Intel® FSP 2.x Measurement and Attestation

Preliminary Specification

July 2021

Document Number: 644001-1.0

Contents

1.0 Introduction 6
- 1.1 Background 6
- 1.2 Overview 6
- 1.3 Terminology 7

2.0 FSP Introduction 9
- 2.1 FSP Architecture 9
- 2.2 Challenges and Solutions 10
  - 2.2.1 FSP Component indicator 10
  - 2.2.2 FSP Image Base 11
  - 2.2.3 Updateable Product Data 11

3.0 FSP Reference Manifest 13
- 3.1 FSP Manifest format 13
- 3.2 FSP Hash Mode 16
- 3.3 FSP Binary Update 16
  - 3.3.1 FSP Image Base 17
  - 3.3.2 FSP Configuration 17
- 3.4 FSP Manifest Signing 17
- 3.5 FSP Public Root Certificate 17

4.0 FSP Runtime Measurement 18
- 4.1 FSP Measurement Mode 18
  - 4.1.1 TCG Event Log for FSP Component 18
  - 4.1.2 TCG Event Log for Sp800_155_PlatformId_Event2 19

5.0 FSP Attestation 20
- 5.1 FSP Identification 20
- 5.2 FSP Verification 21
- 5.3 FSP Attestation Flow 21
  - 5.3.1 Platform Attestation 21
  - 5.3.2 Platform Identification and Verification 21
  - 5.3.3 FSP Identification and Verification 22

6.0 Conclusion 23

Appendix A: References 24
- NIST 24
- TCG 24
- IETF 25
- Intel 25
- EDKII 25
- Book 26

Web 26

Appendix B: Example 27

A. RIM Example (ini format) 27
B. Example of SWID tag (XML format) 27
One-Binary Mode Manifest Info 27
Separation Mode Manifest Info 27
C. Example of CoSWID tag (CBOR format, described in JSON format) 27
One-Binary Mode Manifest Info 27
Separation Mode Manifest Info 27

Tables

Table 1. Terminology 7
Table 2. FSP Manifest 13
Table 3. FSP Measurement (Separation Mode) 18
Table 4. TCG Sp800_155_PlatformId_Event2 19

Revision History

Date	Revision	Description
July 2021	1.0	• Initial release.

§

1.0 Introduction

1.1 Background

According to *NIST SP800-155 – firmware integrity measurement*, each firmware component needs to ensure that industry best practices are followed, such as firmware integrity reporting. To meet this requirement, the Trusted Computing Group (TCG) defined how to provide the reference integrity manifest (RIM) at manufacturing time, and also how to collect the firmware integrity measurement (FIM) at system boot time.

The Intel Firmware Support Package (FSP) is an Intel delivered binary that is integrated into Original Equipment Manufacturer (OEM) system firmware. An OEM may get the original source code, modify the FSP, and subsequently create an OEM specific FSP binary. In some use cases, a third party needs to verify the provenance of the FSP binary. This verification includes supply chain concerns about the authenticity of an FSP during update and manufacturing times. As one mechanism for verification, a third-party attestation service can introspect on the component.

1.2 Overview

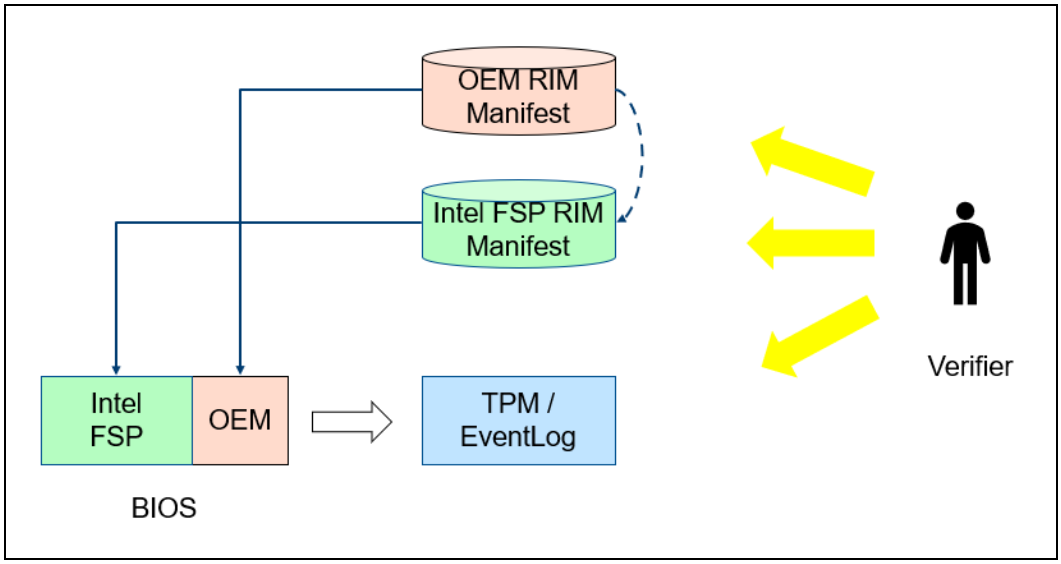
For the Intel FSP binary, we assume Intel is the entity to provide an Intel FSP reference integrity manifest (RIM) for the FSP binary that is released. An OEM may use the unmodified Intel FSP binary. In this case, the OEM can provide an OEM reference integrity manifest (RIM) as a base RIM for the platform and treat the Intel FSP RIM as a support RIM. The base RIM and Intel FSP RIM make up a RIM bundle for the platform.

An OEM may treat the FSP as their own proprietary object and modify it freely. In this case, the OEM will be the entity to provide an OEM FSP reference integrity manifest (RIM) for the OEM FSP binary, and the OEM FSP RIM will be part of the RIM bundle.

The FSP firmware integrity measurement (FIM) should be collected during the system firmware boot by the FSP consumer. Then the remote verifier can use the remote attestation to verify if the FSP FIM matches the FSP RIM.

Figure 1 shows the high level picture on how the flow works.

Figure 1. Intel FSP Remote Attestation



1.3 Terminology

Table 1. Terminology

Term	Description
ACM	(Intel) Authenticated Code Module
ACPI	Advanced Configuration and Power Interface
BIOS	Basic Input/Output System
CBOR	Concise Binary Object Representation, see RFC 7049
COSE	CBOR Object Signing and Encryption, see RFC 8152
CoSWID	(IETF) Concise Software Identification Tags, see https://datatracker.ietf.org/doc/draft-ietf-sacm-coswid/
CTM	Chain-of-Trust for Measurement
DICE	(TCG) Device Identifier Composition Engine, see https://trustedcomputinggroup.org/work-groups/dice-architectures/
FIM	(TCG) Firmware Integrity Measurement, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PC_Client-FIM_v1r24_3feb20.pdf
FSP	Intel Firmware Supported Package
JSON	JavaScript Object Notation, see https://www.json.org/json-en.html
JOSE	JSON Object Signing and Encryption, see RFC 7515
PKCS	Public-Key Cryptography Standards
RATS	(IETF) Remote Attestation Procedures, see https://datatracker.ietf.org/wg/rats/documents/

Term	Description
RIM	(TCG) Reference Integrity Measurement, see https://trustedcomputinggroup.org/wp-content/uploads/TCG_RIM_Model_v1-r13_2feb20.pdf, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PC_Client_RIM_r0p15_15june2020.pdf
RTM	Root-of-Trust for Measurement
SACM	(IETF) Security Automation and Continuous Monitoring, see https://datatracker.ietf.org/wg/sacm/documents/
SMBIOS	System Management BIOS
SWID	(ISO) Software Identification, see https://csrc.nist.gov/projects/Software-Identification-SWID
TPM	Trusted Platform Module
XML	eXtensible Markup Language, see https://www.w3.org/TR/xml/, https://www.w3.org/TR/xmldsig-core1/

§

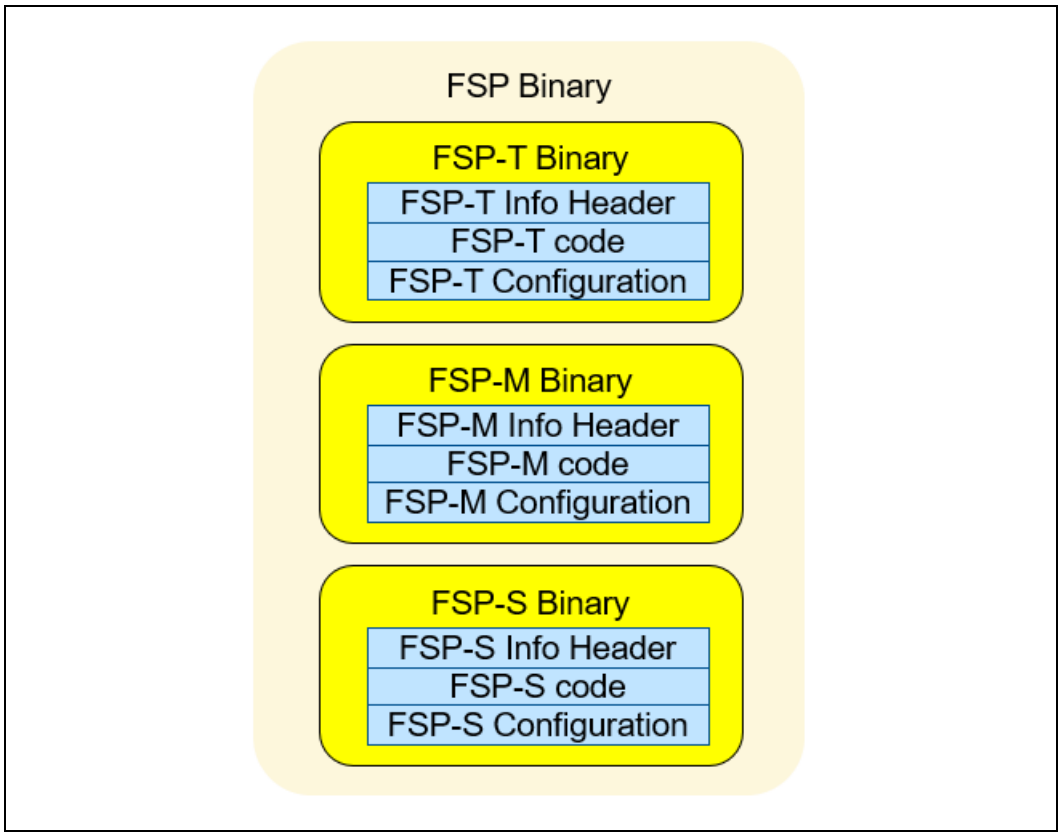
2.0 FSP Introduction

This chapter introduces the FSP architecture, challenges, and solutions for FSP measurement in the system firmware.

2.1 FSP Architecture

According to the Intel FSP External Architecture Specification 2.x, the FSP binary includes three sub-components: FSP-T (temporary-ram initialization), FSP-M (memory-initialization) and FSP-S (silicon initialization). See Figure 2 for more details on the three sub-components. Each subcomponent includes an FSP information header, FSP code, and FSP configuration region. The FSP code exposes a set of FSP APIs for initialization. The FSP configuration region is known as the Updateable Product Data (UPD) and contains the default parameters for FSP initialization. The FSP information header includes the FSP binary version, attribute, image base, the FSP API offset, and FSP configuration offset, etc.

Figure 2. Intel FSP 2.x Binary



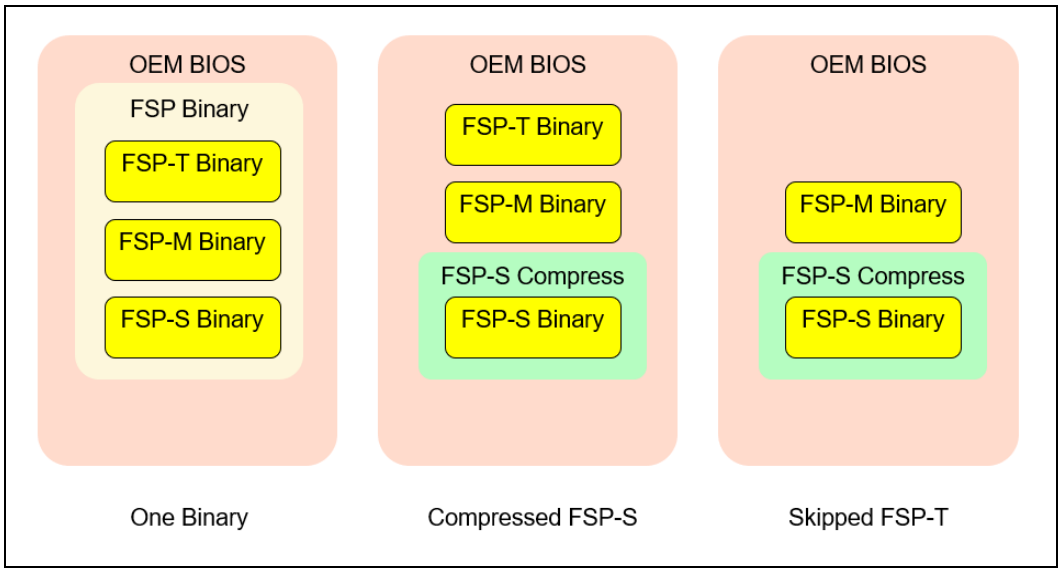
The system firmware integrator can decide how to put the FSP binary into the system firmware. Figure 3 shows the three possible options.

Option 1 – One binary. The system firmware integrator can put the whole FSP binary into the OEM BIOS.

Option 2 – Compressed FSP-S. The system firmware integrator can split the FSP-S from the rest of the FSP sub-components and compress it. Then the compressed FSP-S can be put into any location.

Option 3 – Skipped FSP-T. The system firmware integrator may also skip a subcomponent of FSP if the same functionality is already implemented. For example, the FSP-T is omitted by some implementations, such as Intel-based Chromebooks built using coreboot and FSP.

Figure 3. Intel FSP 2.x Integration Option



2.2 Challenges and Solutions

2.2.1 FSP Component indicator

In the current implementation, the OEM firmware measures the whole FSP binary and its sub-components together, because there is no easy way to know which part is FSP-T, FSP-M, or FSP-S.

The solution entails extending the FSP-T, FSP-M, FSP-S component explicitly by the system firmware root-of-trust of measurement or chain-of-trust for measurement. The system firmware shall use TCG defined **UEFI_PLATFORM_FIRMWARE_BLOB2** data

structure with the FSP specific ASCII descriptor. As such, the verifier can distinguish the FSP component from other OEM firmware volumes.

2.2.2 FSP Image Base

When an FSP image is generated, the base address of an PE/COFF image is fixed. If an OEM chooses to use the FSP binary as is, then the FSP image shall be put to the flash address indicated by the FSP image base address.

If the OEM wants to put the FSP image into a different location, then the FSP image should be rebased. This build time rebase operation may impact the FSP-T and FSP-M because they are executed on flash and runtime rebasing may not possible. Usually, the FSP-S is not impacted because it is loaded into arbitrary memory after the memory initialization and rebased before execution.

In general, it is recommended to use the FSP binary as it. As such, the FSP reference integrity manifest can be used directly.

If an OEM has to rebase the FSP binary, then the OEM may calculate the hash of the rebased FSP binary and provide a supplemental RIM. The verifier can use this information to determine that the OEM FSP is derived from an Intel FSP and the base address has been updated.

In the future, the FSP itself can implement some improvements such as:

- FSP-T execution-in-place (XIP). The FSP-T itself can be updated to support execution in place. As such, the OEM may choose to put the FSP to any flash location.
- FSP-M execution in fixed cache address. The FSP-M can be built with a fixed address. If the fixed address can be in the code cache, then the FSP-M consumer can always load the FSP-M to that fixed address at runtime and no rebase operation is required.
- FSP-M execution with page tables. If the fixed address can be in a flash region, then the FSP-M can use a CPU page table to map the FSP-M to this fixed address.
- FSP-M self-rebase in cache. If the CPU code cache is writable, then the FSP-M can perform a self-rebase in the cache before execution.

2.2.3 Updateable Product Data

Each FSP component includes an updateable product data (UPD) configuration region. In most cases, the OEM just uses this region as is. If there is a need to provide a different configuration, then the configuration region is prepared at runtime. As such, the FSP reference integrity measurement can be used as is.

In some rare cases, if the OEM really wants to update the UPD region in the FSP binary, the OEM can choose to separate the FSP code measurement from the configuration

data measurement. The OEM can calculate the hash of the UPD region and provide a supplemental RIM. Then the verifier can know that the FSP code is unmodified and that the UPD data has been updated by the OEM.

According to the TCG Platform firmware profile specification, the FSP code shall be measured into PCR0 and the configuration shall be measured into PCR1. As such, the isolation of the code measurement and data measurement is a good way to let verifier know the information.

§

3.0 FSP Reference Manifest

This chapter defines the FSP reference integrity manifest.

3.1 FSP Manifest format

An FSP manifest is the data structure to describe an FSP binary. The manifest could be a software Identification (SWID) Tag in eXtensible Markup Language (XML) format defined in “ISO/IEC 19770-2:2015 Part 2: Software Identification Tag” or Concise SWID (CoSWID) Tag in Concise Binary Object Representation (CBOR) format defined in “IETF SACM: Concise Software Identification Tags”.

Table 3 lists the elements and attributes defined in SWID/CoSWID for FSP.

Table 2. FSP Manifest

Element	Attribute	Required	Description
Software Identity	Name	Required	Name of the FSP binary (match **FSP_INFORMATION _HEADER ImageId**)
	Version	Required	Version of the FSP binary (match **FSP_INFORMATION _HEADER ImageRevision Major+Minor+Revision**)
	TagId	Required	GUID to identify the FSP (match **TCG_Sp800-155-PlatformId_Event2 ReferenceManifestGuid**)
	TagVersion	Required	Version of the tag, “0”
	Corpus	Optional	“FALSE”
	Patch	Optional	“FALSE”: Initial RIM “TRUE”: Subsequential RIM

Element	Attribute	Required	Description
	Supplemental	Optional	"FALSE": first RIM (e.g. primary RIM) "TRUE": not firmst RIM (e.g. issued by System Integrator or Value Add Reseller)
Entity	Name	Required	"Intel"
	Regid	Optional	"http://www.intel.com"
	Role	Required	"softwareCreator tagCreator"
	Thumbprint	Required	digest of the signing certificate
Link	Href rel= "installation media"	Optional	Download URL https://github.com/intel/FSP/<FspBinPkg>
	Href rel= "supersedes", "patches", "requires"	Optional	Link to previous RIM
Meta	colloquialVersion	Required	Marketing version of the FSP
	Edition	Required	"FSP" + FSP Specification Version (need match **FSP_INFORMATION _HEADER SpecRevision**)
	Production	Required	Name of the FSP
	Revision	Required	Revision of the FSP
	PayloadType	Optional	"Direct"
	PlatformManufacturerStr	Required	"Intel" (match **TCG_Sp800-155-PlatformId_Event2 PlatformManufacturerStr**)

Element	Attribute	Required	Description
	PlatformManufacturerId	Required	"343" (IANA identifier) (match **TCG_Sp800-155-PlatformId_Event2 PlatformManufacturerId**)
	PlatformModel	Required	Name of FSP platform (match **TCG_Sp800-155-PlatformId_Event2 PlatformModel**)
	PlatformVersion	Optional	Version of the FSP platform (match **TCG_Sp800-155-PlatformId_Event2 PlatformVersion**)
	FirmwareManufacturerStr	Optional	"Intel" (match **TCG_Sp800-155-PlatformId_Event2 FirmwareManufacturer**)
	FirmwareManufacturerId	Optional	"343" (match **TCG_Sp800-155-PlatformId_Event2 FirmwareManufacturerId**)
	FirmwareModel	Optional	Name of FSP platform (match **TCG_Sp800-155-PlatformId_Event2 FirmwareModel**)
	FirmwareVersion	Optional	Version of the FSP platform (match **TCG_Sp800-155-PlatformId_Event2 FirmwareVersion**)
	BindingSpec	Required	"RIMIM"
	BindingSpecVersion	Required	"0.1"
	pcURILocal	Optional	URI for this RIM in local device

Element	Attribute	Required	Description
	pcURIGlobal	Optional	URI for this RIM on web
	RIMLinkHash	Optional	Not needed for Base RIM
Payload	File, Directory	Required	Location and Name of the FSP component
	Name	Required	Name of the FSP component
	Size	Required	Size of the FSP component (need match **FSP_INFORMATION _HEADER ImageSize**)
	Hash	Required	Hash of the FSP component, such as FSP-T, FSP-M, FSP-S.
	supportRIMType	Optional	N/A
	supportRIMFormat	Optional	N/A
	supportRIMURIGlobal	Optional	N/A

3.2 FSP Hash Mode

FSP has two hash modes:

- Separation Mode - The FSP component code and FSP configuration are hashed separately.
- One-binary Mode - The FSP component code and FSP configuration are hashed together.

3.3 FSP Binary Update

After the FSP binary is released by an OEM, the OEM may update the FSP in either of the below two cases. If the OEM updates the FSP binary, the FSP hash will be updated. The OEM may release a Supplemental RIM based upon the primary RIM released by Intel.

3.3.1 FSP Image Base

The OEM may choose to put the FSP into another firmware location. The location is different from the FSP location with which the FSP binary is generated. An FSP binary includes an **FSP_PATCH_TABLE** to provide the binary patch information of an FSP binary. The **FSP_PATCH_TABLE** is similar to the relocation section of a PE image. The FSP base address can be identified by the **ImageBase** in the **FSP_INFO_HEADER**.

3.3.2 FSP Configuration

An updateable product data (UPD) region can be identified by the **CfgRegionOffset** and **CfgRegionSize** in the **FSP_INFO_HEADER**. It contains the default setting for the FSP initialization. The UPD can be designed to be updated by the OEM via a Binary Configuration Tool (BCT) with a Boot Setting File (BSF).

3.4 FSP Manifest Signing

The FSP manifest may use SWID or CoSWID.

The SWID FSP manifest must be signed using an XML format.

The CoSWID FSP manifest must be signed with a CBOR Object Signing and Encryption (COSE) format.

3.5 FSP Public Root Certificate

The FSP public root certificate is published to the Intel GitHub repository when the FSP is released.

Revocation

The revoked public root certificate shall also be published to the same web site.

§

4.0 FSP Runtime Measurement

This chapter defines the FSP runtime measurement collection.

4.1 FSP Measurement Mode

The platform needs to measure the FSP binary component during boot. Because there are two FSP hash modes, there are also two FSP measurement modes:

1. Separation Mode - The FSP code and FSP configuration are measured separately based upon table 4-1.
2. One-binary Mode - The FSP binary is measured as one event based upon table 4-2.

4.1.1 TCG Event Log for FSP Component

If the FSP UPD region is NOT designed to be updated, then the one-binary mode can be used. If the OEM may update the FSP UPD region in the final BIOS image, then the separation mode should be used.

The FSP measurement should be done by the root-of-trust for measurement (RTM) or chain-of-trust for measurement (CTM) based upon the execution time flow. The RTM or CTM can be an Intel Authenticated Code Module (ACM) with Intel® Boot Guard enabled or an OEM PEI module (PEIM).

Table 3. FSP Measurement (Separation Mode)

Component	PCR	Event Type	Event Data	Event Log
FSP Code	0	EV_EFI_PLATFORM_FIRMWARE_BLOB2	FSP binary before UPD || FSP binary after UPD ("||" means concatenate)	EFI_PLATFORM_FIRMWARE_BLOB2 structure (Descriptor: "FSPTAPI", "FSPMAPI", or "FSPSAPI")
FSP Configuration	1	EV_PLATFORM_CONFIG_FLAGS	UPD region	EFI_PLATFORM_FIRMWARE_BLOB2 structure (Descriptor: "FSPTUPD", "FSPMUPD", or "FSPSUPD")

Component	PCR	Event Type	Event Data	Event Log
FSP binary (Code + Configuration)	0	EV_EFI_PLATFORM_FIRMWARE_BLOB2	FSP binary	EFI_PLATFORM_FIRMWARE_BLOB2 structure (Descriptor: “FSPT”, “FSPM”, or “FSPS”)

NOTES:

1. The FSP hash in these event log entries shall be based upon the FSP component binary. It might NOT be the hash of the executed code, which is acceptable as long as the measurement is performed by the RTM or CTM. We have cases to patch the PE COFF image, ACPI table, and SMBIOS table, etc. in designs today.
2. The UPD hash in these event log entries shall be based upon the FSP default configuration in the FSP component binary UPD region. It might NOT be the final policy for the FSP. The final runtime policy might be updated at runtime by the platform FSP wrapper code. In this case, the runtime policy is included in the FSP wrapper code and measured into PCR0.

4.1.2 TCG Event Log for Sp800_155_PlatformId_Event2

If a platform wants to support FSP component measurement, a TCG_Sp800_155_PlatformId_Event2 event shall be created in the event log to provide the FSP information. See Table 4.

Table 4. TCG Sp800_155_PlatformId_Event2

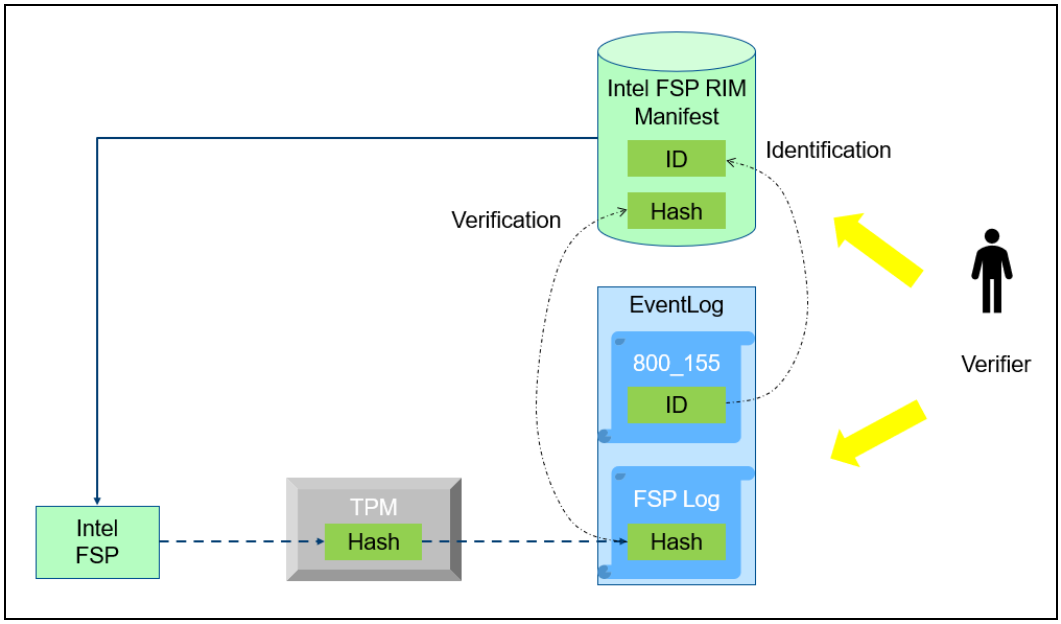
Element	Field	Required	Description
TCG SP800-155 PlatformID event2	Signature	Required	“SP800-155 Event”
	PlatformManufacturerId	Required	“343” (IANA identifier)
	ReferenceManifestGuid	Required	SoftwareIdentity.TagId
	PlatformManufacturerStr	Required	“Intel”
	PlatformManufacturerId	Required	“343” (IANA identifier)
	PlatformModel	Required	Name of FSP platform
	PlatformVersion	Optional	Version of the FSP platform
	FirmwareManufacturerStr	Optional	“Intel”
	FirmwareManufacturerId	Optional	“343”
	FirmwareModel	Optional	Name of FSP platform
	FirmwareVersion	Optional	Version of the FSP platform

§

5.0 FSP Attestation

This chapter describes the FSP attestation process. Figure 4 shows the high level flow.

Figure 4. FSP Component Attestation



5.1 FSP Identification

The FSP identification process is described below:

1. The verifier retrieves the TCG event log and iterates until discovery of a TCG_Sp800_155_PlatformId_Event2 log entry, which includes VendorId, ReferenceManifestGuid, PlatformManufacturerStr, etc.
2. The verifier gets the corresponding FSP reference integrity manifest (RIM) where the RIM.SoftwareIdentity.TagId matches the TCG_Sp800_155_PlatformId_Event2.ReferenceManifestGuid, the RIM.Meta.PlatformManufacturerStr matches the TCG_Sp800_155_PlatformId_Event2.PlatformManufacturerStr, the RIM.Meta.PlatformManufacturerId matches the TCG_Sp800_155_PlatformId_Event2.PlatformManufacturerId, and the RIM.Meta.PlatformModel matches the TCG_Sp800_155_PlatformId_Event2.PlatformModel.

At this point, the verifier knows the target platform FSP information, as recorded in the FSP RIM.

5.2 FSP Verification

Once the verifier identifies the FSP binary RIM, the verifier can then compare the FSP hash in the TCG event log with the hash in the FSP RIM.

The verification process is described below:

1. The verifier gets the event log, parses all EV_EFI_PLATFORM_FIRMWARE_BLOB2 event log entries, and checks the descriptor string.
2. In one binary mode, the descriptor string for FSP component is: “FSPT”, “FSPM” and “FSPS”. In separation mode, the descriptor string for FSP code is: “FSPTAPI”, “FSPMAPI” and “FSPSAPI”, the descriptor string for FSP UPD is: “FSPTUPD”, “FSPMUPD” and “FSPSUPD”.
3. The verifier compares the hash value in the TCG event log with the hash value in the FSP RIM.
4. If the hash values are same, that means the platform’s FSP is unmodified. If the hash values are different, then the verifier can perform the corresponding remediation.

5.3 FSP Attestation Flow

The whole platform attestation includes more steps.

5.3.1 Platform Attestation

The verifier must identify the TPM with TPM endorsement key, verify the TPM PCR by using TPM Quote command with the TPM attestation key, then verify the TCG event log by using the event log for replay and comparison with TPM PCR. [TCG PC-FIM] provides some basic information on this process. Some articles also describe the attestation flow, such as [Attestation1], [Attestation2], and [Attestation3].

5.3.2 Platform Identification and Verification

The verifier retrieves the platform TCG_Sp800_155_PlatformId_Event2 structure from the TCG Event Log. The VendorId, ReferenceManifestGuid, PlatformManufacturerStr, and other fields can be used to identify the platform reference integrity manifest.

If there are any event entries before the FSP event log, then the verifier shall verify those event log entries with the platform RIM.

This step is to guarantee the FSP chain-of-trust for measurement is unmodified.

5.3.3 FSP Identification and Verification

The last step is to identify the FSP and verify the FSP hash according to 5.1 and 5.2. If the FSP hash matches, then the verifier knows the integrated FSP is authentic. Otherwise, the FSP is modified. The verifier may choose to perform remediation or disable some features for cases of a modified FSP.

§

6.0 Conclusion

As noted, supporting supply chain integrity has taken on more prominence in the industry. This paper describes how Intel host firmware based upon the Intel Firmware Support Package can be used with infrastructure defined by the Trusted Computing Group to track the provenance of solutions based upon the Intel FSP. This guidance includes models of combined and split FSP binary integration into OEM firmware.

§

Appendix A: References

NIST

[NIST SP800-155] BIOS Integrity Measurement Guidelines, https://csrc.nist.gov/publications/sp800

[NIST SWID] Software Identification (SWID) tagging, https://csrc.nist.gov/projects/Software-Identification-SWID, https://pages.nist.gov/swid-tools/

TCG

[TCG TPM] Trusted Platform Module, https://trustedcomputinggroup.org/work-groups/trusted-platform-module/

[TCG RIM] TCG Reference Integrity Manifest (RIM) Information Model, https://trustedcomputinggroup.org/resource/tcg-reference-integrity-manifest-rim-information-model/

[TCG PC-RIM] TCG PC Client Reference Integrity Measurement, https://trustedcomputinggroup.org/resource/tcg-pc-client-reference-integrity-manifest-specification/

[TCG PC-FIM] TCG PC Client Platform Firmware Integrity Measurement, https://trustedcomputinggroup.org/wp-content/uploads/TCG_PC_Client_FIM_v1_r40_02dec2020.pdf

[TCG PFP] TCG Platform Firmware Profile, https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/

[TCG Platform-CERT] TCG Platform Certificate Profile, https://trustedcomputinggroup.org/resource/tcg-platform-certificate-profile/

[TCG TPA IM] TCG, Trusted Attestation Protocol (TAP) Information Model for TPM 1.2 and 2.0 and DICE, https://trustedcomputinggroup.org/resource/tcg-tap-information-model/

[TCG TPA Use Case] TCG, Trusted Attestation Protocol (TAP) Use Cases for TPM 1.2 and 2.0 and DICE, https://trustedcomputinggroup.org/resource/tcg-trusted-attestation-protocol-tap-use-cases-for-tpm-families-1-2-and-2-0-and-dice/

IETF

[IETF CoSWID] Concise Software Identification Tag (CoSWID), https://datatracker.ietf.org/doc/draft-ietf-sacm-coswid/, https://github.com/sacmwg/draft-ietf-sacm-coswid, https://datatracker.ietf.org/doc/draft-birkholz-rats-coswid-rim/

[IETF SUIT] Software Update for Internet of Thing (SUIT), https://datatracker.ietf.org/group/suit/, https://datatracker.ietf.org/doc/draft-ietf-suit-manifest/

[IETF RATS] Remote Attestation Procedures Architecture, https://github.com/ietf-rats-wg/architecture, https://tools.ietf.org/id/draft-ietf-rats-architecture-02.xml

[IETF EAT] Entity Attestation Token, https://tools.ietf.org/html/draft-ietf-rats-eat-02, https://tools.ietf.org/html/draft-mandyam-eat-01

[RFC7049] Concise Binary Object Presentation (CBOR), https://tools.ietf.org/html/rfc7049, http://cbor.io/

[RFC8152] CBOR Object Signing and Encryption (COSE), https://tools.ietf.org/html/rfc8152

[RFC8610] Concise Data Definition Language (CDDL), https://tools.ietf.org/html/rfc8610

[RFC8392] CBOR Web Token (CWT), https://tools.ietf.org/html/rfc8392

[RFC7515] JSON Web Signature (JWS), https://tools.ietf.org/html/rfc7515

Intel

[Intel CLA] Intel Compute Lifecycle Assurance, https://www.intel.com/content/www/us/en/security/compute-lifecycle-assurance.html

[Intel TSC] Intel Transparent Supply Chain, https://www.intel.com/content/www/us/en/products/docs/servers/transparent-supply-chain.html, https://servermarketinglibrary.intel.com/intel-transparent-supply-chain

[Intel KGF] Intel Key Generation Facility, https://intel-epid-sdk.github.io/ecosystem/

[Intel PDT] Intel Processor Diagnostic Tool, https://www.intel.com/content/www/us/en/support/articles/000005567/processors.html

[Intel FSP] Intel FSP, https://www.intel.com/content/www/us/en/intelligent-systems/intel-firmware-support-package/intel-fsp-overview.html

EDKII

[EDKII Trusted Boot] https://tianocore-docs.github.io/edk2-TrustedBootChain/release-1.00/edk2-TrustedBootChain-release-1.00.pdf

Book

[Building Secure Firmware] Building Secure Firmware, https://link.springer.com/book/10.1007/978-1-4842-6106-4

Web

[Attestation1] TPM Remote Attestation Best Known Methods, https://tpm2-software.github.io/tpm2-tss/getting-started/2019/12/18/Remote-Attestation.html

[Attestation2] Using TPM: Machine Authentication and Attestation, https://opensecuritytraining.info/IntroToTrustedComputing_files/Day2-1-auth-and-att.pdf

[Attestation3] Principles of Remote Attestation, https://web.cs.wpi.edu/~guttman/pubs/good_attest.pdf

[JSON] JavaScript Object Notation, https://www.json.org/

[XML] Extensible Markup Language, https://www.w3.org/XML/, http://www.w3.org/2000/09/xmldsig#enveloped-signature, http://www.w3.org/2001/04/xmlenc#sha256, https://www.w3.org/TR/xmldsig-core1/

§

Appendix B: Example

A. RIM Example (ini format)

Refer to https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspRimTemplate.ini.

B. Example of SWID tag (XML format)

One-Binary Mode Manifest Info

Refer to
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspSwidTemplate.xml

Separation Mode Manifest Info

Refer to
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspSwidTemplate2.xml

C. Example of CoSWID tag (CBOR format, described in JSON format)

One-Binary Mode Manifest Info

Refer to
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspCoSwidTemplate.cbor
and
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspCoSwidTemplate.json

Separation Mode Manifest Info

Refer to
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspCoSwidTemplate2.cbor
and
https://github.com/jyao1/FSP/blob/FspAttestation/Tools/FspCoSwidTemplate2.json

§