GlobalPlatform Technology

SESIP Profile for Code Update Mechanism

Version 0.0.0.15

Public Review #2

February 2025

Document Reference: GPS_SPE_026

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Tables

Figures

1 INTRODUCTION

GlobalPlatform's *Security Evaluation Standard for IoT Platforms Methodology* ([SESIP]) specifies general requirements for Security Functional Requirements and Security Assurance Requirements designed to evaluate and certify IoT products.

SESIP Profiles define the requirements for the security features and evaluation activities to be addressed while evaluating a platform (part) of the type targeted by the profile.

This document provides a Profile for security evaluations of the Code Update Mechanism. It can be used as a standalone profile, but in most cases, it will be used to complete a profile of a "main" integrating platform.

1.1 Audience

This document is intended primarily for security architects and platform and system developers.

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://globalplatform.org/specifications/ip-disclaimers/. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

This section lists references applicable to this specification. The latest version of each reference applies unless a publication date or version is explicitly stated.

Table 1-1: Normative References

Standard / Specification	Description	Ref
GP_FST_070	GlobalPlatform Technology Security Evaluation Standard for IoT Platforms (SESIP) Methodology v1.2, July 2023	[SESIP]

Table 1-2: Informative References

Standard / Specification	Description	Ref
EN 17927	Security Evaluation Standard for IoT Platforms	[EN 17927]
UN Regulation No. 156	Software update and software update management system	[UNR 156]

1.4 Terminology and Definitions

Selected terms used in this document are included in Table 1-3.

Table 1-3: Terminology and Definitions

Term	Definition
Code Update Authentication Key	The public key or secret symmetric key used to verify the authenticity of a presented Code Update Image.
Code Update Confidentiality Key	The private key or secret symmetric key used to decrypt the Code Update Image.
Code Update Image	The code downloaded to the Platform and used to update or replace the current Platform, in whole or in part.
Code Update Version	The attribute of the Code Update Image presented to the Platform during the code update process.
Current Code Version	The version of the last successfully installed Code Update Image stored persistently in the Platform.

1.5 Abbreviations

Table 1-4: Abbreviations

Abbreviation	Meaning
AN	Application Note
SAR	Security Assurance Requirements
SFR	Security Functional Requirements

1.6 Revision History

GlobalPlatform technical documents numbered *n*.0 are major releases. Those numbered *n*.1, *n*.2, etc., are minor releases where changes typically introduce supplementary items that do not impact backward compatibility or interoperability of the specifications. Those numbered *n.n*.1, *n.n*.2, etc., are maintenance releases that incorporate errata and clarifications; all non-trivial changes are indicated, often with revision marks.

Table 1-5: Revision History

Date	Version	Description
May 2024	0.0.0.10	Committee Review
Jun 2024	0.0.0.11	Member Review
Aug 2024	0.0.0.12	Public Review
Feb 2025	0.0.0.15	Public Review #2
TBD	v1.0	Public Release

2 SESIP PROFILE INTRODUCTION

Software updates with the latest security patches and bug fixes are widely agreed upon (see [EN 17927], [SESIP], [UNR 156]) as necessary for any device that claims security, safety, or privacy protection.

In this document, we use the term Code Update Mechanism to refer to the variety of technical software and hardware means implemented in the device and its operational environment, as well as the management actions required to operate them, to perform software updates and code patching.

This document defines requirements, technical guidelines, and recommendations for implementing the Code Update Mechanism securely and efficiently.

Application Notes (mandatory) for Security Target writing are marked as AN. Content specific to the case of the Security Code Update Mechanism being a part of an integrating Platform is marked as [For platform part].

2.1 Platform [part] Reference

Table 2-1: Platform [part] Reference

Platform [part] name	< Code Update Mechanism name >
Platform [part] version	< Code Update Mechanism version >
Platform [part] identification	< Code Update Mechanism identification details >
Platform [part] type	Code Update Mechanism to be integrated into the IoT platform(s)

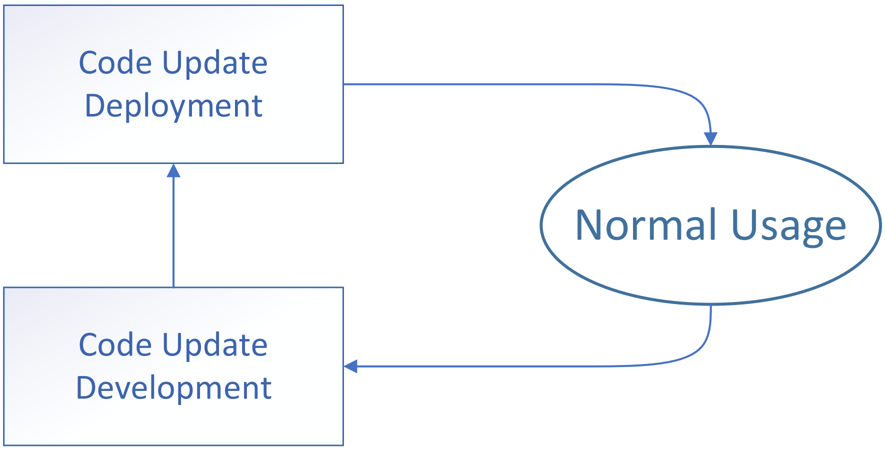
2.2 Platform [part] Functional Overview and Description

The target of this document is the Code Update Mechanism in Secure Consumer IoT devices.

Within the Normal usage phase (see [SESIP]), connected devices experience a continuous cycle of software updates. Promptly developing and deploying security updates is critical for companies to safeguard their customers and contribute to the overall health of the technical ecosystem.

The following diagram incorporates software updates into the Connected Product Life Cycle description of [SESIP]:

Figure 2-1: Software Update in the Normal Usage Phase



The Code Update Mechanism is the variety of technical software and hardware means implemented in the device and its operational environment, as well as the management actions required to operate them, to perform software updates and code patching.

Table 2-2: Assets and Their Protected Attributes

Asset	Protected Secure Attributes
Code Update Authentication Key	Integrity and authenticity. For a symmetric key, also confidentiality.
Code Update Confidentiality Key	Confidentiality, integrity, and authenticity.
Code Update Image	Integrity, authenticity, and freshness. Optionally – confidentiality.
Code Update Version	Integrity and authenticity.
Current Code Version	Integrity, authenticity, and anti-rollback protection.

2.2.1 Physical and Logical Scope

AN The Security Target shall precisely describe the physical and logical scope of the Code Update Mechanism and provide all necessary details required to identify the hardware mechanisms and software libraries, drivers, etc., that constitute it, together with their versions and any other relevant information. Additional details about the security features supported by the Code Update Mechanism must be described.

2.3 Life Cycle

AN The Platform Security Target shall integrate any life cycle information specific to the Code Update Mechanism design and deployment into its life cycle section.

The description must present an overview of the main phases, from the hardware and software design to the product’s end-of-life; each phase must identify all possible state(s). It must also explain how transitions between those states are secured.

The description must include how the provisioning of the relevant assets (see Table 2-2) is performed (directly in the ST or referencing an appropriate guidance document).

3 Security Objectives for the Operational Environment

For the platform to fulfill its security requirements, the operational environment (technical or procedural) must fulfill the following objectives.

3.1 Generic Security Objectives

- Any security guidance for integrating and deploying the Code Update Mechanism shall be followed.
- If the code update authentication is based on a secret symmetric key (i.e., not a public part of an asymmetric key), the key shall be unique for each instance of the Platform.
- Keys used for code update authenticity and confidentiality shall be generated with the required cryptographic strength and entropy amount.
- Keys used for code update authenticity and confidentiality shall be securely provisioned to the Platform and kept secret by the Platform and at the Code Update Deployer’s facility.

3.2 Security Objectives for Code Update Deployer

Keeping software updated with the latest security patches and bug fixes cannot be purely the user’s responsibility. It requires a system-wide role for an entity responsible for Code Update Deployment. The Code Update Deployer plays a critical role in the secure delivery and activation of Code Update Images to devices.

The security objectives for Code Update Deployer typically encompass ensuring the integrity and authenticity of the code updates, safeguarding against unauthorized access during the deployment process, and verifying the secure activation of the updates on the devices. Given the context-sensitive nature of these objectives, it is imperative to analyze the operational environment and potential threats to establish robust and relevant security goals.

AN The “Security Objectives for the Operational Environment” section of the corresponding Security Target must clearly outline the security goals tailored to the specific deployment context.

In any case, the following objectives apply:

- The Code Update Deployer shall use the correct authentication and confidentiality keys for Code Update Image protection.
- The Code Update Deployer shall increment the Code Update Version for subsequent updates.

3.3 Security Objectives for Integrating Platform

AN The objectives of this section shall be in the Security Target only when the Code Update Mechanism is considered a standalone platform.
If the Code Update Mechanism is evaluated as part of an underlying secure platform, those objectives relate to the platform itself and shall be made parts of the corresponding SFRs.

- Before the updated code is executed for the first time, the integrating platform shall verify that the code update has been completed. If not, the previous code version shall be restored, or the platform shall retreat to a secure state without compromising its security assets.
- Secure Initialization of the integrating platform shall cover the Code Update Mechanism.

4 SECURITY ASSURANCE REQUIREMENTS

The claimed assurance requirements package is SESIP2 as defined in [SESIP].

This level can be raised as described in section 6.

Note that when the Code Update Mechanism is part of the including platform, its assurance level can differ from the platform's level as allowed by the multi-assurance concept (see [SESIP] section 4.5.3.3).

4.1 Flaw Reporting Procedure (ALC_FLR.SESIP)

In accordance with the requirement for a flaw reporting procedure (ALC_FLR.SESIP), including a process to report flaws and generate and distribute any needed updates, the developer has defined the following procedure:

<Describe the procedure, including where flaws and security incidents can be reported (website and/or email address), how the reported flaws are handled in a timely manner, and how an application developer/end-user can get informed of the update. If the "Secure update of platform" SFR is removed, you have to provide a strong argumentation here why the platform is not worth getting an update. However, the process for receiving the reports of flaws and handling them in a timely manner needs to be described in any case.>

AN Vulnerabilities often arise even from non-security-related software components. It is essential to keep all software up-to-date and well-maintained. Therefore, all software components within connected IoT devices shall be securely updateable.

AN When a code update is necessary, it shall be applied promptly. This can be achieved by informing device users about the need to update, prompting them to perform updates when needed, or executing the updates automatically whenever possible.

AN A code update should be easy to perform. It should occur in the background and not disrupt the device's functionality until the updated version is activated.

5 Security Requirements and Implementation

The refinements added in the sections below aim to help the developers implement the Code Update Mechanism efficiently and securely and to help the evaluators assess and validate the implementation.

AN If the Code Update Mechanism is evaluated as a part of an underlying secure platform, the following requirements must be merged with the platform's SFRs. In this case, the term "platform" in the SFRs refers to the platform integrating the Code Update Mechanism.

AN Refinements that begin "[For SESIP3]" are required in order to claim SESIP3.

5.1 Verification of Platform Identity

5.1.1 Refinement: Verification of Current Code Version

The Platform shall provide a means for the Code Update Deployer to verify the current version of the Platform code.

[For SESIP3]

- A cryptographic mechanism shall be implemented to prevent an attacker from falsifying the Current Code Versions of the Platform.
- The platform shall provide a cryptographic mechanism to prevent an attacker from interfering with verifying the current version of the Platform code.

5.2 [For platform part] Secure Initialization of Platform

AN A Secure Initialization of the integrating Platform shall cover the Code Update Mechanism. If the Platform Security Target does not claim this SFR, then this one is not applicable and can be omitted.

5.3 Secure Update of Platform

AN The SESIP requirement that "the platform can be updated to a newer version" means, in particular, that the Platform shall prevent updates where the Code Update Version is lower than the Current Code Version. By implementing this restriction, the Current Code Version becomes a Reliable Index as defined in [SESIP].

5.3.1 Refinement: Security Assets Protection

The platform shall protect the security assets regarding their secure attributes as described in Table 2-2: Assets and Their Protected Attributes.

5.3.2 Refinement: Code Update Availability

[For SESIP3]

- Watchdog timers, expirable entitlements, or similar mechanisms shall be employed to prevent an attacker from obstructing platform code updates.

5.3.3 Refinement: Code Update Feasibility

Maintaining essential functionality is critical during a Platform code update, especially if the code update takes time.

[For SESIP3]

- Code Update Image download shall be performed in the background without disrupting the device’s functionality.

Subsequently, the new code can be activated using memory remapping or a similar mechanism.

5.3.4 Refinement: Code Update Authenticity and Integrity

The authenticity of the presented Platform Code Update Image shall be verified using a dedicated Code Update Authentication Key.

[For SESIP3]

- An asymmetric signature mechanism shall be used for code update verification, which does not necessitate the confidentiality of the image verification key.

5.3.5 Refinement: Code Update Confidentiality [optional]

The Platform shall provide means for keeping the contents of the Code Update Image confidential before it is applied.

6 MAPPING AND SUFFICIENCY RATIONALES

- Secure Consumer IoT devices certified to SESIP Assurance Level 2 shall implement the Code Update Mechanism according to the Security Functional Requirements and mandatory refinements listed in section 5.
- Secure Consumer IoT devices certified to SESIP Assurance Level 3 and higher shall also implement the refinement marked “[For SESIP3]”.

6.1 SESIP Level Sufficiency

As the SESIP level is directly used per [SESIP], the sufficiency is also per [SESIP].

Annex A SECURITY PROBLEM DEFINITION

A.1 Threat Analysis

The following threats have been identified:

- Blocking Code Updates
- Forging Code Update Deployer Authorization
- Code Update Mechanism Abuse
- Rollback of a Code Update
- Partial Code Update
- Code Update Image Disclosure

To cover those threats, the following security objectives for the Code Update Mechanism and the associated security features have been identified:

A.1.1 Current Code Version Attestation

The Code Update Deployer shall be able to obtain the Current Code Version of the Platform to determine whether the code running on it is outdated.

A.1.2 Code Update Attainability

The Platform should employ countermeasures to prevent an attacker from blocking code updates for the Platform.

At least, the Code Update Deployer shall inform the Platform User when a code update is scheduled for the Platform.

A.1.3 Code Update Feasibility

The Code Update process shall not risk the functioning of the Platform during and after the update.

A.1.4 Code Update Authenticity and Integrity

The Platform shall reject non-authentic or corrupt Code Update Images.

A.1.5 Anti-Rollback Protection

The Platform shall protect from applying outdated code updates, e.g., by ensuring that the code version increases on update.

A.1.6 Atomicity of the Code Update

The Platform shall conduct the code updates in an all-or-nothing manner, i.e., prohibiting the execution of any part of the new code before and any part of the deprecated code after the successful activation of the new code.

A.1.7 Secure Failure

If a failure occurs during Code Update activation, the Platform shall retreat to a secure state without compromising its security assets and, preferably, without losing its functionality.

A.1.8 Patch Confidentiality

The Platform may optionally provide means for keeping the contents of the Code Update Image confidential before it is applied; e.g., a secure communication channel for its retrieval, allowing its contents to be encrypted before activation.

A.2 Security Objective Coverage

Table A-1: Security Objective Coverage

Security Objectives	Security Functional Requirements
Current Code Version Attestation	5.1
Code Update Availability	5.3.2
Code Update Feasibility	5.3.3
Code Update Authenticity and Integrity	5.3.4
Rollback Protection	5.3
Atomicity of the Code Update	3.3
Secure Fail	3.3
Patch Confidentiality	5.3.5