Firmware Resiliency Features in ADL-N

Intel® Slim Bootloader (SBL) Features Addendum

Date: February 2023

Intel Confidential

Contents

Tables

Figures

Revision History

Date	Revision	Description
January 2023	0.5	Initial release

§

1.0 Introduction

1.1 Purpose

The Intel® Slim Bootloader (SBL) firmware resiliency document explains the recovery and update functionalities of Slim Bootloader.

To learn more in detail about firmware resiliency, see:

https://slimbootloader.github.io/security/firmware-update.html.

https://slimbootloader.github.io/security/security/firmware-resiliency-and-recovery.html

1.2 Reference Documents

Table 1 Related Documents

Document	Document ID
Slim Bootloader Project Documentation	https://slimbootloader.github.io/
Firmware Update in Intel® Slim Bootloader (SBL)	https://slimbootloader.github.io/security/firmware-update.html
Firmware Recovery in Intel® Slim Bootloader (SBL)	https://slimbootloader.github.io/security/security/firmware-resiliency-and-recovery.html

2.0 Supported Features

2.1 Update Features

The following regions are updatable via Slim Bootloader firmware update:

- Slim Bootloader
- uCode region
- ACM region
- CSME region
- Configuration region
- Whole container region
- Container component

2.2 Recovery Features

The following regions are recoverable via Slim Bootloader recovery:

- Stage 1A
- ACM
- uCode
- Stage 1B
- Key Hash
- Config Data
- FW Update Payload
- Stage 2

If boot progress is held back by any of the above components and firmware recovery is enabled, Slim Bootloader will try to fix the boot by copying over the contents of the alternate boot partition.

3.0 Capsule Generation

The capsule generation tool (GenCapsuleFirmware.py) creates a capsule image that can be processed by SBL in its firmware update flow.

Note: GenCapsuleFirmware.py is available in BootloaderCorePkg/Tools/.

The capsule generation tool can incorporate multiple firmware images into single capsule binary.

usage: GenCapsuleFirmware.py [-h] -p PAYLOAD PAYLOAD -k PRIVKEY [-a {SHA2_256,SHA2_384,AUTO}] [-s {RSA_PKCS1,RSA_PSS}] -o NEWIMAGE [-v] [-f]

optional arguments:

-h	Show help message and exit
-p PAYLOAD PAYLOAD	Payload string and its associated file name
-k PRIVKEY	Key ID or private RSA 2048/RSA3072 key in *.pem format for image signing
-a {SHA2_256,SHA2_384,AUTO}	Hash type for image signing (AUTO hash type will be chosen based on key length)
-s {RSA_PKCS1,RSA_PSS}	Signing scheme types
-o NEWIMAGE	Output file name for signed capsule image
-v	Turn on verbose output with informational messages printed, including capsule headers and warning messages
-f	Force update of whole BIOS region in a single shot

Each non-containerized component inside a capsule image is stored using a unique 4-byte string. While using the tool for capsule generation, this 4-byte string and its associated binary should be provided as input to the tool.

Each containerized component inside a capsule image is stored using 2 unique 4-byte strings: one for the component inside the container and one for the container itself. While using the tool for capsule generation, these 2 4-byte strings and their associated binary should be provided as input to the tool.

The following table explains the string used for the components.

Table 2 Component String ID

ID	Component
BIOS	Slim Bootloader
UCOD	Microcode component binary
ACM0	ACM component binary
CSME	CSME update binary
CSMD	CSME driver for updating CSME region
CNFG	Configuration data region binary
IPFW	Container region binary
TCCC	TCC cache configuration container component region
TCCM	TCC CRL container component region
TCCT	TCC streams-tuning container component region
TMAC	TSN MAC address container component region binary

The following example command creates a capsule image for firmware update in Slim Bootloader containing Slim Bootloader binary (sbl.bin).

```
python BootloaderCorePkg/Tools/GenCapsuleFirmware.py -p BIOS
sbl.bin
```

The following example command creates a capsule image for firmware update in Slim Bootloader containing CSME update binary (csme_update.bin) and CSME update driver (csme_update_driver.bin). Note that both items are required for successful CSME update.

```
python BootloaderCorePkg/Tools/GenCapsuleFirmware.py -p CSME
csme_update.bin -p CSMD csme_update_driver.bin -k
SblKeys/ContainerTestKey_Priv_RSA3072.pem -o FwuImage.bin
```

The following example command creates a capsule image for container component TSN MAC Address (signed_tsn_mac_addr.bin).

```
python BootloaderCorePkg/Tools/GenCapsuleFirmware.py -p TMAC:IPFW
signed_tsn_mac_addr.bin -k
SblKeys/ContainerTestKey_Priv_RSA3072.pem -o FwuImage.bin
```

The following sections describe how to generate/obtain individual component binaries for capsule generation.

3.1 Stitching Slim Bootloader binary

Leverage stitching to integrate external binaries (e.g. ACM, TCCC, etc.) as Slim Bootloader components, and update BtG manifests. Please see section 6.0 (Slim Bootloader Stitching) of Intel® Slim Bootloader (SBL) Firmware Release Notes to understand how to stitch using Slim Bootloader.

After stitching, the Slim Bootloader binary is SlimBootloader_adln.bin (at the root of the repo).

3.2 Generating uCode binary

Slim Bootloader can update its uCode region.

The uCode utility tool (UcodeUtility.py) can help create a binary that can be used for updating the entire uCode region.

Note: UcodeUtility.py is available in BootloaderCorePkg/Tools/.

usage: UcodeUtility.py [-h] -s SLOT_SIZE -i INPUT_FILE_NAMES [INPUT_FILE_NAMES ...] -o OUTPUT_FILE_NAME

optional arguments:

-h	Show help message and exit
-s SLOT_SIZE	uCode slot size (in bytes)
-i INPUT_FILE_NAMES [INPUT_FILE_NAMES ...]	uCode patch binary names (*.pdb or *.mcb files)
-o OUTPUT_FILE_NAME	uCode region binary name (output file name)

The following example command creates a uCode region binary (ucode_reg.pad) with 0x3B000 slot size and 2 uCode patch binaries (ucode_patch_1.mcb and ucode_patch_2.mcb).

```
python UcodeUtility.py -s 0x3B000 -i ucode_patch_1.mcb
ucode_patch_2.mcb -o ucode_reg.pad
```

3.3 ACM region binary

Slim Bootloader can update its ACM region.

The binary for the ACM region (e.g. acm0.bin) can be taken directly from the stitching ingredients.

3.4 CSME region update

To update CSME region using Slim Bootloader, a CSME firmware update binary and a driver that understands how to perform CSME region update are needed.

3.4.1 Generating CSME update driver

CSME update binary is available as part of CSME releases. Please see section 6.2 (Gathering FW Ingredients) of Intel® Slim Bootloader (SBL) Firmware Release Notes to understand how to get CSME package. CSME update binary is available at Tools/System_Tools/FWUpdate_RS/sbl32/FwUpdateEfiLib.lib in the CSME package.

- Copy library FwUpdateEfiLib.lib to Silicon/AlderlakePkg /Library/MeFwUpdateLib/MeFwUpdateLib.lib.
- Set BUILD_CSME_UPDATE_DRIVER to 1 in BoardConfigAdlN.py.

Note: Generating CSME update driver feature is only available as part of Windows SBL build.

- Build Slim Bootloader. Please see section 5.6 (Building Slim Bootloader) of Intel® Slim Bootloader (SBL) Firmware Release Notes to understand how to build Slim Bootloader.
- CSME Update driver will be available at Build/BootloaderCorePkg/DEBUG_VS2019/IA32/CsmeUpdateDriver.efi.

3.4.2 Generating CSME update binary

Generating CSME Update binary requires Intel® Modular Flash Image Tool (MFIT). The MFIT tool is part of CSME package. Please see section 6.2 (Gathering FW Ingredients) of Intel® Slim Bootloader (SBL) Firmware Release Notes to understand how to get CSME package. MFIT tool is available in Tools/System_Tools/MFIT/ inside CSME package.

The following steps describe how to generate CSME firmware update binary

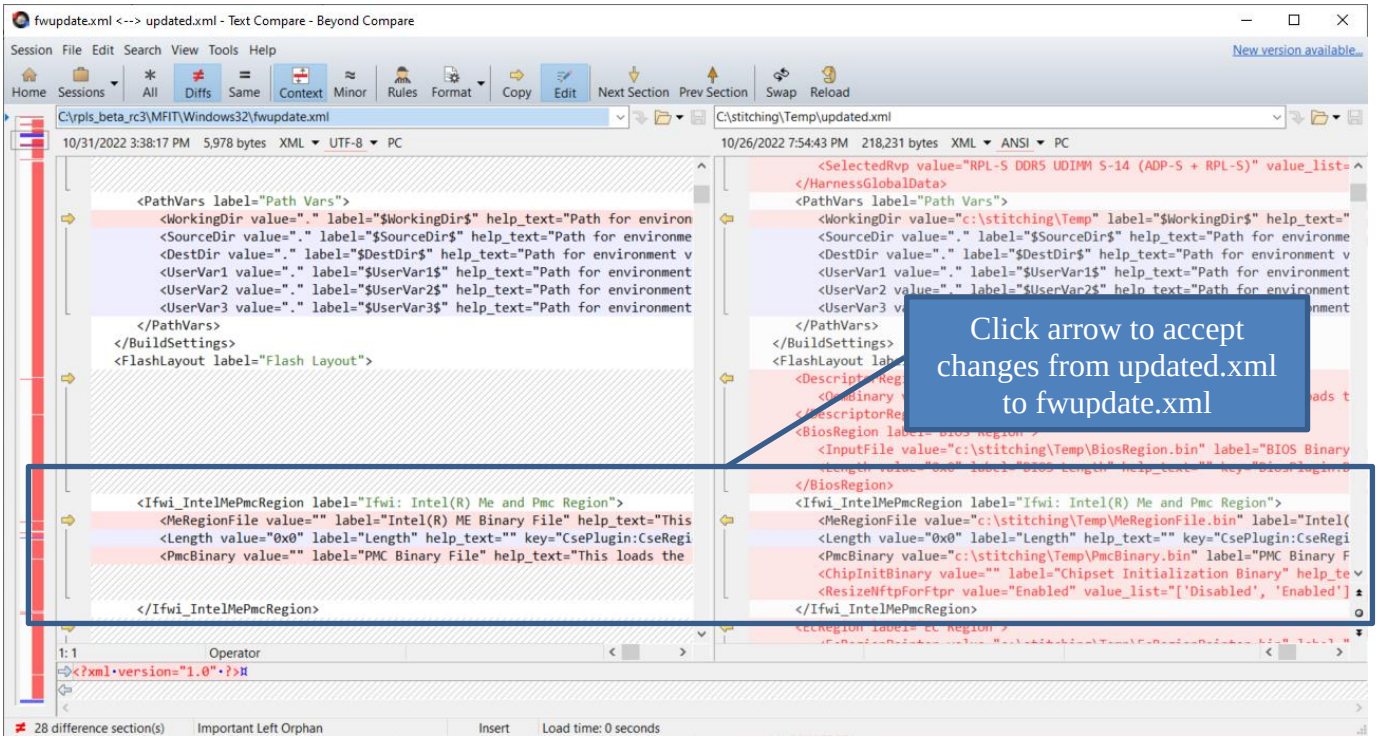
1. Open CMD window.
2. Go to path were MFIT tool is located.
3. Type the following command to generate fwupdate.xml:

```
mfit.exe  -l "Intel(R) AlderLake N Chipset - FWUpdate" -s
fwupdate.xml
```

4. Use a diff tool like Beyond Compare to compare fwupdate.xml and updated.xml (generated during SBL stitch).

5. Copy over the file paths for the regions below from updated.xml to fwupdate.xml. These items are found in the stitching ingredients.
    - MeRegionFile,
    - PmcBinary,
    - PchcSubPartitionData,
    - IshImage,
    - PhyBinaryFile
6. Save the updated paths for the listed regions in fwupdate.xml.
7. Type the following command to build CsmeFwUpdate.bin:

```
mfit.exe -l "Intel(R) AlderLake N Chipset - FWUpdate" --
loadconfig fwupdate.xml --build CsmeFwUpdate.bin
```



Slim Bootloader can update component regions inside containers in the BIOS region. These components are encapsulated using special headers and format inside the container and cannot be updated using a direct region binary.

The following table details some common container components present in SBL.

Table 3 IPFW Component Region Signatures

Component Signature	Component Description
TCCC	TCC cache configuration container component region
TCCM	TCC CRL container component region
TCCT	TCC streams-tuning container component region
TMAC	TSN MAC address container component region binary

The container generation tool (GenContainer.py) can help sign and create a component binary that can be used for updating a specific component region inside the container.

Note: GenContainer.py is available in BootloaderCorePkg/Tools/.

Use the following command to create signed component for capsule.

usage: GenContainer.py sign [-h] -f COMP_FILE [-o OUT_FILE] [-c {lz4,lzma,dummy}] [-a {SHA2_256,SHA2_384, RSA2048_PKCS1_SHA2_256,RSA3072_PKCS1_SHA2_384, RSA2048_PSS_SHA2_256,RSA3072_PSS_SHA2_384,NONE}] [-k KEY_FILE] [-td TOOL_DIR] [-s SVN]

optional arguments:

-h	Show help message and exit
-f COMP_FILE	Component input file path
-o OUT_FILE	Signed output image path
-c {lz4,lzma,dummy}	Compression algorithm
-a {SHA2_256,SHA2_384,RSA2048_PKCS1_SHA2_256, RSA3072_PKCS1_SHA2_384,RSA2048_PSS_SHA2_256, RSA3072_PSS_SHA2_384,NONE}	Authentication algorithm
-k KEY_FILE	Key ID or private key file path to sign component
-td TOOL_DIR	Compression tool directory
-s SVN	Security version number for component

The following example command generates a signed container component from a TSN MAC address (tsn_mac_addr.bin).

```
python BootloaderCorePkg/Tools/GenContainer.py sign -f
tsn_mac_addr.bin -o signed_tsn_mac_addr.bin -a
RSA3072_PSS_SHA2_384 -k
SblKeys/ContainerCompTestKey_Priv_RSA3072.pem -c lz4 -td
BaseTools/Bin/Win32
```

3.6 Configuration Data region binary

Components inside the BIOS region are often padded to certain alignment and size.

The configuration data region inside SBL is padded at build time. CFGDATA.pad is the padded file of configuration data that gets generated during Slim Bootloader build. This should be used in FW update capsules.

After building Slim Bootloader, CFGDATA.pad file is available in Build/BootloaderCorePkg/DEBUG_VS2019/FV/.

4.0 Capsule Location

Intel® Slim Bootloader (SBL) by default will look for the capsule image “FwuImage.bin” at the /boot/efi directory of a USB key.

Capsule location and name of the capsule can be customized from Slim Bootloader CAPSULE_INFO_CFG_DATA configuration data.

5.0 Triggering Firmware Update

Slim Bootloader supports triggering update from Windows and Linux operating systems and Slim Bootloader shell.

To support triggering firmware update from operating system, SBL exposes an ACPI method. Operating system can call these ACPI methods to trigger firmware update. Following a reset, Slim Bootloader boots into firmware update mode.

5.1 Triggering firmware update from Windows

The following sample script uses WMI to call ACPI methods exposed by Slim Bootloader to trigger firmware update.

```
set Service = GetObject("winmgmts:root/wmi")
set EnumSet = Service.InstancesOf("AcpiFirmwareCommunication")
for each Instance in EnumSet
  Wscript.Echo "Current  Val: " & Hex(instance.Command)
  instance.Command = 1
  instance.Put_()
  Wscript.Echo "Set New Val: " & Hex(instance.Command)
next 'instance
```

To use this method to trigger firmware update:

1. Copy above sample code to *.vbs file.
2. Open a command line window with administrator rights on SUT.
3. Run *.vbs script.
4. Reset the system.

5.2 Triggering firmware update from Linux (Yocto)

If your Linux kernel includes the Kconfig option INTEL_WMI_SBL_FW_UPDATE you can trigger a firmware update with the command below followed by restarting the system:

```
echo 1 > /sys/bus/wmi/devices/44FADEB1-B204-40F2-8581-
394BBDC1B651/firmware_update_request

reboot
```

5.3 Triggering firmware update from SBL Shell

Execute the following command from SBL shell to trigger firmware update:

```
fwupdate
```

6.0 Enabling Recovery

Firmware recovery is not enabled by default. It must be enabled through a variety of steps taken at build and stitch time.

To enable recovery:

1. Enable SBL resiliency in board configuration file.
2. Enable BIOS redundancy assistance in stitch configuration file.

Note that recovery requires SBL to be built with BtG profile 5 (FVME).
Also, note that recovery and FSP debug cannot be enabled at the same time.

Example:

1. In BoardConfigAdlN.py set ENABLE_SBL_RESILIENCY.

```
self.ENABLE_SBL_RESILIENCY = 1
```

2. In StitchIfwiConfig_adln.py enable BiosRedAssistance.

```
('./FlashSettings/BiosConfiguration/BiosRedAssistance',
            'Enabled'),
```

3. Build without -fd parameter.

```
python BuildLoader.py build adln
```

4. Stitch with -b fvme parameter.

```
python Platform/AlderlakeBoardPkg/Script/StitchIfwi.py -p adln -w
Stitchifwi_components_adln -c
Platform/AlderlakeBoardPkg/Script/StitchIfwiConfig_adln.py -s
Outputs/adln/SlimBootloader.bin -b fvme -o crb
```

7.0 Triggering Firmware Recovery

Firmware recovery is triggered whenever any of Slim Bootloader's components halt a boot's progress.

To make a Slim Bootloader component halt a boot's progress, a crash, dead loop, or corruption must be introduced into an IFWI or a FW update capsule and pushed to the platform.