuration name “default”, while “downloadfwos” command will also run under configuration “ifwi_flash”.

```
{
  "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"
  ],
}
```

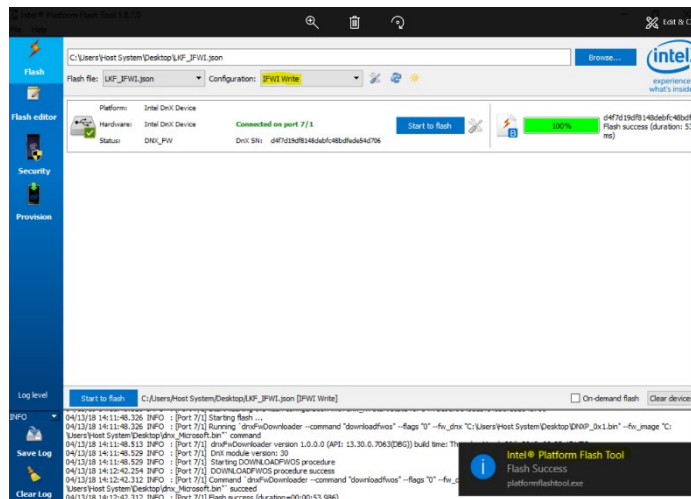
Below example demonstrates json with two DnX commands: Device ID and IFWI write



- Select **“Flash”** tab on the left panel of the GUI.
- 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.
- Once *.json file is selected and it is loaded successfully, under **“Configuration”** drop down menu, there are 2 options – “Device ID” and “IFWI write”.
- Select **“Device ID”** option.
- Once configuration option is selected, click on “Start Flash” Tab.
- On success, Host and Target communication is established and deviceid is presented.



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





5.2.3 Common error messages

Error Code	Detail
0x30003 - Device not found	<p>If the DNX device isn't available in the device manager: Make sure you are following the correct steps to enter DNX mode, power cycle your platform since 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 DNX device is available in the device manager, then the API/Tool version and the DNX driver version used are incompatible.</p>
0x20000007 - Internal system error	This error is caused when using a non-matching DNX SW and DNX driver. Make sure you use the latest DNX driver and 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. DNX uses the boot device chosen by the hard strap, make sure strap and command match.
0x80000008 - Invalid public key	This means your DNX module (DNXP_0x1.bin) is signed with the wrong key. Usually this happens when trying to load the debug signed DNX module on production platforms. You need to use a production signed DNX module on



Error Code	Detail
	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 DNX only supports flashing IFWIs that have the exact same regions as the already flashed IFWI.
Error code: 0x80000039 - Invalid regions	This means your 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 DNX IFWI you are trying to flash has an invalid layout. This happens when using a DNX module that supports a new DNX IFWI layout and trying to flash a DNX IFWI with an older layout. If you see this error - match the versions of Intel® FIT used to generate the DNX IFWI and the DNX module.
Error code: 0x80000045 - No DNX ifwi key in oem key manifest	This means that the DNX IFWI created doesn't support DNX since it misses the DNX IFWI usage in the OEM key manifest. The DNX IFWI usage needs to be added to the OEM key manifest in the IFWI.



Error Code	Detail
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