

Intel® Integrated Sensor Solution Utility

User Guide

November 2020

Revision 1.4

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 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	6
1.1	Overview	6
1.2	Operating System Support	6
1.3	Error Return	6
2	Intel® Integrated Sensor Solution Utility	7
2.1	Purpose	7
2.2	System Requirements	7
2.3	Required Files and Installation:	7
2.4	Intel® ISS Utility Usage	8
2.5	Options Specification	10
2.5.1	Examples:	10
2.6	ISSU -BIST Functionality	10
2.6.1	Examples:	11
2.7	BIST Levels Specifications	12
2.8	ISSU -INFO Functionality	16
2.9	ISSU -PDT Functionality	19
2.10	ISSU -I2C Functionality	21
2.11	ISSU -NVM Functionality	22
3	Appendix A: Error List	25
4	Appendix B: Get Decompressed ISS Binary File	28

Figures

Figure 2-1. ISSU Command Usage and Examples	9
Figure 2-2. Example of "BIST Level 0" Functionality	12
Figure 2-3. Example of "BIST Level 1" Functionality	13
Figure 2-4. Example of "BIST Level 2" Functionality	14
Figure 2-5. Example of "BIST Level 3" Functionality	15
Figure 2-6. Example of -INFO Output	17
Figure 2-7. Example of -BOM Output	18
Figure 2-8. Example of -SensorList Output	19
Figure 2-9. Example of "GetPDT" Functionality	20
Figure 2-10. Example of "SetPDT" Functionality	20
Figure 2-11. Example of "I2C Calibrate" Functionality	21
Figure 2-12. Example of "I2C calibrate and update PDT" Functionality	22
Figure 2-13. Example of "GetExceptions"/"GetErrorLog0"/"GetUserData1" Functionality	23
Figure 2-14. Example of "ParseExceptions" Functionality	24



Tables

Table 2-1. Commands Specification	8
Table 2-2. -BIST [-arguments]	11
Table 2-3. BIST Level 1 Connectivity	13
Table 2-4. BIST Level 2 Calibration	14
Table 2-5. BIST Level 3 Sensors BIST	14
Table 2-6. -INFO [-arguments]	16
Table 2-7. -INFO Command Returns	16
Table 2-8. -PDT [-arguments]	19
Table 2-9. -I2C [-arguments]	21
Table 2-10. -NVM [-arguments]	22



Revision History

Document Number	Revision Number	Description	Revision Date
XXXXXX	0.8	Review and return	July 2016
XXXXXX	0.9	Added Yocto Linux support	December 2016
XXXXXX	1.0	Added I2C calibrate command and EFI support	February 2017
XXXXXX	1.1	Added SensorList function	June 2017
XXXXXX	1.2	Added EClite BIST function	July 2018
XXXXXX	1.3	Added NVM file related function	August 2018
XXXXXX	1.4	Added some test notes of BIST tests.	November 2020



1 Introduction

1.1 Overview

This document will describe the Intel® Integrated Sensor Solution Utility tool which is a stand-alone tool for use by OEMs and ODMs. The tool encompasses several different sensor tests and functionalities into a single tool.

The Intel® Integrated Sensors Solution Utility (or ISSU) will be provided as part of the Intel® ISS kit and will be located in:

- SW\ISSU

This folder includes the following:

- Windows* folder (for win32)
- Windows* 64 folder (for win64)
- Linux folder (for Yocto Linux 64bit)
- EFI64 folder (for UEFI 64bit)
- Intel(R) Integrated Sensor Solution Utility User Guide.pdf

The tool provides several different sets of functionality:

1. Information gathering about Intel® ISS
2. BIST sensor and ECLITE functionality
3. Updating and retrieving the Intel® ISS PDT file
4. Calibrating I2C bus functionality
5. Retrieving and parse NVM file functionality

1.2 Operating System Support

OS	SUPPORT Y/N
EFI Shell 64 bit	Y
Windows PE 64bit & 32bit	Y
Windows 10 DT 64bit & 32bit	Y
Windows 10 PE 64bit & 32bit	Y
Yocto Linux (GLIBCXX_3.4.21, Kernel 4.1.27) 64bit	Y

1.3 Error Return

Intel® ISS Utility will return 0 for success and a value of 1 to indicate an error. A detailed error code is displayed on the screen.



2 Intel® Integrated Sensor Solution Utility

2.1 Purpose

The Intel® Integrated Sensor Solution Utility (or Intel® ISSU) performs a number of useful functions that work with the Intel® Integrated Sensor Solution. These include conducting sensor self-tests, retrieving information about the Intel® Integrated Sensor Solution status and components, and getting or setting the Intel® ISS PDT configuration file.

2.2 System Requirements

Intel® ISSU runs on the OSs described in section "Operating System Support". The tool will execute only on systems that have the Intel® ISS enabled and running.

2.3 Required Files and Installation:

The Intel® ISS utility main executable is **ISSU.exe** for windows version, **ISSU** for Linux version, **ISSU.efi** for EFI version.

For windows version, the following files must be in the same directory as **ISSU.exe**:

- For win32: Idrv.dll, Pmxdll.dll
- For win64: Idrv.dll32e.dll, Pmxdll32e.dll

For windows version, the installation of the following driver is required for Intel® ISSU to run:

- Intel® Integrated Sensor Solution
- ISS Dynamic Bus Enumerator
- HID PCI Minidriver for ISS

For Linux version, the installation of the following driver is required for Intel® ISSU to run:

- intel_ishtp
- intel_ish_ipc
- intel_ishtp_hid
- intel_ishtp_clients

For all the examples/images described in this document, it would use windows version as reference. Linux/EFI version of the tool should have similar behavior except special notes declared in detailed sections.



2.4 Intel® ISS Utility Usage

The tool is divided into several commands. Each command has its own functionality and specific arguments.

In addition to arguments that are specific per command, there are also Options. Options are arguments that are available for all commands.

The tool command line is structured as below:

- ISSU [-command] [-arguments] [-options]

Table 2-1. Commands Specification

Option	Description
No option	Displays the help screen
-BIST	Executes ISS built-in self-tests; Retrieves ECLite BIST results if eligible.
-INFO	Returns information about ISS status and configurations
-PDT	Allows to get or set the PDT
-I2C	Allows to calibrate specific I2C bus
-NVM	Allows to get or parse specific file in NVM
-EXP	Shows examples of how to use the tool. Each command (BIST/PDT/Info) has the -exp command available and will show specific command examples for the functionality
-H or -?*	Displays the usage screen. Each command (BIST/PDT/Info) has the -h command available and will show specific help menu for the functionality *Note: "-?" is not supported in EFI version
-VER	Shows the version of the tool
-ERRLIST	Return a list of available codes



Figure 2-1. ISSU Command Usage and Examples

The figure consists of two screenshots of a Windows Command Prompt window titled "Administrator: Command Prompt". The first screenshot shows the command `ISSU.exe -h` being executed, which displays the help information for the tool. The second screenshot shows the command `ISSU.exe -exp` being executed, which displays a list of example commands and their usage.

```
Administrator: Command Prompt
c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -h
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

Usage: ISSU.exe [-command] [-arguments] [-options]
Command includes:
  -INFO      : Returns information about ISS status and configurations
  -BIST      : Executes ISS built in self tests
  -PDT       : Allows to get or set the PDT
  -I2C       : Executes I2C operations
  -NVM       : Allows to get or parse specific file in NVM
  -VER       : Returns the version of the tool
  -ERRLIST   : Returns a list of available codes
  -h|?      : Help information
  -EXP       : Valid command line example

Options include:
  -Page      : Will pause after output exceeds the screen size
  -Verbose   : Will provide detailed information about tool operations

c:\Users\BXT-P\Desktop\ISSU\Windows64>
```

```
Administrator: Command Prompt
c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -exp
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

ISSU.exe -h
ISSU.exe -BIST -Test 3 -Verbose -Page
ISSU.exe -BIST -Test 1 -LUID <LUID1,LUID2...>
ISSU.exe -BIST -ALL
ISSU.exe -INFO -Page
ISSU.exe -PDT -SetPDT <bin-file-path> -Verbose
ISSU.exe -PDT -GetPDT <bin-file-path>
ISSU.exe -I2C -Calibrate 1 -UpdatePDT
ISSU.exe -NVM -GetExceptions <bin-file-path>
ISSU.exe -NVM -ParseExceptions <bin-file-path>

c:\Users\BXT-P\Desktop\ISSU\Windows64>
```



2.5 Options Specification

Options are generic for the tool and are available for all commands i.e.:

- Usage
- ISSU <<COMMAND: -BIST|-INFO|-PDT|-I2C>> <<COMMAND SPECIFIC ARGUMENTS>> [-Verbose] [-Page]

Option	Description
Verbose	Displays the maximum information available for operations the tool is performing.
Page	When the output exceeds screen size the output will pause and wait for any key press to proceed.

2.5.1 Examples:

- ISSU -BIST -TEST 3 -Verbose -Page
- ISSU -INFO -Page
- ISSU -PDT -SetPDT file.bin -Verbose
- ISSU -I2C -Calibrate 1 -UpdatePDT

2.6 ISSU -BIST Functionality

The BIST functionality will perform several levels of integrated self-tests. The test level desired can be specified as part of the arguments.

Note: Do not run Intel SensorViewer tool during BIST tests.

Note: Some tests may be implemented by the sensor vendor. These tests are not mandatory.

Running the -BIST functionality with no arguments will result in running the default BIST level for all sensors and retrieve ECLite BIST results if eligible. The default BIST level is test level 3.

Note: Up to BIST level 3, each BIST level also executes the levels below it.

For example:

- BIST level 3 will execute BIST level 3, 2, 1, 0 and ECLite if eligible.
- BIST level 2 will execute BIST level 2, 1, 0 and ECLite if eligible.
- BIST level 1 will execute BIST level 1, 0 and ECLite if eligible.



Table 2-2. -BIST [-arguments]

Option	Description
No option	Run default BIST level for all sensors on system
-h -?*	Returns the help screen for ISSU -BIST *Note "-?" is not supported in EFI version
-EXP	Shows detailed examples of how to use the BIST command
-TEST <level>	Executes test level indicated. Single test level is possible per command line and will be executed for all sensors on system unless the -LUID argument is specified
-ALL	Runs all available BISTs (may be higher level than default level) for all sensors on system unless the -LUID argument is specified, and retrieve ECLITE test results if enabled.
-LUID <luid1,luid2...>	Specifies which sensors to run the BIST on. More than one LUID may be specified as an argument (separated with ",").
-ECLITE	Retrieve ECLITE test results individually.

2.6.1 Examples:

- ISSU -BIST
- ISSU -BIST -ALL
- ISSU -BIST -test 3
- ISSU -BIST -test 2 -LUID 0073000100010002
- ISSU -BIST -test 2 -LUID 0073000100010002,020F000100080042
- ISSU -BIST -ALL -Verbose -Page
- ISSU -BIST -ALL -LUID 0073000100010002



2.7 BIST Levels Specifications

Level	BIST Level 0: Configuration
Test Specifics	<p>BIST level 0 checks:</p> <ol style="list-style-type: none">1. FW is alive and responding2. The virtual sensors configured have all required reporters that are needed as input. Virtual sensors rely on other sensors as inputs. This test will verify each virtual sensor has all its required reporters. The test also checks the optional reporters. These reporters are not mandatory but may be added for increased accuracy. If an optional reporter is missing the test will not fail but a notification/warning will be shown to the user.
Test Indication to user	<p>Pass: FW is alive and all virtual sensors have all their defined inputters configured correctly</p> <p>Fail:</p> <ol style="list-style-type: none">1. ISH FW not alive/responding, Please verify CSE/ISS compatibility, debug may be needed2. Missing mandatory reporter:<ol style="list-style-type: none">a. Error: Check in PDT the virtual sensor mandatory reporters.b. Warning: User may want to add the optional reporter for better accuracy.

Figure 2-2. Example of “BIST Level 0” Functionality

```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -bist -test 0 -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Starting self test
Initializing connection to FW
Sending BIST command to FW
Starting to poll FW for result
Test result received

Self Test:
Test Level: 0 (Configuration)
Result: test passed for all sensors

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```

**Table 2-3. BIST Level 1 Connectivity**

Level	BIST Level 1: Connectivity
Test Specifics	<p>BIST level 1 tests connectivity test to each sensor and the GPIO configuration.</p> <p>Checks that FW can reach each physical sensor:</p> <ul style="list-style-type: none"> • A register is ready from each sensor and its value tested against the expected value. <p>Checks GPIO configuration:</p> <ul style="list-style-type: none"> • BIOS GPIO table matches PDT configuration.
Test Indication to user	<p>Pass: GPIO and sensors are configured correctly, Soldering/BUS lines are working properly.</p> <p>Fail:</p> <ul style="list-style-type: none"> • ISS could not reach all the sensors: Sensor is not reachable/configured correctly - check soldering or check the settings in PDT are correct. • uDriver could not generate a GPIO interrupt for that sensor - check with Intel/Vendor GPIO configuration. • BIOS config failure: GPIO that is configured in PDT for this sensor is not assigned in BIOS for ISH - Check BIOS config table.

Figure 2-3. Example of “BIST Level 1” Functionality

```

Administrator: C:\Windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -bist -test 1 -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Starting self test
Initializing connection to FW
Sending BIST command to FW
Starting to poll FW for result
Test result received

Self Test:
Test Level: 1 (Connectivity, Configuration)
Result: test passed for all sensors

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>

```



Table 2-4. BIST Level 2 Calibration

Level	BIST Level 2: Calibration
Test Specifics	BIST level 2 checks required calibration data exists for sensors and is in the correct format. If calibration is not required for a sensor, success is returned.
Test Indication to user	Pass: Calibration configured correctly - note this does not mean calibration values are correct, just that they are present and in the correct format. Fail: Check PDT for missing calibration or for wrong calibration format.

Figure 2-4. Example of “BIST Level 2” Functionality

```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -bist -test 2 -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Starting self test
Initializing connection to FW
Sending BIST command to FW
Starting to poll FW for result
Test result received

Self Test:
Test Level: 2 (Calibration, Connectivity, Configuration)
Result: test passed for all sensors

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```

Table 2-5. BIST Level 3 Sensors BIST

Level	BIST Level 3: Sensors BIST
Test Specifics	BIST level 3 runs vendor defined sensor BIST. This test is not a mandatory test, some sensors may not have BIST level 3 available.
Test Indication to user	Pass: Test completed successfully. Fail: Need to contact Intel/Vendor. Each test is sensor specific and failure may mean different things for different sensors.



Figure 2-5. Example of “BIST Level 3” Functionality

```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -bist -test 3 -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Starting self test
Initializing connection to FW
Sending BIST command to FW
Starting to poll FW for result
Test result received

Self Test:
Test Level: 3 (Sensors BIST, Calibration, Connectivity, Configuration)
Result: test passed for all sensors

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```

```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -bist -test 3 -luid 02350000000000C0 -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Starting self test
Initializing connection to FW
Sending BIST command to FW
Starting to poll FW for result
Test result received

Self Test:
Test Level: 3 (Sensors BIST, Calibration, Connectivity, Configuration)
Result: test passed for all sensors
Sensor LUID: 02350000000000C0 Sensor Model: Dummy Intel Result: success

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```



2.8 ISSU -INFO Functionality

The -INFO functionality retrieves various information about Intel® ISS.

It can be used for triage and for gathering information for triage.

Table 2-6. -INFO [-arguments]

Option	Description
No option	Reports information for all components listed in table below
-h -?*	Returns the help screen for ISSU -INFO *Note: "-?" is not supported in EFI version
-EXP	Shows detailed examples of how to use the INFO command
-BOM	Retrieves sensor LIUD and sensor model for all physical sensors on the system in order to get a clear BOM on the current system
-SensorList <bin-file-path>	Retrieves sensor related information from uncompressed ISS binary file and save to local file

Examples:

- ISSU -INFO -Page
- ISSU -INFO -BOM

The -INFO command will return information as specified in the table below:

Table 2-7. -INFO Command Returns

Feature Name	Field Value
Integrated Sensor Solution FW state	Responding / Not Responding
Integrated Sensor Solution FW status	FW is Running / Sensor Core loaded / PDT loaded / Sensor Core running
Integrated Sensor Solution FW version	Version string
Intel® Integrated Sensor Solution Driver version *	Version string
ISS Dynamic Bus Enumerator version *	Version string
HID PCI Minidriver for ISS version *	Version string
PDT version	Version string
Vendor defined data version	Version string



Feature Name	Field Value
Sensors Information	<p>Information on the various Sensors configured on the platform.</p> <p>For each sensor (physical and virtual):</p> <ul style="list-style-type: none"> • Sensor LUID • Sensor name • Vendor • Sensor model • Bus type • Bus address • Calibration status (set/not set)

*Note: Linux and EFI version won't show such information

Figure 2-6. Example of -INFO Output

```

Administrator: C:\Windows\system32\cmd.exe - ISSU.exe -info -page
C:\Users\BXT-P\Desktop\ISSU\Windows64\ISSU.exe -info -page
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

FW Information :
FW State: responding
FW Status: 0x0F - Sensor Core running
FW Version: 4.0.0.3257

Drivers Information :
Intel(R) Integrated Sensor Solution Driver version: 3.1.0.3357
ISS Dynamic Bus Enumerator version: 3.1.0.3357
HID PCI Minidriver for ISS version: 3.1.0.3357

PDT Information :
PDT version: 4
Vendor defined data version: 0

Sensors Information :

Sensor LUID: 0073000100010002
Sensor Name: Motion Accelerometer 3D
Vendor: ST Micro
Sensor Model: LSM303D
Bus Type: I2C
Bus Address: 1d
Calibration Status: set

Sensor LUID: 0041000200010002
  
```



```
Administrator: C:\Windows\system32\cmd.exe - ISSU.exe -info -page
Bus Address: 48
Calibration Status: set

Sensor LUID: 0076000100040042
Sensor Name: Motion Gyrometer 3D
Vendor: ST Micro
Sensor Model: L3GD20H
Bus Type: I2C
Bus Address: 6b
Calibration Status: set

Sensor LUID: 020F000100010042
Sensor Name: Intel Oreintation
Vendor: ST Micro
Sensor Model: LSM303D
Bus Type: I2C
Bus Address: 1d
Calibration Status: set

Sensor LUID: 0073000100020102
Sensor Name: Motion Accelerometer 3D
Vendor: ST Micro
Sensor Model: LIS3DE
Bus Type: I2C
Bus Address: 29
Calibration Status: set

Sensor LUID: 0031000100060002
Sensor Name: Enviromental Atmospheric Pressure
press any key_
```

Figure 2-7. Example of -BOM Output

```
Administrator: C:\windows\system32\cmd.exe
C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -info -bom
Intel(R) Integrated Sensor Solution Utility Tool
Version: 3.0.0.1000

BOM Information:
Sensor LUID: 0073000100010002      Sensor Name: Motion Accelerometer 3D      Sensor Model: ST Micro, LSM303D
Sensor LUID: 0041000200010002      Sensor Name: Light Ambientlight           Sensor Model: Capella, CM32181
Sensor LUID: 0076000100040042      Sensor Name: Motion Gyrometer 3D         Sensor Model: ST Micro, L3GD20H
Sensor LUID: 020F000100010042      Sensor Name: Intel Motion Magnetometer 3D Sensor Model: ST Micro, LSM303D
Sensor LUID: 0073000100020102      Sensor Name: Motion Accelerometer 3D     Sensor Model: ST Micro, LIS3DE
Sensor LUID: 0031000100060002      Sensor Name: Environmental Atmospheric Pressure Sensor Model: ST Micro, LPS25H

C:\Users\BXT-P\Desktop\ISSU\Windows64>
```



Figure 2-8. Example of -SensorList Output

```

Administrator: C:\windows\system32\cmd.exe
C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -info -sensorlist iss.bin
Intel(R) Integrated Sensor Solution Utility Tool
Version: 3.0.0.1000

Type      Version      Name
Virtual   3.5.2.1      intel_sdo_sample
Virtual   3.5.2.3430   inclinometer
Virtual   3.5.2.3430   geomagnetic_orientation
Virtual   3.5.2.3430   device_orientation
Virtual   3.5.2.3430   compass
Virtual   3.5.2.3430   calibrated_gyro
Virtual   3.5.2.3430   calibrated_accelerometer
Virtual   3.5.2.3430   intel_accelerometer_shake
Virtual   3.5.2.3430   shake
Virtual   3.5.2.3430   device_movement
Virtual   3.5.2.3430   rotation_matrix
Physical   3.5.2.1      INTEL_MOTION_MAGNETOMETER_3D_ST_MICRO_LSM303D
Physical   3.5.2.1      MOTION_ACCELEROMETER_3D_ST_MICRO_LSM303D
Physical   3.5.2.1      MOTION_GYROMETER_3D_ST_MICRO_L3GD20H
Library   3.5.2.1      intel_calibrated_magnetometer_algorithm
Virtual   3.5.2.1      intel_calibrated_magnetometer_sample
Library   3.5.2.1      intel_sdo_algorithm

For more details, please check saved file: SensorList.xml

C:\Users\BXT-P\Desktop\ISSU\Windows64>

```

Note: The input binary file ISS.bin should be decompressed ISS binary instead of the compressed one. For details of how to get decompressed ISS binary, please refer to appendix B.

2.9 ISSU –PDT Functionality

Intel® ISS uses a configuration file called PDT. This file contains all the system topology and configuration and also the calibration parameters that are used by Intel® ISS. The PDT command allows you to get or set the PDT on the SPI flash.

Sample PDT files can be found in kit release\FW\bin\PDT\Bare_PDTs, and it's needed to add ".bin" extension when set the PDT binary.

In order to get the current PDT that is stored on the flash use:

Table 2-8. –PDT [-arguments]

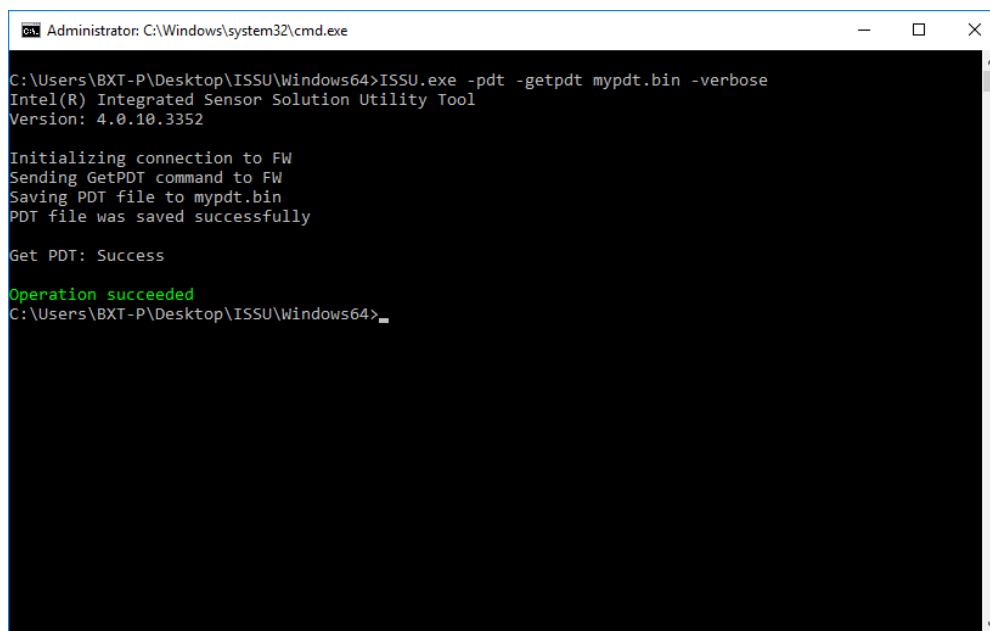
Option	Description
No option	Returns the help screen for ISSU –PDT
-H -?*	Returns the help screen for ISSU –PDT *Note: "-?" is not supported in EFI version
-EXP	Shows detailed examples of how to use the PDT command
-SetPDT <bin-file-path>	Writes the PDT from bin-path-file to ISH
-GetPDT <bin-file-path>	Reads the PDT file currently on the NVM and exports it to bin-file-path



Examples:

- ISSU -PDT -SetPDT <bin-file-path>
- ISSU -PDT -SetPDT <bin-file-path> -Verbose
- ISSU -PDT -GetPDT <bin-file-path>

Figure 2-9. Example of “GetPDT” Functionality



```
Administrator: C:\Windows\system32\cmd.exe

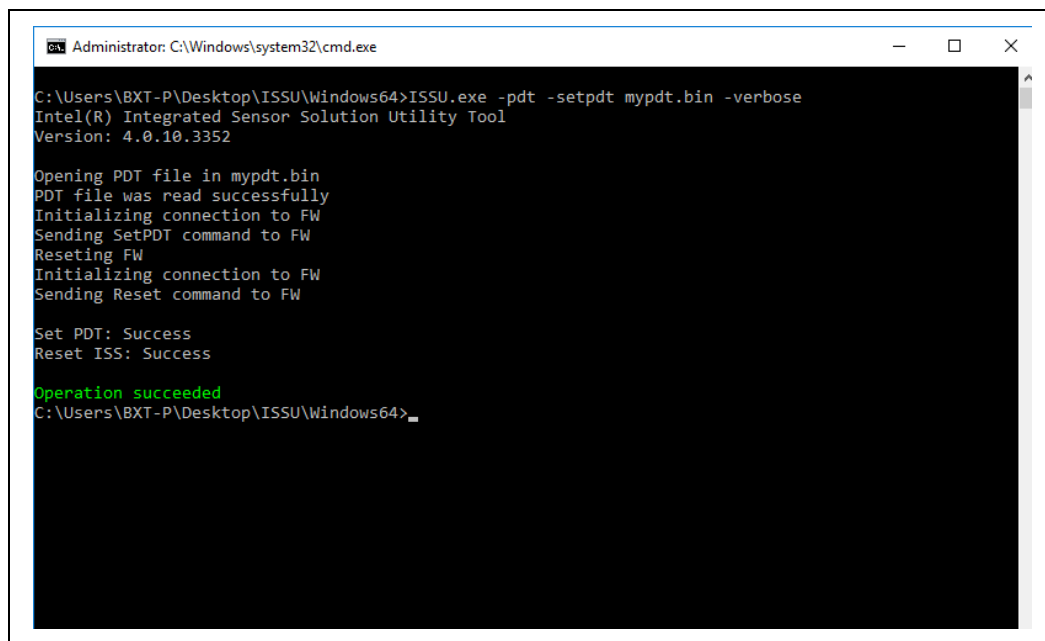
C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -pdt -getpdt mypdt.bin -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Initializing connection to FW
Sending GetPDT command to FW
Saving PDT file to mypdt.bin
PDT file was saved successfully

Get PDT: Success

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```

Figure 2-10. Example of “SetPDT” Functionality



```
Administrator: C:\Windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -pdt -setpdt mypdt.bin -verbose
Intel(R) Integrated Sensor Solution Utility Tool
Version: 4.0.10.3352

Opening PDT file in mypdt.bin
PDT file was read successfully
Initializing connection to FW
Sending SetPDT command to FW
Resetting FW
Initializing connection to FW
Sending Reset command to FW

Set PDT: Success
Reset ISS: Success

Operation succeeded
C:\Users\BXT-P\Desktop\ISSU\Windows64>
```



Note: In Windows and Linux, the tool support relative path for <bin-file-path>. E.g. if user specify only the file name "pdt.bin", it should be in current folder where you run the command. While in EFI, it only support absolute path for <bin-file-path>. E.g. if user specify only the file name "pdt.bin", it would in the root folder \pdt.bin. To put the binary file into specified folder, \<path_to_folder>\pdt.bin should be used as <bin-file-path> argument to the tool.

2.10 ISSU –I2C Functionality

Intel® ISS uses several I2C buses to communicate with sensors or output debug information. ISS provided functions to calibrate I2C bus speed. The tool would send specific I2C bus calibrating request to ISS and parse the result.

Table 2-9. –I2C [-arguments]

Option	Description
No option	Returns the help screen for ISSU –I2C
-H -?	Returns the help screen for ISSU –I2C *Note: "-?" is not supported in EFI version
-EXP	Shows detailed examples of how to use the I2C command
-Calibrate <busId>	Request to calibrate specific I2C bus and show the results
-UpdatePDT	Save I2C calibration results in PDT and reset ISS

Examples:

- ISSU –I2C –Calibrate <busid>
- ISSU –I2C –Calibrate <busid> -UpdatePDT

Figure 2-11. Example of "I2C Calibrate" Functionality

```

Administrator: C:\windows\system32\cmd.exe

C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -i2c -calibrate 1
Intel(R) Integrated Sensor Solution Utility Tool
Version: 3.0.0.1000

std      hcnt = 4490    lcnt = 5170
fast     hcnt = 710    lcnt = 1600
Calibrate I2C bus 1: Success

Operation succeeded

C:\Users\BXT-P\Desktop\ISSU\Windows64>

```



Figure 2-12. Example of “I2C calibrate and update PDT” Functionality

```
Administrator: C:\windows\system32\cmd.exe
C:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -i2c -calibrate 1 -updatepdt
Intel(R) Integrated Sensor Solution Utility Tool
Version: 3.0.0.1000

std      hcnt = 4490    lcnc = 5170
fast     hcnt = 710    lcnc = 1600
Calibrate I2C bus 1, save to PDT and reset ISS: Success

Operation succeeded

C:\Users\BXT-P\Desktop\ISSU\Windows64>
```

2.11 ISSU –NVM Functionality

Intel® ISS can store last 3 exceptions, 2 error log files and 2 user data files into NVM. The NVM command allows you to retrieve these files from the SPI flash and save to binary file, together with parsing exceptions to readable string.

In order to get/parse the files that are stored in NVM use:

Table 2-10. –NVM [-arguments]

Option	Description
No option	Returns the help screen for ISSU –NVM
-H -?*	Returns the help screen for ISSU –NVM *Note: “-?” is not supported in EFI version
-EXP	Shows detailed examples of how to use the NVM command
-GetErrorLog0 <bin-file-path>	Reads error log file 0 currently on the NVM and exports it to bin-file-path
-GetErrorLog1 <bin-file-path>	Reads error log file 1 currently on the NVM and exports it to bin-file-path
-GetUserData0 <bin-file-path>	Reads user data file 0 currently on the NVM and exports it to bin-file-path
-GetUserData1 <bin-file-path>	Reads user data file 1 currently on the NVM and exports it to bin-file-path



Option	Description
-GetExceptions <bin-file-path>	Reads the exceptions log file currently on the NVM and exports it to bin-file-path
-ParseExceptions <bin-file-path>	Parse the exceptions information in bin-file-path to readable string

Examples:

- ISSU –NVM -GetErrorLog0 <bin-file-path>
- ISSU –NVM -GetExceptions <bin-file-path>
- ISSU –NVM -ParseExceptions <bin-file-path>

Figure 2-13. Example of “GetExceptions”/“GetErrorLog0”/“GetUserData1” Functionality

```

Administrator: Command Prompt
c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -nvm -getexceptions esc.bin
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

Get Exceptions: Success

Operation succeeded

c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -nvm -geterrorlog0 err.bin
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

Get NVM file: No such file

Operation succeeded

c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -nvm -getuserdata1 err.bin
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

Get NVM file: Success

Operation succeeded

c:\Users\BXT-P\Desktop\ISSU\Windows64>
  
```



Figure 2-14. Example of "ParseExceptions" Functionality

```
Administrator: Command Prompt
c:\Users\BXT-P\Desktop\ISSU\Windows64>ISSU.exe -nvm -parseexceptions exc.bin
Intel(R) Integrated Sensor Solution Utility Tool
Version: STUBVERSIONSTR

number of exception entries in log : 2.

**** Exception 0 ****
timestamp: 15338680417902420 (0x367e72, 0xd5a1df54)
exception_number: 2
app: 0
reason: 22 - UNKNOWN
call stack: 3 frames
           0) 0x4a538
           1) 0x48b02
           2) 0xffff

**** Exception 1 ****
timestamp: 15344003171298180 (0x36834a, 0x22752b84)
exception_number: 1
app: 0
reason: 100 - ISH_WATCHDOG
call stack: 0 frames

Parse Exceptions: Success

Operation succeeded

c:\Users\BXT-P\Desktop\ISSU\Windows64>
```

Note: In Windows and Linux, the tool support relative path for <bin-file-path>. E.g. if user specify only the file name "exc.bin", it should be in current folder where you run the command. While in EFI, it only support absolute path for <bin-file-path>. E.g. if user specify only the file name "exc.bin", it would in the root folder \exc.bin. To put the binary file info specified folder, \"<path_to_folder>\exc.bin\" should be used as <bin-file-path> argument to the tool.



3 Appendix A: Error List

Here is the error list returned by the tool for BIST:

Error	Test Level *	Explanation	Possible Reason	Note
0: Test passed	any	Sensor self-test passed		
1: Test failed	any	Sensor self-test general failure		
2: Test unsupported	any	Sensor does not support the particular test		
3: Sensor not active	any	Sensor is not initialized or not ready	<ol style="list-style-type: none"> One of the LUID parts is not supported on platform There is a problem in PDT that causes sensor to fail in initialization 	
6: Missing calibration	2	Sensor has no appropriate calibration data	<ol style="list-style-type: none"> Sensor doesn't have calibration data configured Sensor's calibration data has wrong format 	For physical AGM/ALS sensors only
7: Connectivity failure	1	Sensor connectivity test failed		Returned only on CHT and SKL, Following generations use errors 9-11
9: GPIO BIOS configuration failure	1	Sensor GPIO BIOS configuration error	The sensor GPIO ID is not allocated for ISS in the BIOS	Replacing error 7 starting APL,KBL
10: GPIO connectivity failure	1	Sensor connectivity test failed for GPIO	Sensor supports GPIO configuration but fails to configure GPIO for receiving interrupt (GPIO ID that is not connected \ configured to the sensor)	Replacing error 7 starting APL,KBL



11: I2C connectivity failure	1	Sensor connectivity test failed on I2C	Sensor is disabled in switch	Replacing error 7 starting APL,KBL
------------------------------	---	--	------------------------------	------------------------------------

* Note that errors appearing in test level x will also appear in test levels greater than x since up to test level 3, the lower test levels are also called. I.e. test level 2 also calls tests 1 and 0 therefore an error that is relevant for test level 1 might appear when calling test level 2 or test level 3.

Here is the general error list returned by the tool:

0 : Success

1 : Command is currently not supported

2 : SMHI command failed

3 : Please wait, Working in progress

4 : SMHI error: Invalid parameter

8192 : General error

8193 : Cannot locate Intel® Integrated Sensor Solution driver

8194 : Memory access failure, please contact Intel support

8195 : Write register failure

8196 : Memory allocation failure, please contact Intel support

8197 : Circular buffer overflow

8198 : Not enough memory in circular buffer

8199 : Transmission error to Intel® Integrated Sensor Solution

8200 : Unsupported TXE bus message protocol version

8201 : Unexpected interrupt reason

8202 : Intel® Integrated Sensor Solution timeout occurred

8203 : Unexpected result in Intel® Integrated Sensor Solution response, please contact Intel support

8204 : Unsupported message type

8205 : Cannot find host client

8206 : Cannot find Intel® Integrated Sensor Solution client



- 8207 : Client already connected
- 8208 : No free connection available
- 8209 : Illegal parameter
- 8210 : Flow control error
- 8211 : No message
- 8212 : Requesting Intel® Integrated Sensor Solution receive buffer size is too large
- 8213 : Requesting Intel® Integrated Sensor Solution receive buffer size is too small
- 8214 : Circular buffer not empty
- 8215 : Cannot access Intel® Integrated Sensor Solution device
- 8216 : Invalid command-line arguments
- 8217 : Failed to read or write file
- 8218 : Invalid PDT file length
- 8219 : Invalid PDT file extension, PDT format is xxx.bin
- 8220 : Failed to get driver version
- 8221 : Unsupported OS
- 8222 : No required privileges, run as Administrator
- 8223 : Invalid sensor luid format
- 8224 : Failed to get device address
- 8225 : Failed to load driver (PCI access for windows)
- 8226 : Bist failed
- 8227 : Could not execute BIST: no sensors configured on the device. Please check the PDT configuration
- 8228 : Unknown response from SMHI
















4 Appendix B: Get Decompressed ISS Binary File

Use FIT tool to get compressed ISS binary from IFWI:

```
> fit.exe -f InputIFWI_xxx.bin -save Decompose.xml
```

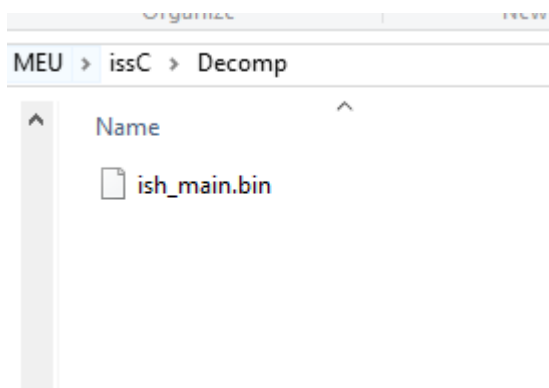
Find ISHC.bin in Decomp folder:

APL_SPI_RVP1A_NS_BOM1 > Decomp	
Name	Date
 BIOS Region.bin	6/8/21
 CEKBinary.bin	6/8/21
 Descriptor Region.bin	6/8/21
 IntelTrcHubBinary.bin	6/8/21
 ISHC.bin	6/8/21
 IUnit Region.bin	6/8/21
 oem.key.bin	6/8/21
 PdtBinary.bin	6/8/21
 PMC Region.bin	6/8/21
 Smip Region.bin	6/8/21
 TXE Region.bin	6/8/21
 uCode Patch 1.bin	6/8/21
 UnlockToken.bin	6/8/21

Use MEU tool to get decompressed ISS binary:

```
> meu -decomp CodePartition -f ISHC.bin -save iss.xml
```

The get the ish_main.bin which is uncompressed ISS binary:



Specify Lzma binary path "LzmaToolPath" in meu_config.xml, to tell meu.exe how to decompose ISHC.bin.

An example of meu_config.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<MeuConfig version="2.10" >
  <PathVars label="Path Variables">
    <WorkingDir value="." label="$WorkingDir" help_text="Path for environment
variable $WorkingDir" />
    <SourceDir value="." label="$SourceDir" help_text="Path for environment
variable $SourceDir" />
    <DestDir value="." label="$DestDir" help_text="Path for environment variable
$DestDir" />
    <UserVar1 value="." label="$UserVar1" help_text="Path for environment
variable $UserVar1" />
    <UserVar2 value="." label="$UserVar2" help_text="Path for environment
variable $UserVar2" />
    <UserVar3 value="." label="$UserVar3" help_text="Path for environment
variable $UserVar3" />
  </PathVars>
  <SigningConfig label="Signing Configuration">
    <SigningTool value="OpenSSL"
value_list="Disabled,,OpenSSL,,MobileSigningUtil" label="Signing Tool"
help_text="Select tool to be used for signing, or disable signing." />
    <SigningToolPath value="$SourceDir\..\openssl\openssl.exe" label="Signing
Tool Path" help_text="Path to signing tool executable." />
  </SigningConfig>
</MeuConfig>
```



```
<SigningToolXmlPath value="" label="Signing Tool Config XML Path"
help_text="Configuration XML template for MobileSigningUtil. Leave blank if not using
MSU." />
```

```
<SigningToolExecPath value="" label="Signing Tool Execution Path"
help_text="Specify a directory from which the signing tool should be executed. This
can be useful if relative paths are used in the Signing Tool Config XML. If no path is
provided, the signing tool will be executed from the same directory as this tool was
executed. Leave blank if not using MSU." />
```

```
</SigningConfig>
```

```
<CompressionConfig label="Compression Configuration">
```

```
<LzmaToolPath value="$SourceDir\..\lzma\lzma.exe" label="LZMA Tool Path"
help_text="Path to lzma tool executable." />
```

```
</CompressionConfig>
```

```
</MeuConfig>
```

§ §