Modular Flash Image Tool

User Guide

July 2020

Revision: 0.33

Intel Confidential

Tables

Revision History

Table 1. Revision history

Revision Number	Description	Revision Date
0.1	Initial version	June 2019
0.2	Command line changes	March 2020
0.3	Names updates	April 2020
0.31	Update	May 2020
0.32	Decomposition section removed	May 2020
0.33	Add key lifetime chapter	July 2020

1 Introduction

The purpose of this document is to describe usage of Modular Flash Image Tool - Modular FIT - including system requirements, proper configuration and common use cases.

2 Overview

2.1 Modular Flash Image Tool description

Modular Flash Image Tool creates and configures a complete SPI image file for platforms in the following way:

- Creates and allows configuration of the Flash Descriptor Region, which contains configuration information for platform hardware and FW.
- Exposes multiple ME related settings.
- Assembles binary region files into a single SPI flash image.

Manipulation of the SPI image is available via configuration XML which enables changes of various chipset parameters in order to match the target hardware. All settings can be saved to independent files, so it is not required to recreate a new image configuration at each build.

Modular FIT supports a set of command line parameters that can be used to build an image from the CLI or from a batch file. Almost all settings configured via XML can also be overwritten with usage of CLI interface.

Modular FIT generates a complete SPI image file, it does not program the flash device. The complete SPI image must be programmed into the flash with any flash burning tool or some other flash programmer device.

2.2 System Requirements

Modular FIT is available as:

- python script
- standalone executable file.

In both cases the tool does not have to be run on an Intel ME-enabled system.

In case of the executable file all required libraries and dependencies are already integrated into the package and no further setups of user environment configuration are needed. All usage examples described in this document will be presented in respect to executable version of the tool.

Rest of this chapter summarizes requirements for using script version of Modular FIT. The tool requires machine with Python 3.6 or later installed. Table 2 describes external packages that must be added to python libraries.

Table 2. Required python modules

Name	How to get
colorama *[optional]*	Can be installed with python package installer *(optional library, responsible for coloring errors and warning in CLI):* *pip install colorama*
cryptography	Can be downloaded directly from: https://pypi.python.org/pypi/cryptography or installed with python package installer: *pip install* cryptography
lxml	Can be downloaded directly from: https://pypi.python.org/pypi/lxml or installed with python package installer: *pip install lxml*
crcmod	Can be installed with python package installer: *pip install crcmod*
Yapsy	Can be installed with python package installer: *pip install yapsy*
Pywin32	Can be installed with python package installer: *pip install pywin32* (WINDOWS ONLY)
Pywin32-Ctypes	Can be installed with python package installer: *pip install pywin32-ctypes* (WINDOWS ONLY)

All required modules are also listed in requirements_linux.txt or requiremends_windows.txt files and can be installed with one command:

i.e.:

pip install -r requirements_windows.txt

2.2.1 Security requirements

Modular FIT also considers runtime integrity protection:

1. Checking access rights of a tool root and all nested folders/files. If user receives following message, it means that unexpected access rights to some of the files were found:

```
Wrong access rights for: <path>

For further information, please refer to User Guide or run application as
Administrator/root with 'set_dir_acl' argument only to set required Access
Control List.

Otherwise, run application with 'skip_access_check' argument to skip access
rights check.
```

2. Allowing absolute paths only for plugins folder as parameter, next to the executable/script file by default and do access rights check (*--plugins_dir*). The following error may occur:

   ```
   Plugins directory must be an absolute path, but given relative: '<path>'
   ```

3. Checking access rights on parsing Modular FIT parameters:

   a. *plugins_dir* parameter value

   b. *layout_xml* parameter value

4. Allowing to bypass such restrictions by running Modular FIT with *–skip_access_check* with below warning:

   ```
   Warning: accepting relative plugins dir and skipping checking access rights
   for any directory.
   ```

5. Allowing to set all required permissions by running Modular FIT with *–set_dir_acl*.

   NOTE*: This feature requires to run Modular FIT with administrator/root privileges.*

In sake of security, communication of the plugins and core is verified in runtime by Modular FIT. This means, that only privileged users can modify any data in Modular FIT package. Therefore, it is required to define a permissions configuration so that no one except whitelisted users/user groups can use and change data (code, plugins, etc.). This whitelist is called a Modular FIT Access Control List.

Compliance with Modular FIT Access Control List is done in both Modular FIT packages:

- python script
- standalone executable file.

Access Control List is a list of whitelisted user groups which represents user that have specific permissions to data in Modular FIT package. Those groups are represented by well-known security identifiers in Windows operating systems.

Table 3. Windows Access Control List

Name	SID	Description
Builtin Administrators	S-1-5-32-544	A built-in group. After the initial installation of the operating system, the only member of the group is the Administrator account. When a computer joins a domain, the Domain Administrators group is added to the Administrators group. When a server becomes a domain controller, the Enterprise Administrators group also is added to the Administrators group.

Domain Administrators	S-1-5-21-<domain>-512	A global group whose members are authorized to administer the domain. By default, the DOMAIN_ADMINS group is a member of the Administrators group on all computers that have joined a domain, including the domain controllers. DOMAIN_ADMINS is the default owner of any object that is created by any member of the group.
Authenticated Users	S-1-5-11	A group that includes all users whose identities were authenticated when they logged on.

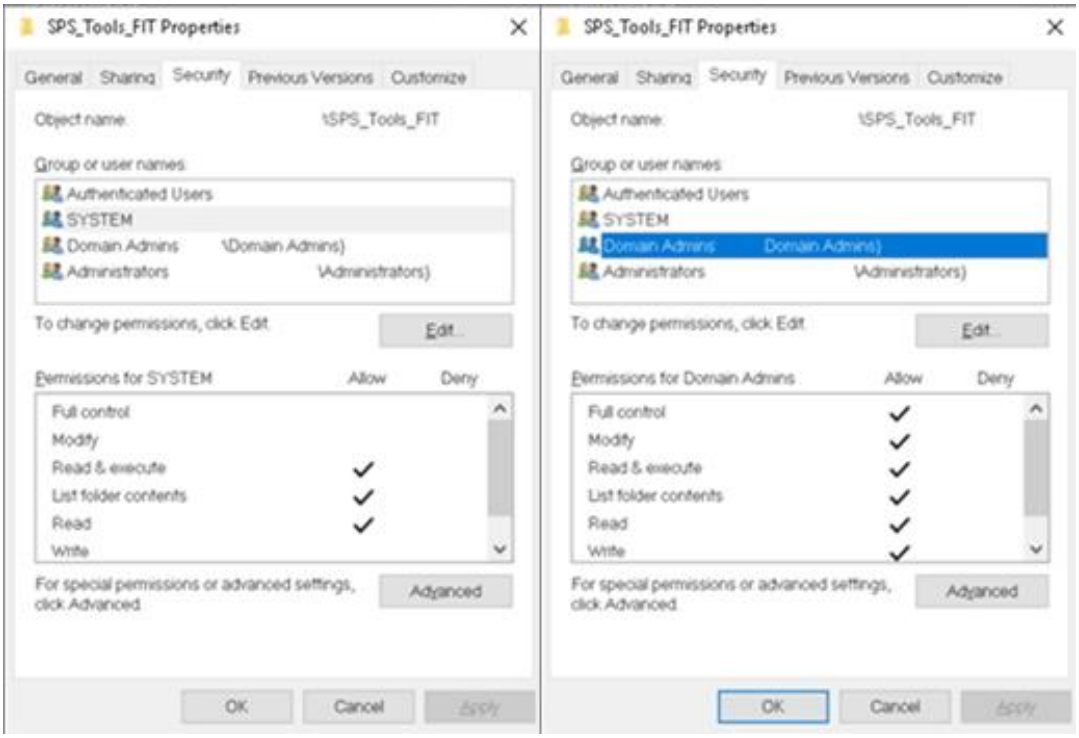
Access Control Entry of any directory (folder/file) in Modular FIT package – as well root directory - with any of the whitelisted SIDs can hold specific Generic Access Rights. Any other ACE with non-whitelisted SID should not have any permission from the list below defined in access mask format accepted by Modular FIT. Only by ACEs with whitelisted SID.

Table 4. Windows Access Permissions

Constant Name	Generic meaning
GENERIC_ALL	All possible access rights
GENERIC_WRITE	Write access
FILE_WRITE_DATA	File write access
FILE_APPEND_DATA	File append data

IMPORTANT:
Any other Access Control Entry found in ACL with non-whitelisted SID should not have above.

User can manually set required permissions by editing Security settings of Modular FIT package:



Additionally, this can be set with the tool itself by running Modular FIT with *–set_dir_acl* parameter.

NOTE:
This feature requires Administrator privileges.

Linux user environment differs from Windows, that is why this can be optimized by verifying specific bit permission of that folder/file.

Table 5. Permission bitmasks

Bit name	Octal / Binary Mode Number	Description
User sticky bit (s) mask	4000 / 0b100000000000	The ability to inherit USER permissions from parent directory.
Group sticky bit (s) mask	2000 / 0b010000000000	The ability to inherit GROUP permissions from parent directory.
Others write (w) bit mask	0002 / 0b000000000010	The ability to rename files in the directory, create new files, or delete existing files by other users.

IMPORTANT:
Others write (o-w) bit should be cleared to prevent other users from any file manipulation as only root and authenticated users should have permissions. Additionally, user and group sticky bits should be set to provide permission inheritance for any child folder/file and any newly generated object.

User can manually set required permissions with Linux built-in tool *chmod* for all folders and files including root directory of the tool's executable/script file:

chmod u+s <path>

chmod g+s <path>

chmod o-w <path>

Described procedure can be done automatically as well with Modular FIT by running the tool with *–set_dir_acl* parameter only.

NOTE:
This feature requires root privileges.

Installation for Windows

- Install python
- It may be necessary to update PIP with command:
  - *c:\Python36\python.exe -m pip install --upgrade pip*
- Install all required modules as listed in Table 2, this can be done with command:
  - *c:\Python36\python.exe -m pip install -r requirements.txt*

2.3 Required files

The Modular FIT main file is Modular FIT_cmd.exe amd it must be present along with all other files in the package.

There are mandatory external binary requirements in order to build an output image using Modular FIT. Table lists those required files. It's important to note that when specifying a path to each required file the working directory is the main directory with FITm_cmd.exe. Full paths can also be used.

Table 6. Required files

Required files	Xml setting name	CLI parameter	Comment
ME Region	input_file	--params me: input_file=<path>	**Provided with FW package as "MeRegion.bin"**
Pmc binary	PMCBinary	--params pmc: PMCBinary=<path>	
Bios	"binary" attribute of "bios" container	--layout bios=<path>	

2.4 Command Line Interface

```
FITm_cmd.exe [-b] [-d PLUGINS_DIR] [-l LAYOUT_XML]
                [-i INPUT] [-s SAVE_XML] [-h [HELP]]
                [--license [LICENSE]]
                [--params PARAMS [PARAMS ...]] [--verbose]
                [--layout LAYOUT [LAYOUT ...]]
                [--skip_access_check]
                [--set_dir_acl]
```

Table 7. Command Line Options

Option	Description
-b, --build	Triggers building a binary image
-d, --plugins_dir	Specifies plugins directory
-l layout_xml	Specifies kind of platform layout to be chosen
-i, --input	Specifies path for the user configuration xml or a binary file to decompose
-s, --save_xml	Specifies path where to save the configuration xml
-h, --help	Shows this help or overridable setting list for the specified container
--license	Shows license agreement information
--params	Specifies settings to be overridden. Sample definition: container:setting=0x1 FilePath=input.bin
--verbose	Displays additional information during tool execution
--layout	Configures a binary file to be used for a specified container
--skip_access_check	Allows user to provide plugins directory as a relative path and omit directories access rights check
--set_dir_acl	Sets required ACL for each item in root directory

Not every command line switch can be joined with any other. Possible arguments combinations are displayed by Modular FIT help screen and they are presented in the table below:

Table 8. Possible argument combinations

Argument used:	Limits the rest of possible arguments to set of:
'—build'	'--input', '--plugins_dir', '--layout_xml', '--save_xml', '--params', '--layout', '--verbose' '—skip_access_check'
'—help'	
'--input'	'--verbose', '--save_xml', '--layout', '—skip_access_check'
'--license'	
'--save_xml'	'--input', '--plugins_dir', '--layout_xml', '--params', '--layout', '--verbose' '—skip_access_check'
'--gui'	'--port' '—skip_access_check'
'—set_dir_acl'	

2.4.1 Basic usage of Modular FIT in Command Line Interface

Table 9. Build default image

Use Case	**Build default image**
CLI parameters	FITm_cmd.exe –b –params <set_paths_to_firmware –- layout:bios=<file_path>
Example	*FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin --layout bios=fw\bios.rom*
SPS FITc comparison	*spsFITc.exe -b -me MeRegion.bin -pmcp pmcp.bin –bios firmware\BIOS.bin*

Table 10. Override setting parameters

Use Case	Override setting parameters
CLI parameters	FITm_cmd.exe -b --params <container_name>:<setting_name>=<setting_value>
Example	*FITm_cmd.exe -b **--params** me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin **descriptor:NumberOfSpiComponents=1** --layout bios=fw\bios.rom*
SPS FITc comparison	*spsFITc.exe -b -me MeRegion.bin -pmcp pmcp.bin **-flashcount 1** –bios firmware\BIOS.bin*

Additional information:

The *params* switch allows to override several settings in a single call. To define multiple overrides for a region simply separate each setting with a space. Same rule can be applied to define parameter overrides for multiple regions.

FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin idlm_file=fw\idlm.bin pmc:PMCBinary=fw\pmcp.bin descriptor:FlashComponent1Size=0x2000000

Table 11. Show help screen

Use Case	Show help screen
CLI parameters	FITm_cmd.exe -h
Example	*FITm_cmd.exe -h*
SPS FITc comparison	*spsFITc.exe -h*

Table 12. Show configurable settings for specified region

Use Case	Show configurable settings for specified region
CLI parameters	FITm_cmd.exe –h <container_name>
Example	*FITm_cmd.exe –h descriptor*
SPS FITc comparison	Not applicable. General help screen displays all settings that can be changed for all regions: *spsFITc.exe -h*

Table 13. Save configuration to user xml

Use Case	**Save configuration to user xml**
CLI parameters	FITm_cmd.exe *-s <fitm_configuration_file>*
Example	FITm_cmd.exe ***-s configuration.xml*** *--params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin --layout bios=fw\bios.rom*
SPS FITc comparison	*spsFITc.exe -save configuration.xml -me MeRegion.bin -pmcp pmcp.bin –bios firmware\BIOS.bin*

Note: there is no option yet to save simple xml in Modular FIT.

Table 34. Load configuration and build image

Use Case	**Load configuration and build image**
CLI parameters	FITm_cmd.exe *-b -i <fitm_configuration_file>*
Example	FITm_cmd.exe *-b -i configuration.xml*
SPS FITc comparison	*spsFITc.exe -b -f configuration.xml*

Note: If some settings are missing, the default values are used.

Table 45. Override regions with specified binaries

Use Case	**Override regions with specified binaries**
CLI parameters	FITm_cmd.exe *-b --layout <container_name>=<path_to_binary>*
Example	FITm_cmd.exe *-b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***--layout bios=fw\bios.rom gbe=gbe.bin***
SPS FITc comparison	*spsFITc.exe -b -me MeRegion.bin -pmcp pmcp.bin* ***–bios firmware\BIOS.bin -gbe gbe.bin***

Table 56. Override default output binary name and build image

Use Case	Override default output binary name and build image
CLI parameters	FITm_cmd.exe -b --params layout:output_file=<newName>
Example	FITm_cmd.exe *–b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***layout:output_file=newName.bin*** *--layout bios=fw\bios.rom*
SPS FITc comparison	*spsFITc.exe -b -me MeRegion.bin -pmcp pmcp.bin* ***-o newName.bin*** *– bios firmware\BIOS.bin*

Table 67. Override number of flash components, override flash component 1 size and save configuration

Use Case	Override number of flash components, override flash component 1 size and save configuration
CLI parameters	FITm_cmd.exe --params descriptor:NumberOfSpiComponents=1 descriptor:FlashComponent1Size=0x4000000 -s configuration2.xml
Example	FITm_cmd.exe *--params descriptor:NumberOfSpiComponents=1 descriptor:FlashComponent1Size=0x4000000 -s configuration2.xml*
SPS FITc comparison	*spsFITc.exe -flashcount 1 -flashsize1 7 -save configuration2.xml*

Table 18. Override default platform and build image

Use Case	Override default platform and build image
CLI parameters	FITm_cmd.exe -b --params layout:default_platform=<selected_platform>
Example	FITm_cmd.exe *-b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin layout:default_platform=”Archer City SPR” --layout bios=fw\bios.rom*
SPS FITc comparison	*spsFITc.exe –b -me MeRegion.bin -pmcp pmcp.bin -platform "Archer City" –bios firmware\BIOS.bin*

Supported platforms: 'Archer City SPR', 'Archer City CLX', 'Archer City xPV'

Table 79. Enable/disable specified region and build image

Use Case	**Build image with gbe region enabled**
CLI parameters	FITm_cmd.exe –-params regions_enable_switches:<region_name>=<0x0 or 0x1>
Example	FITm_cmd.exe *-b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***regions_enable_switches:gbe=1*** *gbe:input_file=gbe.bin*
SPS FITc comparison	*Not applicable – enabling/disabling containers can be done by GUI/config xml*

Table 20. Build image with non-default region order

Use Case	**Build image with me placed before gbe**
CLI parameters	FITm_cmd.exe -b --params layout:region_order=<desired_order>
Example	FITm_cmd.exe *-b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***layout:region_order=23456789ABCDEF1*** *--layout bios=fw\bios.rom*
SPS FITc comparison	*Not applicable – modifying region order can be done by GUI/config xml–bios firmware\BIOS.bin*

Additional information:

descriptor is always the first region in the binary. Order of the rest of the regions can be set by string value, where single letter stands for single region, specified as follows:

{'1': 'bios', '2': 'me', '3': 'gbe', '4': 'pdr', '5': 'der1', '6': 'sec_bios', '7': 'ucode', '8':'embed_ctrl', '9': 'der2', 'A': 'ie', 'B': 'gbeA', 'C': 'gbeB', 'D':'spare1', 'E':'spare2','F':'PTT'}

Table 21. Change region size and build image

Use Case	**Build image with 0x1000 pdr binary extended to 0x9000**
CLI parameters	FITm_cmd.exe –b –-params layout:<region_name>_size=<size>
Example	FITm_cmd.exe *-b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***layout:pdr_size=0x9000*** *--layout pdr=one_kilo.bin*

SPS FITc comparison	*Not applicable – modifying region length can be done by GUI/config xml*

2.5 General Modular FIT comparison to FITc

Modular FIT introduces concept of containers. Single container refers to specified region of binary image. In case the region is built by Modular FIT, the container describes its configuration and layout. Otherwise, the container is a binary file. This concept results in easy configuration of regions. There is only one CLI switch to override region with a binary file instead of separate switches for single regions – the *--layout* switch. There is also only one switch to override parameters values of any region, the *--params* switch. Furthermore, some settings that are configured by a general switch in FITc are settings of layout container in Modular FIT.

There are only a few settings that do not belong to any container and those settings are listed among other CLI switches at the beginning of this chapter. The rest of the settings belong to containers and they can be:

-viewed by –help <container_name> switch

-configured via –params <container_name>:<setting_name> switch

Environmental variables:

In opposite to FITc, Modular FIT does not use Environmental Variables to set paths. Every relative path refers to tool directory (directory where Modular FIT executable exist). Full paths can be used.

New feature - verbose logging:

"—verbose" switch makes the logging during image build more detailed. Additionally, Modular FIT prints image tree that shows dependencies among containers:

```
Resolving dependencies
Image containers dependency tree:
      ┌descriptor v. 0.1 in plugin: ebg_descriptor v. 1.0.85
      │                                                    └pch_straps v. 0.1 in plugin: ebg_pch_straps v. 1.0.85
image─┤
      │                                          ┌mfs v. 0.1 in plugin: ebg_mfs v. 1.0.85
      ├me v. 0.1 in plugin: me_ebg v. 1.0.85─┤
      │                                          └uep v. 0.1 in plugin: uep v. 1.0.85
      └bios
```

Figure 2 Image containers dependency tree

2.6 Modular FIT configuration file

Default user configuration file can be created by calling Modular FIT with '-s' parameter:

FITm_cmd.exe -s configuration.xml

Configuration will be saved to configuration.xml file as stated by tool's output:

```
Configuration file saved successfully: C:\Modular Flash Image Tool\configuration.xml
```

Figure 3. Configuration saved successfully

Saved configuration file contains all settings with configurable values from all used regions and layout. Configuration can then be loaded by using '-i' option, e.g.: the command below will build image from previously saved configuration:

FITm_cmd.exe -b -i configuration.xml

Please note, that the default configuration will not include paths to firmware collaterals as ME binary or pmc. These paths can be:

a)provided with –params argument when saving the configuration, like in the example below:

FITm_cmd.exe --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin --layout bios=fw\bios.rom -s configuration.xml

b)edited in the saved configuration later

```
<?xml version="1.0" ?>
<user_settings layout="EagleStream" layout_version="1.0.0" core_version="1.0.0">
    <container name="layout">
    <container name="descriptor" version="0.1" plugin_name="ebg_descriptor" plugin_version="0.1"
    config_version="0.5">
    <container name="pch_straps" version="0.1" plugin_name="ebg_pch_straps" plugin_version="0.1"
    config_version="0.5">
    <container name="me" version="0.1" plugin_name="ebg_me" plugin_version="0.1" config_version=
    <container name="mfs" version="5.113" plugin_name="ebg_mfs" plugin_version="0.1" config_vers
        <setting name="Features_Configuration-NmEnabled" value="0x1"/>
        <setting name="Features_Configuration-MctpInfrastructureEnabled" value="0x0"/>
        <setting name="Features_Configuration-CupsEnabled" value="0x1"/>
        <setting name="Features_Configuration-ThermalReportingEnabled" value="0x1"/>
```

Figure 4. User configuration file

Configuration contains settings of containers with configurable parameters and binary containers with no settings (except binary path)

```
<container config_version="0.5" name="gbe" plugin_name="gbe" plugin_version="1.0.108" version="0.1">
    <setting name="input_file" value="plugins\gbe\data\gbe.bin"/>
    <setting name="mac_address" value="FFFFFFFFFFFF"/>
</container>
```

Figure 5. Example of container with configurable settings

```
<container binary="" name="sec_bios"/>
```

Figure 6. Example of binary container

Settings for each configurable container are grouped within the *container* node in configuration xml. This node contains information about the version of plugin and configuration file which was used to generate this file. These attributes appear for the containers with configurable settings.

This class of containers is by default enabled which means that they take a part in building process and they are embedded in the produced binary image. To disable a configurable container one can define *enabled="false"* attribute on the *container* node level as presented below:

```
<container name="gbe" version="1.0" plugin_name="gbe" plugin_version="0.1"
config_version="0.5" enabled="False">
    <setting name="input_file" value=""/>
    <setting name="mac_address" value="FFFFFFFFFFFF"/>
</container>
```

Figure 7. Disabling container

There is also a number of containers which don't have a corresponding xml configuration and don't require any settings to be defined. These are so called binary containers. They can be used by simply defining a proper file path in the *binary* attribute (e.g. sec_*bios* container). If the path is not specified (empty), the container is not placed in binary.

***Binary* attribute can also be used to disable building a configurable container. Then the specified binary is used instead of building the container.**

```
<container config_version="0.5" binary="binary.bin" enabled="True"
name="gbe" plugin_name="gbe" plugin_version="1.0.108" version="0.1">
```

```
Building main phase containers
Dependency stage 0
Building pch_straps container...
Using binary at 'binary.bin' for gbe container...
Dependency stage 1
```

Figure 8. Using a binary instead of building a container

Note: To see detailed log that reports such message, Modular FIT build has to be run with
--verbose argument.

2.7 Signing binary image

To sign data in generated binary Modular FIT uses cryptographic keys. One way to generate such key is as follows:

openssl genpkey -algorithm RSA -out key.pem -pkeyopt rsa_keygen_bits:3072

The generated OEM key format is defined in RFC 1421 through 1424, which is a container format that may include an entire certificate chain including public key, private key, and root certificates.

2.7.1 Keys' handling

Note that Modular FIT doesn't store nor sends private keys provided by the user. Keys are only passed to the library which is cryptography.

Public keys - modulus and public exponent parts - may be placed inside the generated binary if provided configuration file specifies so. This is under full user's control and mostly desired.

2.7.2 Key lifetime

According to Intel crypto guidelines, Modular FIT supports crypto algorithms, SHAs and padding schemes that are approved for current use or allowed for legacy use for the moment of Modular FIT release.

Items that are utilized by Eaglestream are bolded in following tables:

Table 8. Crypto algorithms supported by Modular FIT

Supported algorithm	Notes
RSA-2048	**Legacy mode only**
RSA-3072	
RSA-4096	
ECC-256 **secp256r1 curve**	**Legacy mode only**
ECC-384 secp384r1 curve	
ECC-384	

brainpoolP384r1 curve	
AES-128	**Legacy mode only**
AES-256	

Table 9. SHAs supported by Modular FIT

Supported SHA	Notes
SHA-256	**Legacy mode only**
SHA-384	
SHA-512	

Table 10. Padding supported by Modular FIT

Supported padding	Notes
PSS	
v1_5	**Legacy mode only**

2.7.3 Signing Flash Descriptor

This chapter introduces following names:

fd0v manifest – Flash descriptor verification manifest

fd0v manifest extension – extension of the fd0v manifest, a necessary part of fd0v manifest

Signing Flash descriptor can be done by Modular FIT or it can be done by an external tool ("offline signing"). Signing descriptor means creating (and including in image) an fd0v manifest that includes fd0v manifest extension.

Signing by Modular FIT occurs when private key for descriptor signing is provided. Signing outside Modular FIT occurs when offline signing parameter is enabled - there is no need to provide signing key then, since in such case Modular FIT does not sign the binary itself.

When signing key is provided or offline signing is enabled**, Modular FIT produces fd0v manifest extension** that depends on following data:

- hash which is calculated from flash descriptor regions excluding ranges defined in FITm configuration
- all exclude ranges data (offsets and lengths)

```
<setting name="FdManifest" value=""/>
<setting name="exclude_enabled_0" value="0x1"/>
<setting name="exclude_base_0" value="0x800"/>
<setting name="exclude_limit_0" value="0xc00"/>
<setting name="exclude_enabled_1" value="0x0"/>
<setting name="exclude_base_1" value="0x0"/>
<setting name="exclude_limit_1" value="0x0"/>
```

Figure 9. Hash exclude ranges configuration in user xml

exclude_base_<number> - start offset of excluded range

exclude_limit_<number> - end offset of excluded range

exclude_enabled_<number> - exclude range turned on (0x1) or off (0x0)

Up to 8 exclude ranges (0-7) can be defined.

By default, fd0v manifest extension is saved in *plugins\ebg_descriptor\fd_hash.bin*. Path, where the fd0v manifest extension is saved, can be set by "–params descriptor:FdHashPath=<path>" argument

Signing descriptor by Modular FIT

Table 22. Signing descriptor by Modular FIT – in one step

Use Case	***Signing descriptor***
CLI parameters	FITm_cmd.exe -b --params descriptor: FdSigningKey=<private_key>
Example	*FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***descriptor:FdSigningKey=3k_key_private.pem*** *--layout bios=fw\bios.rom*

Signing descriptor outside Modular FIT – in two steps

Table 113. First step

Use Case	**Offline signing**

CLI parameters	FITm_cmd.exe -b --params descriptor:OfflineSigning=0x1
Example	*FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* ***descriptor: OfflineSigning=0x1*** *--layout bios=fw\bios.rom*

First step is to create fd0v manifest extension. This binary file is an input for generating whole fd0v manifest, which has to be generated outside Modular FIT before the second step. Fd manifest can be built by IBST with FD0V_Manifest.xml. Please refer to IBST user guide for more information.

Table 124. Second step

Use Case	**Offline signing**
CLI parameters	FITm_cmd.exe -b --params descriptor:FdManifest=<signature_path> OfflineSigning=0x1
Example	*FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin* **descriptor: FdManifest=fd_manifest.bin *OfflineSigning*=***0x1** *--layout bios=fw\bios.rom*

Building an image with provided manifest (offline signing) includes manifest verification (fd0v manifest binary includes required key data). If the manifest and key does not match descriptor region hash then an error is displayed.

3 Build SPI Image

To build SPI image with Modular Flash Image Tool:

1. Provide all necessary files as listed in Table

2. Provide paths to required files within –params argument. Remember that paths must be relative to directory with fit_cmd.py, also full paths can be used

3. Run Modular FIT as shown in Basic usage of Modular FIT in Command Line Interface chapter, e.g.:
   FITm_cmd.exe -b --params me:input_file=fw\MeRegion.bin pmc:PMCBinary=fw\pmcp.bin ***--layout*** *bios=fw\bios.rom*

4. Image will be generated. The tool will print path to the generated binary. The default name is outimage.bin

```
Building image...
Image successfully built
Map created at: C:\Repo\SPS_Tools_FIT\outimage.map
Binary created at: C:\Repo\SPS_Tools_FIT\outimage.bin
```

Most of the settings available to be configured via an xml can be overridden via CLI which in cases of scripts in integration process can be easier to use.

Sample build sequence via CLI switches:
FITm_cmd.exe -b -i config.xml --layout bios=fw\bios.rom pdr=pdr.bin gbeA=gbea.bin gbeB=gbeb.bin --params me:input_file=fw\MeRegion.bin descriptor:FlashComponent1Size=0x4000000 NumberOfSpiComponents=0 pmc:PMCBinary=fw\pmcp.bin layout:output_file=image.bin