Written by: Aurel Sorin Spornic
Quang-Huy Nguyen

Odysseus *Platform* Security Target

PUBLIC VERSION

CONTENT

Figures

Tables

1 ST introduction

1.1 ST Identification

Title:	ODYSSEUS Platform Security Target (Public version)
Version:	1.5 issued January 24th, 2008
Registration:	Ref. D1049224
Origin:	GEMALTO
Commercial name:	MultiApp ID
Product ref.	T1002711

The TOE is composed with:

Component	Version number	Supplier
Hardmask in ROM	1.0	Gemalto
Corrective softmask #1 in EEPROM, PDM ref. S10325940[1]	3.1	Gemalto
Micro-controller SLE66CX680PE	A13	Infineon
RMS library	2.5	Infineon
RSA2048 library	1.4	Infineon

This evaluation is done under the French scheme for Common Criteria. The certification body is the 'Direction Centrale de la Sécurité des Systèmes d'Information' (DCSSI).

1.2 ST overview

The product is compliant with Java Card 2.1.1 and has also some JC 2.2.1 features as Application and Object deletion (it is a JavaCard system 2.2.1 without the logical channels and RMI). It is conformant to Global Platform 2.1.1. Therefore it enables value added services to be installed on, securely executed and deleted from the card.

This TOE provides the security of an EAL4+ evaluated card with the flexibility of an open platform.

It allows the loading of applets before or after the issuance of the card. These applets MAY or MAY NOT be EAL4+ evaluated on this platform.
The applications using only EAL4+ applets WILL BE EAL4+ even if NOT EAL4+ applets are loaded on the platform, provided the secure loading rules are obeyed.
The applications using a NOT EAL4+ applets will NOT BE EAL4+.

The Issuer can forbid the loading of applets before or after the issuance of the card.

[1] The softmask is a PDM item attached to the technical product T1002711.

The Java Card technology combines a subset of the Java programming language with a runtime environment optimized for smart cards and similar small-memory embedded devices [JCVM211]. The Java Card platform is a smart card platform enabled with Java Card technology (also called a "Java card"). This technology allows for multiple applications to run on a single card and provides facilities for secure interoperability of applications. Applications for the Java Card platform ("Java Card applications") are called applets.

The Security Target focuses on the following security functions:

- **Hardware Tamper Resistance:**
  - This is the chip security layer that meets PP SSVG [PP/BSI-0002].
- **Secure operation of the Java Card Virtual Machine (meet PP JCS):**
  - This is the Java Card Virtual Machine and Operating System that meets [PP/JCS].

  The security requirements used from [PP/JCS], Javacard 2.1.1 standard configuration, are those from the groups: *CoreG* (§5.1.1), *InstG* (§5.1.2)and *CarG* (§5.1.8) with the addition of two groups that appear in Javacard 2.2 standard configuration, *ADELG* (§5.1.4), *ODELG* (§5.1.7).

  The IT environment security requirements are the ones defined in the groups *BCVG* (§5.1.3). Due to the extension of the TOE scope compared to its PP definition, the groups *SCPG* (§5.1.9) and *CMGRG* (§5.1.10) became TOE security requirements.

1.3 CC conformance

The compliance is assumed with CC version V2.3 (ISO 15408) ([CC-1], [CC-2], [CC-3]).

This product is a Java Card compliant to the Protection Profile Java Card 2.1.1 Standard from SUN [PP/JCS] and uses a certified chip conformant to the Protection Profile SSVG (also known as PP/BSI-0002) from Eurosmart, providing the necessary security so that value added applications can be safely loaded and executed on card. Therefore the ST is conformant to [PP/JCS] Protection Profile and IC is conformant to [PP/BSI-0002] Protection Profile.

Group (group name)	**Java Card System Standard 2.1.1**	**Java Card System Standard 2.2**	**TOE**
Core (*CoreG*)	TOE	TOE	TOE
Smart card platform (*SCPG*)	IT	IT	TOE
Installer (*InstG*)	TOE	TOE	TOE
RMI (*RMIG*)	--	TOE	--
Logical channels (*LCG*)	--	TOE	--
Object deletion (*ODELG*)	--	TOE	TOE
Bytecode verification (*BCVG*)	IT	IT	no (IT)
Applet deletion (*ADELG*)	--	TOE	TOE
Secure carrier (*CarG*)	TOE	TOE	TOE
Card manager (*CMGRG*)	IT	IT	TOE

Table 1 Relationship between JC PP Groups and Configurations versus ST

This ST is CC V2.3 conformant with Part 2.
This ST is CC V2.3 conformant with Part 3 conformant and EAL4 augmented as stated in [PP/JCS], [PP/BSI-0002].

The assurance level for this product is EAL4 augmented by:

- ADV_IMP.2 (Development – Implementation of the TSF)

- ALC_DVS.2 (Sufficiency of security measures)
- AVA_VLA.4 (Vulnerability Assessment – Analyse, Highly resistant)
- AVA_MSU.3 (Vulnerability Assessment – Analysis and testing for insecure states).

The augmentations may also be found in section §5.2.
The strength level for the TOE security functional requirements is "SOF high" (Strength Of Functions high).

1.4 References

1.4.1 External References [ER]

Reference	Title
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model CCIMB-2005-08-001, version 2.3, August 2005 (conform to ISO 15408).
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements CCIMB-2005-08-002, version 2.3, August 2005 (conform to ISO 15408).
[CC-3]	Common Criteria for Information Technology security Evaluation Part 3: Security Assurance Requirements CCIMB-2005-08-003, version 2.3, August 2005 (conform to ISO 5408).
[CEM]	Common Methodology for Information Technology Security Evaluation CCIMB-2005-08-004, version 2.3, August 2005.
[PP/JCS]	Java Card System Protection Profile Version 1.0b issued August 2003, standard 2.1.1 configuration, SUN document, registered by DCSSI under PP/0304.
[PP/BSI-0002]	Smart Card IC Platform Protection Profile, version 1.0, registered by BSI in 2001 under PP-BSI-0002, Eurosmart document (SSVG Protection Profile).
[ST/INFINEON]	Security Target of SLE66CX680PE Integrated Circuit.
[FIPS 46-3]	FIPS 46-3: DES Data Encryption Standard (DES and TDES). National Institute of Standards and Technology
[SP 800-38 A]	NIST Special Publication 800-38A: Recommendation for Block Cipher Modes of operation
[FIPS 197]	FIPS 197: AES Advanced Encryption Standard. National Institute of Standards and Technology.
[RSA PKCS#1]	PKCS #1 v2.1: RSA Cryptography Standard
[FIPS 180-2]	FIPS-46-3: Secure Hash Standard (SHA). National Institute of Standards and Technology.
[ISO 7816-4]	Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange
[ISO 7816-6]	Identification cards - Integrated circuit(s) cards with contacts, Part 6: Interindustry data elements
[ISO 7816-9]	Identification cards - Integrated circuit(s) cards with contacts, Part 9: Additional Inter industry commands and security attributes.
[ISO 9796-2]	ISO/IEC 9796-2
[JCAPI221]	Application Programming Interface Java Card™ Platform, version 2.2.1 Sun Microsystems, Inc., June 23, 2003
[JCRE221]	Runtime Environment Specification Java Card™ Platform, version 2.2.1 Sun Microsystems, Inc., June 2003

Reference	Title
[JCVM221]	Virtual Machine Specification Java Card™ Platform, version 2.2.1 Sun Microsystems, Inc., June 2003
[JCAPN221]	Application Programming Notes for the Java Card™ Platform, Sun Microsystems, Inc, version 2.2.1, October 2003.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[GP]	Global Platform. Card Specification – v2.1.1, March 2003.
[VOP]	Visa Open Platform Card Implementation Requirements Configuration 3 – v2.0

1.4.2 Internal References [IR]

Reference	Title
[FSP_ODYSSEUS]	**ODYSSEUS Functional Specification** D1050016 (FSP_D1050016)
[HLD_ODYSSEUS]	**ODYSSEUS High-level Design (overview document)** D1049625 (HLD_D1049625)
[LLD_ODYSSEUS]	**ODYSSEUS Low-level Design (overview document)** D1050412 (LLD_D1050412)
[IMP_ODYSSEUS]	**ODYSSEUS Implementation representation** D1050415 (IMP_D1050415)
[SPM_ODYSSEUS]	**ODYSSEUS Security Policy Model** D1050410 (SPM_D1050410)
[AUT_ODYSSEUS] [CAP_ODYSSEUS] [SCP_ODYSSEUS]	**ODYSSEUS Partial CM automation** **ODYSSEUS Generation, support and acceptance procedure** **ODYSSEUS Problem tracking CM coverage** D1049554 (ACM_D1049554)
[DEL_ODYSSEUS]	**ODYSSEUS Detection of modification** D1051233 (DEL_D1051233)
[IGS_ODYSSEUS]	**ODYSSEUS Installation, Generation and Start Up Procedures** D1050210 (IGS_D1050210)
[ADM_ODYSSEUS]	**ODYSSEUS Administrator Guidance** D1050012 (ADM_D1050012)
[USR_ODYSSEUS]	**ODYSSEUS User Guidance** D1050014 (USR_D1050014)
[DVS_ODYSSEUS]	**ODYSSEUS Development Security Documentation** D1049550 (DVS_D1049550)
[LCD_ODYSSEUS]	**ODYSSEUS Life-cycle definition documentation** D1051235 (LCD_D1051235)
[TAT_ODYSSEUS]	**ODYSSEUS Documentation of development tools** D1050010 (TAT_D1050010)
[COV_ODYSSEUS]	**ODYSSEUS Analysis of test coverage** D1050664 (COV_D1050664)
[DPT_ODYSSEUS]	**ODYSSEUS Analysis of the depth of testing** D1050662 (DPT_D1050662)

Reference	Title
[FUN_ODYSSEUS]	**ODYSSEUS Test Documentation** D1050661 (FUN_D1050661)
[MSU_ODYSSEUS]	**Odysseus Analysis and testing for insecure states** D1051227 (MSU_D1051227)
[SOF_ODYSSEUS]	**ODYSSEUS Strength of TOE security functions analysis** D1051229 (SOF_D1051229)
[VLA_ODYSSEUS]	**ODYSSEUS Vulnerability Analysis** D1051231 (VLA_D1051231)
[PERS_ODYSSEUS]	**ODYSSEUS Personalisation Manual** D1051441

1.4.3 Reference of Associated Distribution Sheet

[DS_ODYSSEUS]	**ODYSSEUS Security Target Distribution sheet** D1049710

2 TOE Description

This part of the ST describes the TOE as an aid to the understanding of its security requirements.
It addresses the product type, the smart card product life cycle, the TOE environment along the smart card life cycle and the general features of the TOE.

2.1 Product type

The Target of Evaluation (TOE) is the Smart Card Integrated Circuit with Embedded Software in operation and in accordance to its functional specifications.

The TOE is composed of:

- Integrated Circuit including crypto libraries, which is certified separately according to [ST/INFINEON] claiming [PP/BSI-0002],
- Operating System
- Card Manager & Open Platform functionalities [GP].
- Java Card Virtual Machine 2.2.1 [JCVM221] and most of the features of Java Card Runtime Environment 2.2.1 [JCRE221] and Java Card API 2.2.1 [JCAPI221]. Logical Channels and RMI are not supported.

The TSF are composed of:

- The chip and its crypto library,
- The GEOS operating system,
- The Card Manager loader (applets and packages loading),
- The Java Card System. The TOE is conformant to Java Card 2.1.1 Standard configuration of [PP/JCS], as it misses Logical Channels and RMI groups.

As this ST is a composition, each chapter indicates the items covered by the IC security target.

Next picture represents the TOE, which doesn't include the applets that may be loaded post-issuance.

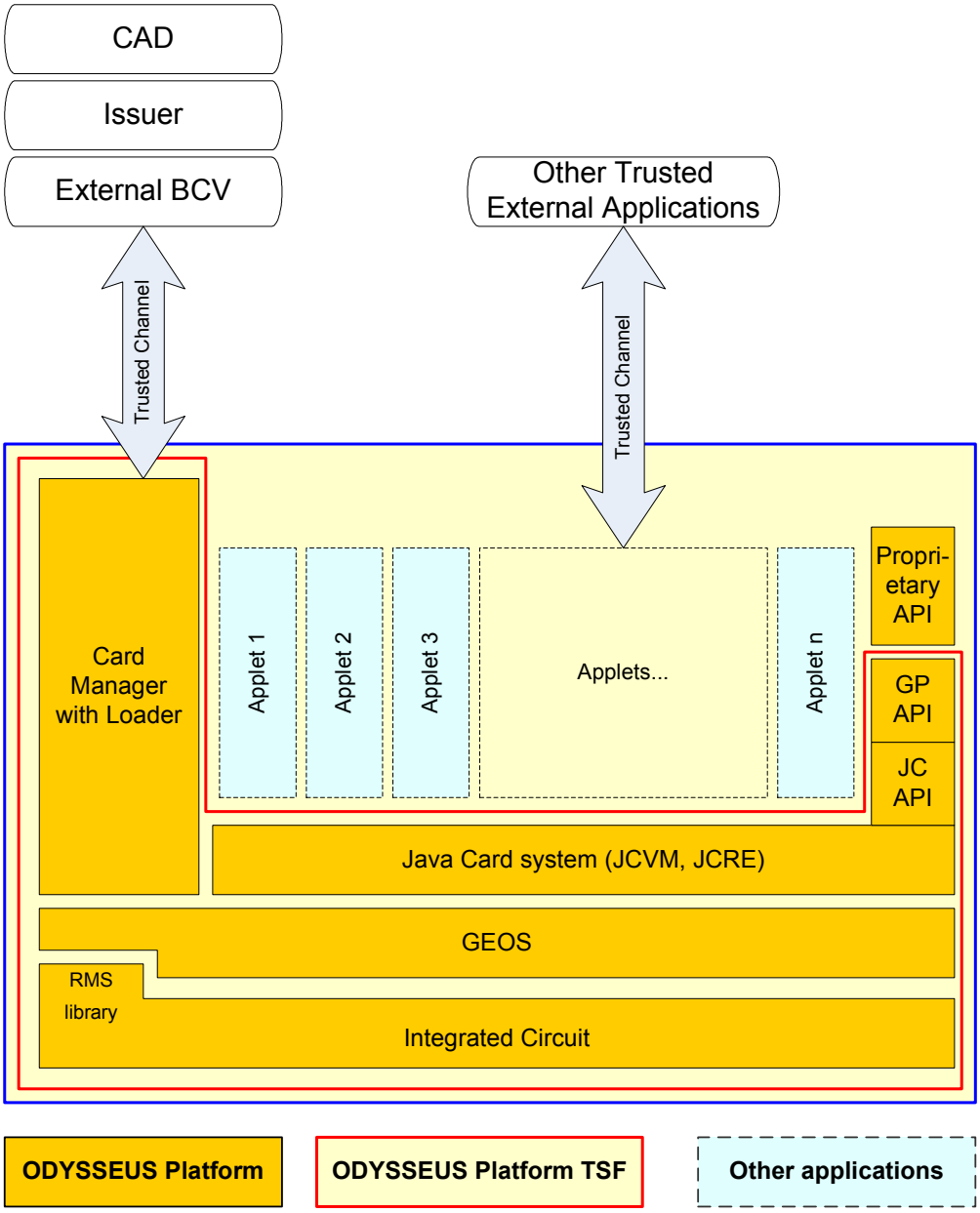

Figure 2-1 – The TOE

2.2 TOE Usage

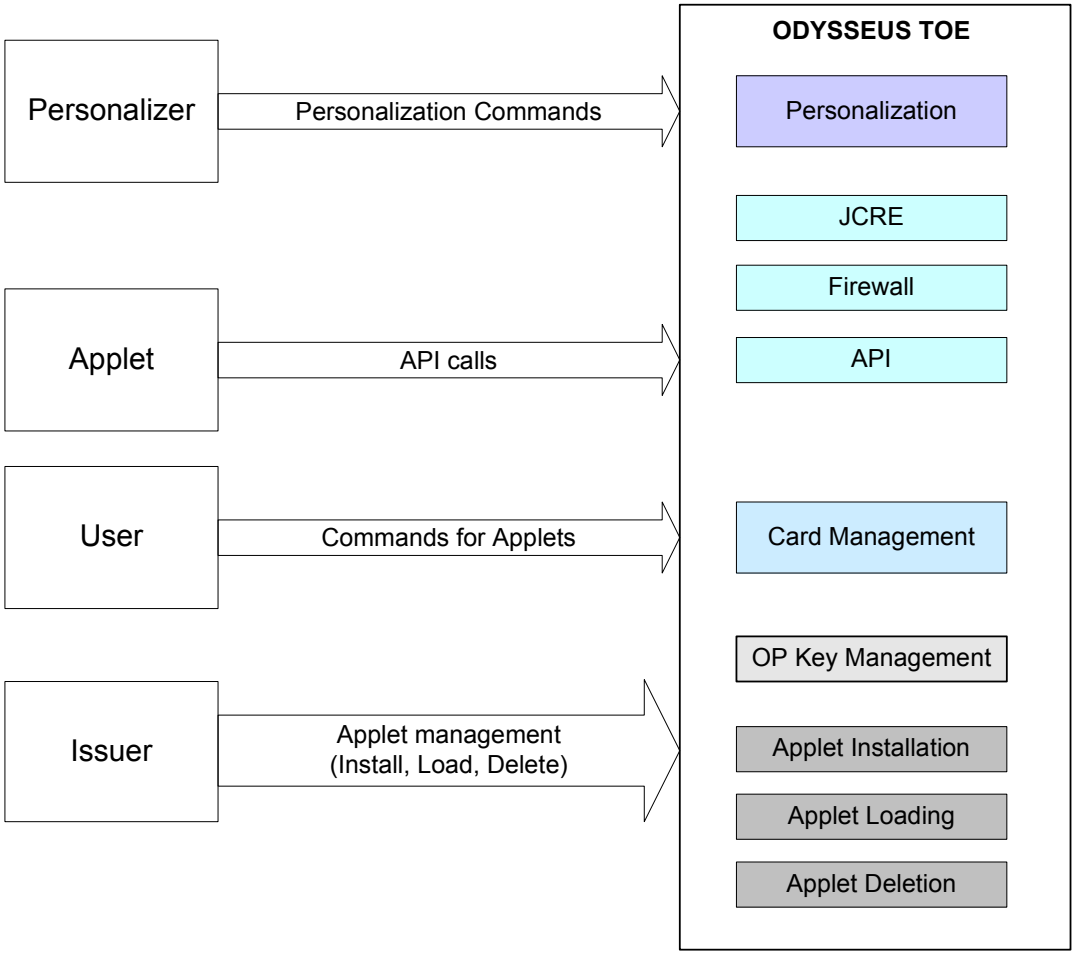

Figure 2-2 – TOE operations

2.2.1.1 Personalization Phase

During the Personalization Phase the following Administrative Services are available:

- Applet Load
- Applet Install
- Applet Delete
- Applet Management Lock

All applet management operations require the authentication of the Issuer. By erasing the authentication keys with random numbers, the Issuer can prevent all subsequent applet management operations. This operation is not reversible.
In the Personalization phase, Applet Management Lock is optional.

2.2.1.2 Usage Phase

During the Usage Phase, if the Applet Management lock has not been put, the Administrative Services are available as during the Personalization phase:

- Applet Load
- Applet Install
- Applet Delete
- Applet Management Lock

In addition, the following User services are available:

- Applet Selection
- Applet Interface

2.3 Smart Card Products Life-cycle

The Smart card product life cycle, as defined in [PP/SSVG], is split up into 7 phases where the following authorities are involved:

Phase 1	Smart card software development	**The smart card embedded software developer** is in charge of the smart card embedded software development and the specification of IC pre-personalization requirements.
Phase 2	IC Development	**The IC designer** designs the integrated circuit, develops IC firmware if applicable, provides information, software or tools to the smart card software developer, and receives the software from the developer, through **trusted delivery and verification procedures**. From the IC design, IC firmware and smart card embedded software, he constructs the smart card IC database, necessary for the IC photomask fabrication.
Phase 3	IC manufacturing and testing	**The IC manufacturer** is responsible for producing the IC through three main steps: IC manufacturing, testing, and IC pre-personalization.
Phase 4	IC packaging and testing	**The IC packaging manufacturer** is responsible for the IC packaging and testing.
Phase 5	Smart card product finishing process	**The smart card product manufacturer** is responsible for the smart card product finishing process and testing, and the smart card pre-personalization
Phase 6	Smart card personalization	**The Personaliser** is responsible for the smart card personalization and final tests.
Phase 7	Smart card end-usage	**The smart card issuer** is responsible for the smart card product delivery to **the smart card end-user**, and for the end of life process.

The JCS Platform life Cycle as described in the standard smart cards life cycle can be matched as shown in Figure 2.3 – TOE Life Cycle.

OS design and **JCVM design** correspond to life phase 1 "Smart card software development".
Hardware design corresponds to life phase 2 "IC development".
Hardware fabrication OS and JCVM embedding correspond to life phase 3 "IC manufacturing and testing", phase 4 "IC packaging and testing", phase 5 "Smart card product finishing process".

The global security requirements of the TOE mandate to consider, during the development phase, the threats to security occurring in the other phases. This is why this ST addresses the functions used in phases 6 and 7 but developed during phases 1 to 5.
The limits of the evaluation process correspond to phases 1 to 5 including the TOE under development delivery from the party responsible of each phase to the parties responsible of the following phases.

These different phases may be performed at different sites. This implies that procedures on the delivery process of the TOE must exist and be applied for every delivery within a phase or between phases. This includes any kind of delivery performed from phase 1 to 5 to subsequent phases, including:

- Intermediate delivery of the TOE or the TOE under construction within a phase,
- Delivery of the TOE or the TOE under construction from one phase to the next.

These procedures must be compliant with the security assurance requirements developed later in this document.

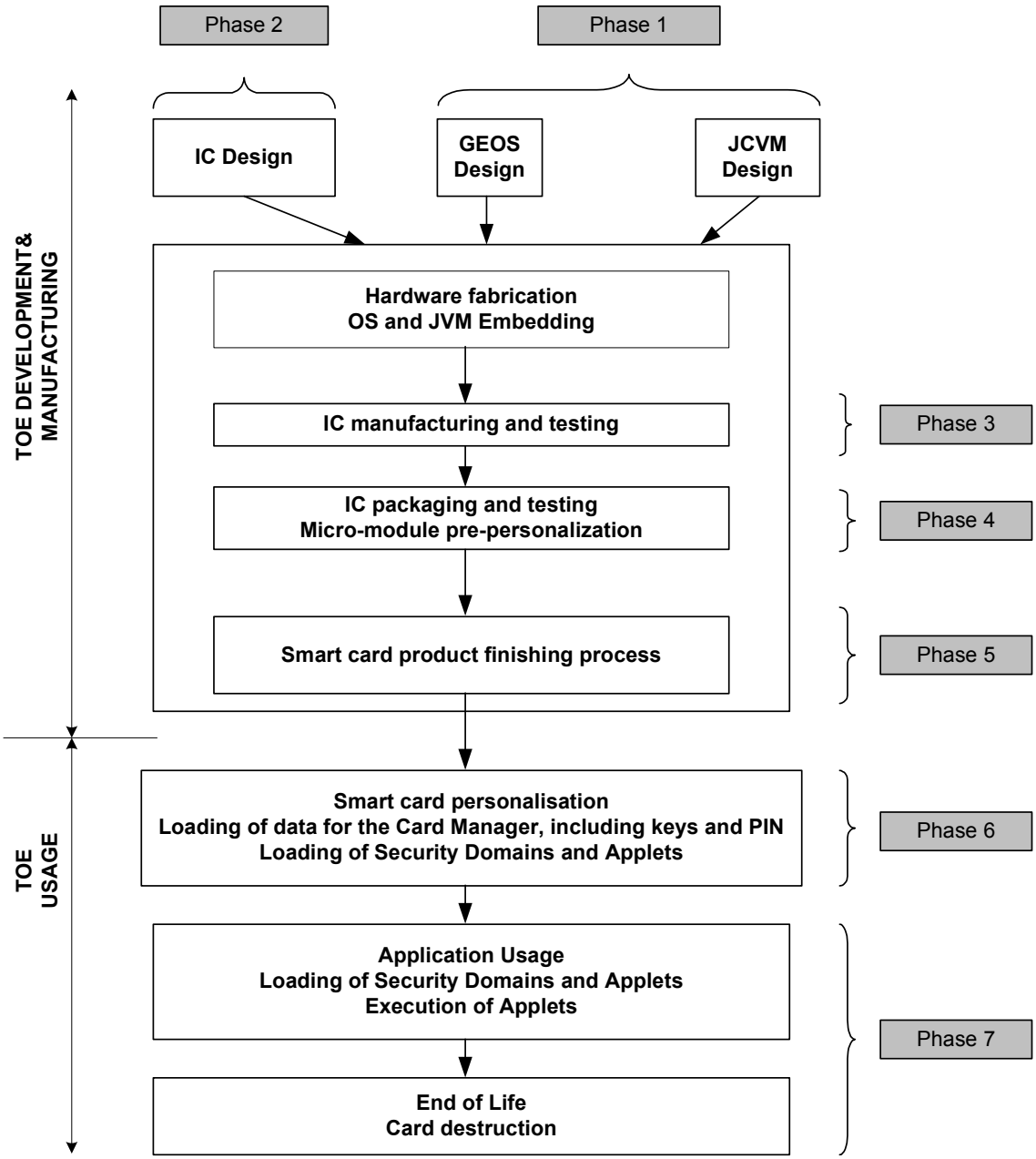

Figure 2-3 – TOE Life Cycle

2.4 TOE Environment

Considering the TOE, four types of environment are defined:

- Development and fabrication environment (phase 1 to 4),
- Initialization environment corresponding to smart card pre-personalization (phase 5)
- Personalization environment, during which the card loads and installs applets (phase 6),
- User environment, during which the card owner uses the Applets loaded on the card. In this environment the card can also load, install and delete applets (phase 7),
- End of life environment, during which the TOE is made inapt for any operation (end of the phase 7).

2.4.1 TOE Development & Production Environment

The TOE described in this ST is developed in different places as indicated below:

IC design	Infineon München
Secure OS Design	Gemalto Meudon
JavaCard Platform design	Gemalto Meudon
IC manufacturing and Testing	Infineon München
IC packaging and testing	Gemalto Orleans and Gemalto Gemenos
PrePersonalization	Gemalto Orleans and Gemalto Gemenos

In order to ensure security, the environment in which the development takes place must be made secure with access control tracing entries. Furthermore, it is important that all authorized personnel feels involved and fully understands the importance and the rigid implementation of the defined security procedures.
The development begins with the TOE specification. All parties in contact with sensitive information are required to abide by Non-disclosure Agreement.
Design and development of the ES then follows. The engineers use a secure computer system (preventing unauthorized access) to make the conception, design, implementation and test performances.
Storage of sensitive documents, databases on tapes, CDs, and printed circuit layout information are in appropriately locked cupboards/safe. Of paramount importance also is the disposal of unwanted data (complete electronic erasures) and documents (e.g. shredding).
Testing, programming and deliveries of the TOE then take place. When these are done offsite, they must be transported and worked on in a secure environment with accountability and tractability of all (good and bad) products.
During the electronic transfer of sensitive data, procedures must be established to ensure that the data arrive, only at the destination and is not accessible at intermediate stages (e.g. stored on a buffer server where system administrators make backup copies). It must also be ensured that transfer is done without modification or alteration.
During fabrication, phases 3, and 4, all the persons involved in storage and transportation operations should fully understand the importance of the defined security procedures.
Moreover, the environment in which these operations take place must be secured.
The TOE Initialization is performed in [Infineon München phase 3; Orleans phase 4 & 5].
In the initialization environment of the TOE, smart card pre-personalization takes place (phase 5).
Initialization requires a secure environment, which guarantees the integrity and confidentiality of operations.

2.4.2 Usage of the TOE

2.4.2.1 Personalization Environment

The personalization (phase 6) is done when the TOE is constructed. However it should take place in a protected environment. During the personalization, applets and data may be loaded into the card.
After the personalization, the TOE is issued to the end User.

2.4.2.2 Usage Environment

Once delivered to the end user (phase 7), the TOE can activate the applets. The TOE can also perform management operations on applets (Load, Install and Delete).
The TOE is owned by the end user and the environment is not secure. Therefore the TOE itself has to ensure that the security requirements are met.

2.4.3 TOE End of life

End of life must be considered for several reasons:
The Keys can be compromised.
The TOE can be stolen.
The TOE physical support can come to the end of its useful life.
In all these cases, it must be ensured that the TOE cannot be used any more.

2.4.4 The actors and roles

The users of the TOE include
People like the card issuer and the card owner,
Hardware like the CAD where the card is inserted
Software components like the application packages installed on the card.

The Issuer can Load, Install, Delete Applets. He can also Lock Applet management.
The Card Owner can activate Applets and functions inside the Applets.
The application packages respond to the Card Owner. They can use functions of the Card by activating the Java API.

2.5 TOE intended usage

The TOE intended usage is the management and the use of applets in a Secure Environment.

3 TOE security environment

3.1 Assets

The assets of the TOE are those defined in [PP/JCS].

The assets of [PP/BSI-0002] are studied in [ST/INFINEON].

3.1.1 User Data

3.1.1.1 Java Card

D.APP_CODE

The code of the *applet*s and libraries loaded on the card.

To be protected from unauthorized modification.

D.APP_C_DATA

Confidential sensitive data of the applications, like the data contained in an object, a static field of a *package*, a local variable of the currently executed method, or a position of the operand stack.

To be protected from unauthorized disclosure.

D.APP_I_DATA

Integrity sensitive data of the applications, like the data contained in an object, a static field of a *package*, a local variable of the currently executed method, or a position of the operand stack.

To be protected from unauthorized modification.

D.PIN

Any end-user's PIN.

To be protected from unauthorized disclosure and modification.

D.APP_KEYs

Cryptographic keys owned by the *applets*.

To be protected from unauthorized disclosure and modification.

3.1.2 TSF Data

3.1.2.1 Java Card

D.JCS_CODE

The code of the *Java Card System*.

To be protected from unauthorized disclosure and modification.

D.JCS_DATA

The internal runtime data areas necessary for the execution of the *JCVM*, such as, for instance, the stack frame, the program counter, the class of an object, the length allocated for an array, any pointer used to chain data-structures.

To be protected from monopolization and unauthorized disclosure or modification.

D.SEC_DATA

The runtime security data of the *JCRE*, like, for instance, the *AID* s used to identify the installed *applets*, the *Currently selected applet*, the *current context* of execution and the owner of each object.

To be protected from unauthorized disclosure and modification.

D.API_DATA

Private data of the API, like the contents of its private fields.

To be protected from unauthorized disclosure and modification.

D.JCS_KEYs

Cryptographic keys used when loading a file into the card.

To be protected from unauthorized disclosure and modification.

D.CRYPTO

Cryptographic data used in runtime cryptographic computations, like a seed used to generate a key.

To be protected from unauthorized disclosure and modification.

3.2 Subjects

3.2.1 Java Card

The main *subjects* of the Java Card PP are described hereafter. One of the subjects listed in §3.3 Users & Subjects is not included, as it is related to RMI, that is not part of JC 2.1.1:

S.CAD: The *CAD* is involved in the ***JCRMI security policy*** defined in JCRMI Policy.

S.PACKAGE

Packages used on the Java Card platform that act on behalf of the applet Developer. These subjects are involved in the ***FIREWALL security policy*** defined in [SPM] and they should be understood as instances of the subject *S.PACKAGE*.

S.JCRE

The *JCRE*, which acts on behalf of the card issuer. This subject is involved in several of the security policies defined in [PP/JCS] and is always represented by the subject *S.JCRE*.

S.BCV

The bytecode verifier (*BCV*), which acts on behalf of the verification authority. This subject is involved in the ***PACKAGE LOADING security policy*** defined in Security Functional Requirements and is represented by the subject *S.BCV*.

Application note:

The bytecode verifier is off-card.

S.CRD

The *installer*, which acts on behalf of the card issuer. This subject is involved in the loading of *packages* and installation of *applets*. It could play the role of the on-card entity in charge of package loading, which is involved in the ***PACKAGE LOADING security policy*** defined in CarG Security Functional Requirements and is represented by the subject *S.CRD*.

Application note:

The *installer* is on-card.

S.ADEL

The *applet deletion manager*, which also acts on behalf of the card issuer. This subject is involved in the ***ADEL security policy*** defined in applet Deletion Manager Policy.

S.SPY

A special subject is involved in the ***PACKAGE LOADING security policy***, which acts as the entity that may potentially intercept, modify, or permute the messages exchanged between the verification authority and the on-card entity in charge of package loading.

With the exception of *packages*, the other subjects have special privileges and play key roles in the security policies of the TOE.

3.3 Assumptions

The Assumptions of the TOE are based on those defined in [PP/JCS].

The assumptions described in [PP/BSI-0002] are studied in [ST/INFINEON].

3.3.1 Java Card

These Assumptions are those described in [PP/JCS] with the following exceptions:

- *A.NATIVE* is just removed because API belongs to the TOE and the assumptions described in *A.NATIVE* have been transferred to *OSP.NATIVE*.
- *A.DELETION* has been transformed into *OT.DELETION* as the ST includes *ADELG* group of SFRs additionally to the Standard 2.1.1 configuration. This SFR group appears with Standard 2.2 configuration of the JCS PP collection.

A.VERIFICATION

All the bytecodes are verified at least once, before the loading, in order to ensure that each bytecode is valid at execution time.

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM221], §3.3) outside the API.

3.4 Threats

The TOE as defined in chapter 2 is required to counter the threats described hereafter.

A threat agent wishes to abuse the assets either by functional attacks or by environmental manipulation, by specific hardware manipulation, by a combination of hardware and software manipulations or by any other type of attacks.

The threats of the TOE are those defined in [PP/JCS].

The threats of [PP/BSI-0002] are included in the evaluation but are studied in [ST/INFINEON].

3.4.1 Common

T.PHYSICAL

The attacker **discloses** or **modifies** the design of the TOE, its sensitive data or application code by physical (opposed to logical) tampering means.

This threat includes IC failure analysis, electrical probing, unexpected tearing, [Note: extension of the original text from the PP "and DP analysis"] fault injection, side channel attacks using Power and Electromagnetic analysis.

That also includes the modification of the runtime execution of *Java Card System* or *SCP* software through alteration of the intended execution order of (set of) instructions through physical tampering techniques.

3.4.2 Java Card

The threats are those of the [PP/JCS] to which have been added the following ones from the same PP, but 2.2 standard configuration:

- *T.DELETION*
- *T.OBJ-DELETION*

as the TOE is able to perform also application and object deletion additionally to Java Card 2.1.1 requirements (see SFRs from [PP/JCS]).

3.4.2.1 Confidentiality

T.CONFID-JCS-CODE

The attacker executes an application without authorisation to disclose the *Java Card System* code.

T.CONFID-APPLI-DATA

The attacker executes an application without authorisation to disclose data belonging to another application.

T.CONFID-JCS-DATA

The attacker executes an application without authorisation to disclose data belonging to the **Java Card System**.

3.4.2.2 Integrity

T.INTEG-APPLI-CODE

The attacker executes an application to alter (part of) its own or another application's code.

T.INTEG-JCS-CODE

The attacker executes an application to alter (part of) the *Java Card System* code.

T.INTEG-APPLI-DATA

The attacker executes an application to alter (part of) another application's data.

T.INTEG-JCS-DATA

The attacker executes an application to alter (part of) *Java Card System* or API data.

T.INTEG-APPLI-CODE.2

The attacker modifies (part of) its own or another application code when an application *package* is transmitted to the card for installation.

T.INTEG-APPLI-DATA.2

The attacker modifies (part of) the initialisation data contained in an application *package* when the package is transmitted to the card for installation.

3.4.2.3 Identity Usurpation

T.SID.1

An *applet* impersonates another application, or even the *JCRE*, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal.

T.SID.2

The attacker modifies the identity of the privileged roles.

3.4.2.4 Unauthorized Executions

T.EXE-CODE.1

An *applet* performs an unauthorized **execution** of a **method**.

T.EXE-CODE.2

An *applet* performs an unauthorized **execution** of a **method fragment** or **arbitrary data**.

T.NATIVE

An *applet* **executes** a **native method** to bypass a security function such as the *firewall*.

3.4.2.5 Denial of Service

T.RESOURCES

An attacker prevents correct operation of the *Java Card System* through consumption of some resources of the card: RAM or NVRAM.

3.4.2.6 Modifications of the Set of Applications

T.INSTALL

The attacker fraudulently **installs** post-issuance an *applet* on the card. This concerns either the installation of an unverified *applet* or an attempt to induce a malfunction in the TOE through the installation process.

3.4.2.7 Card Management

T.DELETION

The attacker **deletes** an *applet* or a *package* already in use on the card, or uses the deletion functions to pave the way for further attacks (putting the TOE in an insecure state).

3.4.2.8 Services

T.OBJ-DELETION

The attacker keeps a reference to a garbage collected object in order to force the TOE to execute an unavailable method, to make it to crash, or to gain access to a memory containing data that is now being used by another application.

3.5 Organisational security policies

The Organisational security policies described in [PP/BSI-0002] are studied in [ST/INFINEON].

3.5.1 Java Card

OSP.VERIFICATION

This policy shall ensure the adequacy between the export files used in the verification and those used for installing the verified file. The policy must also ensure that no modification of the file is performed in between its verification and the signing by the verification authority.

OSP.NATIVE

Those parts of the APIs written in native code as well as any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

Application note:

Note 1. This was an assumption in the [PP/JCS]. It is most likely to be seen as an OSP as the native applications are included in the TOE and evaluated with.

Note 2. *#.NATIVE* from [PP/JCS] is reduced as follows: No untrusted native code may reside on the card. Loading of native code is forbidden.

4 Security objectives

4.1 Security objectives for the TOE

The security objectives stated in [PP/BSI-0002] can be found in [ST/INFINEON] but are covered by this evaluation.

4.1.1 Java Card

The objectives are those of the standard 2.1.1 configuration to which are added *O.DELETION* and *O.OBJ-DELETION* from standard 2.2 configuration.

4.1.1.1 Identification

OT.SID

The TOE shall uniquely identify every subject (*applet*, or *package*) before granting him access to any [*JCS*] service.

4.1.1.2 Execution

OT.OPERATE

The TOE must ensure continued correct operation of its security functions.

OT.RESOURCES

The TOE controls the availability of resources for the applications.

Application note:

Here the resources include those of the *S.JCRE*.

OT.FIREWALL

The TOE shall ensure controlled sharing of data containers owned by *applets* of different *packages*, and between *applets* and the TSFs.

Application note:

By security domain here it is intended "execution space", which should not be confused with other meanings of "security domains".

OT.NATIVE

The only means that the *JCVM* shall provide for an application to execute native code is the invocation of a method of the Java Card API, or any additional API.

OT.REALLOCATION

The TOE shall ensure that the re-allocation of a memory block for the runtime areas of the *JCVM* does not disclose any information that was previously stored in that block.

Application note:

To be made unavailable means to be physically erased with a default value. Except for local variables that do not correspond to method parameters, the default values to be used are specified in [JCVM221].

OT.SHRD_VAR_CONFID

The TOE shall ensure that any data container that is shared by all applications is always cleaned after the execution of an application. Examples of such shared containers are the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API.

OT.SHRD_VAR_INTEG

The TOE shall ensure that only the currently selected application may grant write access to a data memory area that is shared by all applications, like the APDU buffer, the byte array used for the invocation of the process method of the selected applet, or any public global variable exported by the API. Even though the memory area is shared by all applications, the TOE shall restrict the possibility of getting a reference to such memory area to the application that has been selected for execution. The selected application may decide to temporarily hand over the reference to other applications at its own risk, but the TOE shall prevent those applications from storing the reference as part of their persistent states.

4.1.1.3 Services

OT.ALARM

The TOE shall provide appropriate feedback information upon detection of a potential security violation.

OT.TRANSACTION

The TOE must provide means to execute a set of operations atomically.

Application note:

1. Transactions are provided to *applets* as Java Card class libraries.
2. See also the *OT.KEY-MNGT* Application note.

OT.CIPHER

The TOE shall provide means to cipher sensitive data for applications in a secure way. In particular, the TOE must support cryptographic algorithms consistent with cryptographic usage policies and standards.

Application note:

See also the *OT.KEY-MNGT* Application note.

OT.PIN-MNGT

The TOE shall provide means to securely manage PIN objects.

Application note:

1. PIN objects may play key roles in the security architecture of client applications. The way they are stored and managed in the memory of the smart card must be carefully

considered, and this applies to the whole object rather than the sole value of the PIN. For instance, the try counter's value is as much sensitive as that of the PIN.

2. See also the *OT.KEY-MNGT* Application note.

OT.KEY-MNGT

The TOE shall provide means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys.

Application note:

OT.KEY-MNGT, OT.PIN-MNGT, OT.TRANSACTION and *OT.CIPHER* are actually provided to *applets* in the form of Java Card API's.

4.1.1.4 Applet Management

OT.INSTALL

The TOE shall ensure that the installation of an *applet* is safe.

OT.LOAD

The TOE shall ensure that the loading of a *package* into the card is safe.

Application note:

Usurpation of identity resulting from a malicious installation of an applet on the card may also be the result of perturbing the communication channel linking the *CAD* and the card. Even if the *CAD* is placed in a secure environment, the attacker may try to capture, duplicate, permute or modify the *packages* sent to the card. He may also try to send one of its own applications as if it came from the card issuer. Thus, this objective is intended to ensure the integrity and authenticity of loaded *CAP files*.

OT.DELETION

The TOE shall ensure that both *applet* and *package* deletion are safe.

4.1.1.5 Object Deletion

OT.OBJ-DELETION

The TOE shall ensure the object deletion shall not break references to objects.

4.1.1.6 IC and Native

These objectives were Environment objectives into [PP/JCS]. They become Objectives for the TOE because the TOE in this ST includes the SCP.

OE.NATIVE is renamed to *OT.NATIVE.2* in order to avoid a duplication of names.

OT.NATIVE.2

Those parts of the APIs written in native code as well as any pre-issuance native application on the card shall be conformant with the TOE so as to ensure that security policies and objectives described herein are not violated.

OT.SCP.RECOVERY

If there is a loss of power, or if the smart card is withdrawn from the *CAD* while an operation is in progress, the *SCP* must allow the TOE to eventually complete the interrupted operation successfully, or recover to a consistent and secure state (#.SCP.1 from [PP/JCS]).

OT.SCP.SUPPORT

The SCP shall provide functionalities that support the well-functioning of the TSFs of the TOE (avoiding they are bypassed or altered) and by controlling the access to information proper of the TSFs. In addition, the smart card platform should also provide basic services, which are required by the runtime environment to implement security mechanisms such as atomic transactions, management of persistent and transient objects and cryptographic functions. These mechanisms are likely to be used by security functions implementing the security requirements defined for the TOE.

OT.SCP.IC

The *SCP* shall possess IC security features.

Application note:

The TOE must be resistant to physical attack and prevent use of security relevant assets derived from these attacks.

OT.CARD-MANAGEMENT

The card manager shall control the access to card-management functions such as the installation, update or deletion of *applets*. It shall also implement the card issuer's policy on the card.

4.2 Security objectives for the environment

4.2.1 Common

OE.DEVELOPMENT

During phases 1 to 6, procedures shall be used suitably to maintain the integrity and confidentiality of the assets of the TOE.

4.2.2 Java Card

OE.APPLET

No *applet* loaded post-issuance contains native methods.

OE.VERIFICATION

Any byte-code must be verified prior to being loaded, in order to ensure that each bytecode is valid at execution time.

The card manager is an application with specific rights, which is responsible for the administration of the smart card. This component is part of the TOE, but outside Java Card,

and enforces some of its security functions. Typically the card manager is in charge of the life cycle of the whole card, as well as that of the installed applications (*applet*s). The card manager should prevent that card content management (loading, installation, deletion) is carried out, for instance, at invalid states of the card or by non-authorized actors. It shall also enforce security policies established by the card issuer.

5 IT security requirements

5.1 TOE security functional requirements

[ST/Infineon] deals with the security functional requirements of [PP/BSI-0002].

5.1.1 Common

5.1.1.1 Smart card platform (SCPG)

This category includes functional requirements that belonged previously to the IT environment into [PP/JCS]. The Smart Card Platform is now part of the TOE; therefore they are now TOE security functional requirements. The following modifications were done with respect to [PP/JCS]:

- *FPT_RCV.3/SCP* has been merged into *FPT_RCV.3/JCS*;
- *FPT_RVM.1/SCP* has integrated *FPT_RVM.1*.

FPT_AMT.1/SCP Abstract machine testing

FPT_AMT.1.1/SCP The TSF shall run a suite of tests **at each Power-on** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the TSF.

FPT_FLS.1/SCP Failure with preservation of secure state

FPT_FLS.1.1/SCP The TSF shall preserve a secure state when the following types of failures occur: **same as in FPT_FLS.1/JCS**.

FRU_FLT.1/SCP Degraded fault tolerance

FRU_FLT.1.1/SCP [Editorially Refined] The TSF shall ensure **all operations but package loading, applet installation and extension (need the creation of a new chained container)** when the following failures occur: **not enough memory left**.

FPT_PHP.3/SCP Resistance to physical attack

FPT_PHP.3.1/SCP [Editorially Refined] The TSF shall resist to the **physical attacks** on the **TOE** by responding automatically such that the TSP is not violated.

FPT_SEP.1/SCP TSF domain separation

FPT_SEP.1.1/SCP The TSF shall maintain a security domain for its own execution that protects it from interference and tampering by untrusted subjects.

FPT_SEP.1.2/SCP The TSF shall enforce separation between the security domains of subjects in the TSC.

Application note:

By security domain it is intended "execution context" which should not be confused with other meanings of "security domains".

FPT_RVM.1/SCP Non-bypassability of the TSP

FPT_RVM.1.1/SCP The TSF shall ensure that TSP enforcement functions are invoked and succeed before each function within the TSC is allowed to proceed.

Application note:

This component supports *OT.SCP.SUPPORT*, which in turn contributes to the secure operation of the TOE, by ensuring that these latter as well as the supporting platform security mechanisms cannot be bypassed.

FPT_RCV.4/SCP Function recovery

FPT_RCV.4.1/SCP The TSF shall ensure that **reading from and writing to static and objects' fields interrupted by power loss** have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

5.1.1.2 Card manager (CMGRG)

This category includes functional requirements that belonged previously to the IT environment into [PP/JCS]. The Card Manager is now part of the TOE; therefore they are now TOE security functional requirements. The following items are missing from the original [PP/JCS] group:

- *FMT_SMR.1/CMGR*; all Security Roles have been merged into *FMT_SMR.1/JCS*.
- *FIA_UID.1/CMGR*; the actions on behalf of the user have been merged into *FIA_UID.1/JCS*.

FDP_ACC.1/CMGR Subset access control

FDP_ACC.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** on **loading of code and keys by the Operator**.

FDP_ACF.1/CMGR Security attribute based access control

FDP_ACF.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to objects based on the following:

- **Subjects: Byte Code Verifier, Operator**
- **Objects: applets, keys and PINs.**
- **Security Attributes: DAP for applets; type and KEK for keys**.

FDP_ACF.1.2/CMGR The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed:

- **the Byte Code Verifier loads applets into the card**
- **the Operator deletes applets in the card**.

FDP_ACF.1.3/CMGR The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4/CMGR The TSF shall explicitly deny access of subjects to objects based on the **none**.

FMT_MSA.1/CMGR Management of security attributes

FMT_MSA.1.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to restrict the ability to **modify** the security attributes **code category** to **none**.

FMT_MSA.3/CMGR Static attribute initialisation

FMT_MSA.3.1/CMGR The TSF shall enforce the **CARD CONTENT MANAGEMENT access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/CMGR The TSF shall allow the **none** to specify alternative initial values to override the default values when an object or information is created.

5.1.2 Java Card

5.1.2.1 Additional needed SFR

This category contains a requirement that is not listed by [PP/JCS], but that is needed as a dependency for already included SFRs. PP/JCS] is no longer complete according to [CC-2].

FMT_SMF.1/CM Specification of management functions

FMT_SMF.1.1/CM The TSF shall be capable of performing the following security management functions:

- **the loading and the installing of the applets, with their DAP and AID by the Operator;**
- **the selection of applets**.

Application note:

This requirement satisfies the dependencies with the *FMT_MSA.1* requirements.

5.1.2.2 Merged SFRs

This group contains SFRs that have been merged from different requirements that [PP/JCS] originally placed into several groups.

FIA_UID.1/JCS Timing of identification

FIA_UID.1.1/JCS The TSF shall allow **JCAPI with already installed applets** on behalf of the user to be performed before the user is identified.

FIA_UID.1.2/JCS The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

Application note:

This SFR was built with *FIA_UID.1/CM* and *FIA_UID.1/CMGR*.

By users here it must be understood the ones associated to the *packages* (or *applets*), which act as subjects of policies. In the Java Card System, every action is always performed by an identified user interpreted here as the currently selected *applet* or the *package* that is the subject's owner. Means of identification are provided during the loading procedure of the *package* and the registration of *applet* instances.

FMT_SMR.1/JCS Security roles

FMT_SMR.1.1/JCS The TSF shall maintain the roles **JCRE, Operator, Card Manager**.

FMT_SMR.1.2/JCS The TSF shall be able to associate users with roles.

Application note:

1. This SFR combines *FMT_SMR.1* from [PP/JCS], *ADEL* from 2.2 Standard configuration from the same JCS PP collection and it excepts the *BCV* requirement:

- *FMT_SMR.1/JCRE*;
- *FMT_SMR.1/Installer*;
- *FMT_SMR.1/ADEL*;
- *FMT_SMR.1/CM*,
- *FMT_SMR.1/CMGR*.

2. The Operator includes the *CAD* and the Bytecode Verifier (*FMT_SMR.1/BCV*).

The Card Manager (*FMT_SMR.1/CMGR*) includes the Installer (*FMT_SMR.1/Installer*) and the Applet Deletion Manager (*FMT_SMR.1/ADEL*).

FPT_RCV.3/JCS Automated recovery without undue loss

FPT_RCV.3.1/JCS When automated recovery from **a failure** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.

FPT_RCV.3.2/JCS For **Applet loading, installation and deletion failure; Sensitive data loading failure**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.

FPT_RCV.3.3/JCS The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **any security attribute** for loss of TSF data or objects within the TSC.

FPT_RCV.3.4/JCS The TSF shall provide the capability to determine the objects that were or were not capable of being recovered.

Application note:

1. This SFR combines *FPT_RCV.3/SCP* and *FPT_RCV.3/Installer* from [PP/JCS].

2. The TOE has no maintenance mode. Therefore all operations must be interrupted.

3. The secure state is the mute state.

4. The actual recovery services are provided by the Operating System.

5.1.2.3 Core (CoreG)

This group includes the Security Functional Requirements as described by the PP, except *FMT_SMR.1/JCRE*. All Security Roles have been merged into *FMT_SMR.1/JCS*.

Firewall Policy

The following requirements are modified with respect to the PP:

- *FMT_MSA.1/JCRE*. *FPT_SEP.1* was integrated into *FPT_SEP.1/SCP*, as now the Smart Card platform is part of the TOE.

FDP_ACC.2/FIREWALL Complete access control

FDP_ACC.2.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** on **S.PACKAGE, S.JCRE, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

Non editorial refinement:

Subjects (prefixed with an " *S* ") and objects (prefixed with an " *O* ") covered by this policy are:

Subject, Object	**Description**
S.PACKAGE	Any *package*, which is the security unit of the firewall policy.
S.JCRE	The *JCRE*. This is the process that manages *applet* selection and de-selection, along with the delivery of *APDUs* from and to the smart card device. *This subject is unique.*
O.JAVAOBJECT	Any object. Note that KEYS, PIN, arrays and *applet* instances are specific objects in the Java programming language.

Operations (prefixed with " *OP* ") of this policy are described in the following table. Each operation has a specific number of parameters given between brackets, among which there is the " **accessed object** ", the first one, when applicable. Parameters may be seen as security attributes that are under the control of the subject performing the operation.

Operation	Description
OP.ARRAY_ACCESS (O.JAVAOBJECT, field*)*	Read/Write an array component.
OP.INSTANCE_FIELD (O.JAVAOBJECT, field*)*	Read/Write a field of an instance of a class in the Java programming language
OP.INVK_VIRTUAL (O.JAVAOBJECT, method, arg1,...*)*	Invoke a virtual method (either on a class instance or an array object)
OP.INVK_INTERFACE (O.JAVAOBJECT, method, arg1,...*)*	Invoke an *interface* method.
OP.THROW (O.JAVAOBJECT)	Throwing of an object (**athrow**).
OP.TYPE_ACCESS (O.JAVAOBJECT, class*)*	Invoke **checkcast** or **instanceof** on an object.
OP.JAVA (...)	Any access in the sense of [JCRE221], §6.2.8. In our formalization, this is one of the preceding operations.
*OP.CREATE (*Sharing, LifeTime*)*	Creation of an object (**new** or **makeTransient call**).

Note that accessing array's components of a **static** array, and more generally fields and methods of **static** objects, is an access to the corresponding *O.JAVAOBJECT*.

FDP_ACC.2.2/FIREWALL The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1/FIREWALL Security attribute based access control

FDP_ACF.1.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP** to objects based on the following: **(1) the security attributes of the covered subjects and objects, (2) the currently active context and (3) the SELECTed applet Context**.

Non editorial refinement:

The following table describes which security attributes are attached to which subject/object of our policy:

Subject /Object	**Attributes**
S.PACKAGE	Context
S.JCRE	None
O.JAVAOBJECT	Sharing, Context, LifeTime

The following table describes the possible values for each security attribute:

Name	Description
Context	*Package AID* or "JCRE"
Sharing	Standard, SIO, JCRE Entry Point, or global array
LifeTime	**CLEAR_ON_DESELECT** or **PERSISTENT**.
SELECTed applet Context	*Package AID* or "None"

In the case of an array type, we state that fields are components of the array ([JVM], §2.14, §2.7.7), as well as the length; the only methods of an array object are those inherited from the **Object** *class*.

The Sharing attribute defines four categories of objects:

- Standard ones, whose both fields and methods are under the firewall policy,
- Shareable interface Objects (SIO), which provide a secure mechanism for inter-applet communication,
- *JCRE entry points* (Temporary or Permanent), who have freely accessible methods but protected fields,
- Global arrays, having both unprotected fields (including components; refer to JavaCardClass discussion above) and methods.

When a new object is created, it is associated with the currently active context. But the object is owned by the *applet* instance within the currently active context when the object is instantiated ([JCRE221], §6.1.2). An object is owned by an *applet* instance, by the *JCRE* or by the *package* library where it has been defined (these latter objects can only be arrays that initialize static fields of packages).

Finally both "the currently active context" and "the SELECTed applet context" are security attributes internal to the VM, that is, not attached to any specific object or subject of the Security Policy Model ("SPM"). They are TSF data that play a role in the SPM.

> ([JCRE221], Glossary) *Currently selected applet*. The JCRE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's *AID*, the JCRE makes this applet the currently selected applet. The JCRE sends all APDU commands to the currently selected applet.

While the expression "selected applet" refers to a specific installed applet, the relevant aspect to the policy is the *context* of the selected *applet*; that is why the associated security attribute is a package *AID*.

> ([JCRE221] §6.1.1) At any point in time, there is only **one active context** within the VM (this is called the *currently active context*).

This should be identified in our model with the acting *S.PACKAGE*'s context (see " *Current context* " in the glossary). This value is in one-to-one correspondence with *AIDs* of *package*s (except for the JCRE context, of course), which appears in the model in the "Context" attribute of both subjects and objects of the policy. The reader should note that the invocation of static methods (or access to a static field) is not considered by this policy, as there are no firewall rules. They have no effect on the active context as well and the "acting package" is not the one to which the static method belongs in this case.

FDP_ACF.1.2/FIREWALL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **by the FIREWALL SFP:**

R.JAVA.1 ([JCRE221] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*, *OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE*, *OP.THROW* or *OP.TYPE_ACCESS* upon any *O.JAVAOBJECT* whose Sharing attribute has value "JCRE Entry Point" or "Global Array".

R.JAVA.2 ([JCRE221] §6.2.8) An *S.PACKAGE* may freely perform any of *OP.ARRAY_ACCESS*, *OP.INSTANCE_FIELD*, *OP.INVK_VIRTUAL*, *OP.INVK_INTERFACE* or *OP.THROW* upon any *O.JAVAOBJECT* whose Sharing attribute has value "Standard" and whose Lifetime attribute has value "PERSISTENT" only if *O.JAVAOBJECT* 's Context attribute has the same value as the active context.

R.JAVA.3 ([JCRE221] §6.2.8.10) An *S.PACKAGE* may perform *OP.TYPE_ACCESS* upon an *O.JAVAOBJECT* whose Sharing attribute has value "SIO" only if *O.JAVAOBJECT* is being cast into (checkcast) or is being verified as being an instance of (instanceof) an *interface* that extends the Shareable *interface*.

R.JAVA.4 ([JCRE221]§6.2.8.6) An *S.PACKAGE* may perform *OP.INVK_INTERFACE* upon an *O.JAVAOBJECT* whose Sharing attribute has the value "SIO" only if the invoked *interface* method extends the Shareable *interface*.

R.JAVA.5 An *S.PACKAGE* may perform an *OP.CREATE* only if the value of the Sharing parameter is "Standard".

At last, rules governing access to and creation of *O.JAVAOBJECT* s by *S.JCRE* are essentially implementation-dependent (however, see *FDP_ACF.1.3/FIREWALL*).

FDP_ACF.1.3/FIREWALL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules:

The subject *S.JCRE* can freely perform *OP.JAVA(...)* and *OP.CREATE*, with the exception given in *FDP_ACF.1.4/FIREWALL*.

FDP_ACF.1.4/FIREWALL The TSF shall explicitly deny access of subjects to objects based on the **rules:**

1) Any subject with *OP.JAVA* upon an *O.JAVAOBJECT* whose LifeTime attribute has value "CLEAR_ON_DESELECT" if *O.JAVAOBJECT* 's Context attribute is not the same as the SELECTed applet Context.

2) Any subject with *OP.CREATE* and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the SELECTed applet Context.

Application note:

The deletion of *applet*s may render some *O.JAVAOBJECT* inaccessible, and the JCRE may be in charge of this aspect. This can be done, for instance, by ensuring that references to objects belonging to a deleted application are considered as a null reference. Such a mechanism is implementation-dependent.

FDP_IFC.1/JCVM Subset information flow control

FDP_IFC.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** on **the following subjects, information and operations**.

Non editorial refinement:

Subjects (prefixed with an " *S* ") and information (prefixed with an " *I* ") covered by this policy are:

Subject/Information	**Description**
S.LOCAL	Operand stack of a JCVM frame, or local variable of a JCVM frame containing an object or an array of references.
S.MEMBER	Any object's field, static field or array position.
I.DATA	JCVM Reference Data: *objectref addresses of temporary JCRE Entry Point objects and global arrays.*

There is a unique operation in this policy:

Operation	**Description**
OP.PUT(S_1, S_2, I)	Transfer a piece of information *I* from S_1 to S_2.

Application note:

References of temporary *JCRE entry points*, which cannot be stored in *class* variables, instance variables or array components, are transferred from the internal memory of the *JCRE* (TSF data) to some stack through specific APIs (*JCRE* owned exceptions) or *JCRE* invoked methods (such as the **process(APDU apdu)**); these are causes of *OP.PUT(S_1,S_2,I)* operations as well.

FDP_IFF.1/JCVM Simple security attributes

FDP_IFF.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** based on the following types of subject and information security attributes: **S.LOCAL, S.MEMBER, I.DATA and the currently active Context**.

FDP_IFF.1.2/JCVM The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: **An operation *OP.PUT(S_1, S.MEMBER, I)* is allowed if and only if the active context is "JCRE"; other *OP.PUT* operations are allowed regardless of the active context's value**.

FDP_IFF.1.3/JCVM The TSF shall enforce the **none**.

FDP_IFF.1.4/JCVM The TSF shall provide the following **none**.

FDP_IFF.1.5/JCVM The TSF shall explicitly authorise an information flow based on the following rules: **all JCRE Permanent Entry Point Object may be stored in a S.MEMBER**.

FDP_IFF.1.6/JCVM The TSF shall explicitly deny an information flow based on the following rules: **the storage of the reference of an object with attribute JCRE Temporary Entry Point Object or Global Array in a static field, instance field or array element is forbidden**.

Application note:

The storage of temporary *JCRE* -owned objects' references is runtime-enforced ([JCRE221], §6.2.8.1-3).

Note that this policy essentially applies to the execution of bytecode. Native methods, the JCRE itself and possibly some API methods can be granted specific rights or limitations through the *FDP_IFF.1.3/JCVM* to *FDP_IFF.1.6/JCVM* elements. The way the virtual machine manages the transfer of values on the stack and local variables (returned values, uncaught exceptions) from and to internal registers is implementation-dependent. For instance, a returned reference, depending on the implementation of the stack frame, may transit trough an internal register prior to being pushed on the stack of the invoker. The **areturn** bytecode would cause more than one *OP.PUT* operation under this scheme.

FDP_RIP.1/OBJECTS Subset residual information protection

FDP_RIP.1.1/OBJECTS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **class instances and arrays**.

Application note:

The semantics of the Java programming language requires for any object field and array position to be initialised with default values when the resource is allocated [JVM], §2.5.1.

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to restrict the ability to **modify** the security attributes **the active context and the SELECTed applet Context** to **the JCRE (*S.JCRE*)**.

Application note:

The modification of the active context, SELECTed applet Context and ActiveApplets security attributes should be performed in accordance with the rules given in [JCRE221], §4 **and ([JCVM221], §3.4**.

FMT_MSA.2/JCRE Secure security attributes

FMT_MSA.2.1/JCRE The TSF shall ensure that only secure values are accepted for security attributes.

Application note:

Secure values conform to the following rules:

- The Context attribute of a **.JAVAOBJECT* must correspond to that of an installed applet or be "JCRE".
- An *O.JAVAOBJECT* whose Sharing attribute is *JCRE Entry Point* or Global Array necessarily has "JCRE" value for its Context security attribute.
- An *O.JAVAOBJECT* whose Sharing attribute value is Global Array necessarily have "array of primitive Java Card type" as JavaCardClass security attribute's value.
- Any *O.JAVAOBJECT* whose Sharing attribute value is not "Standard" has a PERSISTENT-LifeTime attribute's value.
- Any *O.JAVAOBJECT* whose LifeTime attribute value is not PERSISTENT has an array type as JavaCardClass attribute's value.

FMT_MSA.3/FIREWALL Static attribute initialisation

FMT_MSA.3.1/FIREWALL The TSF shall enforce the **FIREWALL access control SFP and the JCVM information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/FIREWALL The TSF shall allow the **authorized identified roles: None** to specify alternative initial values to override the default values when an object or information is created.

Application Programming Interface

Application note for all FCS_COP.1 SFRs

The TOE shall provide a subset of cryptographic operations defined in [JCAPI221] in accordance with related JC API specifications (see javacardx.crypto.Cipher and javacard.security packages).

FCS_CKM.1/TDES Cryptographic key generation

FCS_CKM.1.1/TDES The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **TDES Key generation** and specified cryptographic key sizes **112 bits for TDES 2 keys, 168 bits for TDES 3 keys** that meet the following: **none (random numbers generation)**.

FCS_CKM.1/RSA Cryptographic key generation

FCS_CKM.1.1/RSA The TSF shall generate cryptographic keys in accordance with a specified cryptographic key generation algorithm **RSA Key generation** and specified cryptographic key sizes **1536, 1792, 2048 bits** that meet the following: **none (generation of random numbers and Miller- Rabin primality testing)**.

FCS_CKM.2/TDES Cryptographic key distribution

FCS_CKM.2.1/TDES The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **JC API getkey()** that meets the following: **none**.

FCS_CKM.2/RSA Cryptographic key distribution

FCS_CKM.2.1/RSA The TSF shall distribute cryptographic keys in accordance with a specified cryptographic key distribution method **JC API getkey()** that meets the following: **none**.

FCS_CKM.3/TDES Cryptographic key access

FCS_CKM.3.1/TDES The TSF shall perform **access to TDES keys** in accordance with a specified cryptographic key access method **Secure reading in Memory** that meets the following: **none**.

FCS_CKM.3/RSA Cryptographic key access

FCS_CKM.3.1/RSA The TSF shall perform **access to RSA keys** in accordance with a specified cryptographic key access method **Secure reading in Memory** that meets the following: **none**.

FCS_CKM.4/TDES Cryptographic key destruction

FCS_CKM.4.1/TDES The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method**: physical irreversible destruction of the stored key value** that meets the following: **no standard**.

FCS_CKM.4/RSA Cryptographic key destruction

FCS_CKM.4.1/RSA The TSF shall destroy cryptographic keys in accordance with a specified cryptographic key destruction method**: physical irreversible destruction of the stored key value** that meets the following: **no standard**.

FCS_COP.1/TDES Cryptographic operation

FCS_COP.1.1/TDES The TSF shall perform **TDES encryption and decryption** in accordance with a specified cryptographic algorithm **TDES-CBC, TDES-EBC** and cryptographic key sizes **112 bits for TDES 2 keys, 168 bits for TDES 3 keys** that meet the following: **[FIPS 46-3]**.

Application note:

The TOE can also encrypt and decrypt using DES algorithm with 56 bits key, but this is to be considered as a service. The DES algorithm is no longer considered as resistant to high level attacks. The applet developer must take into account this consideration.

FCS_COP.1/RSA Cryptographic operation

FCS_COP.1.1/RSA The TSF shall perform **RSA encryption and decryption** in accordance with a specified cryptographic algorithm **RSA** and cryptographic key sizes **1536, 1792 and 2048 bits** that meet the following: **[RSA PKCS#1]**.

Application note:

The TOE can also encrypt and decrypt using DES algorithm with 56 bits key, but this is to be considered as a service. The DES algorithm is no longer considered as resistant to high level attacks. The applet developer must take into account this consideration.

FCS_COP.1/Signature Cryptographic operation

FCS_COP.1.1/Signature The TSF shall perform **Signature and verification of signature** in accordance with a specified cryptographic algorithm **RSA_SHA_PKCS#1, RSA_SHA_ISO9796_2** and cryptographic key sizes **1536, 1792 and 2048 bits** that meet the following: **[RSA PKCS#1], [ISO 9796_2]**.

FCS_COP.1/SHA-1 Cryptographic operation

FCS_COP.1.1/SHA-1 The TSF shall perform **secure hashing** in accordance with a specified cryptographic algorithm **SHA-1** and cryptographic key sizes **none** that meet the following: **[FIPS 180-2]**.

Global refinement:

This cryptographic algorithm does not use a key.

FDP_RIP.1/ABORT Subset residual information protection

FDP_RIP.1.1/ABORT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any reference to an object instance created during an aborted transaction**.

Global refinement:

The resource is *any reference to an object instance created during an aborted transaction*.

FDP_RIP.1/APDU Subset residual information protection

FDP_RIP.1.1/APDU The TSF shall ensure that any previous information content of a resource is made unavailable upon the **allocation of the resource to** the following objects: **the APDU buffer**.

FDP_RIP.1/bArray Subset residual information protection

FDP_RIP.1.1/bArray The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the bArray object**.

FDP_RIP.1/KEYS Subset residual information protection

FDP_RIP.1.1/KEYS The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the cryptographic buffer (D.CRYPTO)**.

FDP_ROL.1/FIREWALL Basic rollback

FDP_ROL.1.1/FIREWALL The TSF shall enforce **the FIREWALL access control SFP and the JCVM information flow control SFP** to permit the rollback of the **OP.JAVA, OP.CREATE** on the **O.JAVAOBJECTs**.

FDP_ROL.1.2/FIREWALL The TSF shall permit operations to be rolled back within the **scope of a select(), deselect(), process() or install() call, notwithstanding the restrictions given in [JCRE221], §7.7, within the bounds of the Commit Capacity ([JCRE221], §7.8), and those described in [JCAPI221]**.

FDP_RIP.1/TRANSIENT Subset residual information protection

FDP_RIP.1.1/TRANSIENT The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **any transient object**.

Application note:

The events that provoke the de-allocation of a transient object are described in [JCRE221], §5.1.

Card Security Management

This category contains the all SFRs from [PP/JCS], but *FPT_RVM.1* that was merged within the corresponding requirement from the **SCPG**.

FAU_ARP.1/JCS Security alarms

FAU_ARP.1.1/JCS The TSF shall take **the following actions: throw an exception, lock the card session or reinitialise the JCS and its data** upon detection of a potential security violation.

Global refinement:

Potential security violation is refined to one of the following events:

- Applet life cycle inconsistency
- Card tearing (unexpected removal of the Card out of the *CAD*) and power failure
- Abortion of a transaction in an unexpected context (see (abortTransaction(), [JCAPI221] and ([JCRE221], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow
- Other runtime errors related to applet's failure (like uncaught exceptions)

Application note:

The thrown exceptions and their related events are described in [JCRE221], [JCAPI221] and [JCVM221].

FDP_SDI.2 Stored data integrity monitoring and action

FDP_SDI.2.1 The TSF shall monitor user data stored within the TSC for **integrity errors** on all objects, based on the following attributes: **JCS integrity checked stored data**.

Non editorial refinement:

The following data persistently stored by TOE have the user data attribute "integrity checked stored data":

1. PINs
2. keys
3. application sensitive data
4. applet code.

FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall **warn the connected entity**.

FPT_TDC.1 Inter-TSF basic TSF data consistency

FPT_TDC.1.1 The TSF shall provide the capability to consistently interpret **the CAP files (shared between the Byte Code Verifier and the TOE), the bytecode and its data arguments (shared with applets and API packages),** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use **the following rules:**

- **The [JCVM221] specification;**
- **Reference export files;**
- **The ISO 7816-6 rules;**
- **The [GP] specification**

when interpreting the TSF data from another trusted IT product.

Application note:

This TOE includes the Card Manager. So this SFR is modified and the interpretation consistency is required between the BCV and the TOE instead of the CM and the TOE.

FPT_FLS.1/JCS Failure with preservation of secure state

FPT_FLS.1.1/JCS The TSF shall preserve a secure state when the following types of failures occur: **those associated to the potential security violations described in FAU_ARP.1/JCS**.

FPR_UNO.1 Unobservability

FPR_UNO.1.1 The TSF shall ensure that **S.SPY** are unable to observe the operation **cryptographic operations / comparisons operations** on **Key values / PIN values** by **S.JCRE, S.Applet**.

FPT_TST.1 TSF testing

FPT_TST.1.1 The TSF shall run a suite of self tests **during initial start-up** to demonstrate the correct operation of **the TSF**.

FPT_TST.1.2 The TSF shall provide authorised users with the capability to verify the integrity of **the TSF data**.

FPT_TST.1.3 The TSF shall provide authorised users with the capability to verify the integrity of stored TSF executable code.

Application note:

"Initial start-up" means at each power on.

AID Management

FMT_MTD.1/JCRE Management of TSF data

FMT_MTD.1.1/JCRE The TSF shall restrict the ability to **modify** the **list of registered applets' *AID*** to **the JCRE**.

FMT_MTD.3 Secure TSF data

FMT_MTD.3.1 The TSF shall ensure that only secure values are accepted for TSF data.

FIA_ATD.1/AID User attribute definition

FIA_ATD.1.1/AID The TSF shall maintain the following list of security attributes belonging to individual users: **the *AID* and version number of each *package*, the *AID* of each registered *applet*, and whether a registered *applet* is currently selected for execution ([JCVM221], §6.5)**.

FIA_UID.2/AID User identification before any action

FIA_UID.2.1/AID The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

Application note:

The User here mentioned are those associated to Packages or Applets.

FIA_USB.1 User-subject binding

FIA_USB.1.1 The TSF shall associate the following user security attributes with subjects acting on behalf of the user: **the Context of the package**.

FIA_USB.1.2 The TSF shall enforce the following rules on the initial association of user security attributes with subjects acting on the behalf of users: **each *S.Package* has a different context independent of the one of the JCRE**.

FIA_USB.1.3 The TSF shall enforce the following rules governing changes to the user security attributes associated with subjects acting on the behalf of users: **none**.

Application note:

For *S.PACKAGE* s, the Context security attribute plays the role of the appropriate user security attribute; see *FMT_MSA.1.1/JCRE*.

5.1.2.4 Installer (InstG)

This group includes the Security Functional Requirements as described by the [PP/JCS], except:

- *FMT_SMR.1/Installer*, as all Security Roles have been merged into *FMT_SMR.1/JCS*;
- *FPT_RCV.3/Installer* as Automated Recovery without Undue Loss is usually provided by the Operating System, that is no longer located outside the current TOE. Please see *FPT_RCV.3/JCS*.

FDP_ITC.2/Installer Import of user data with security attributes

FDP_ITC.2.1/Installer The TSF shall enforce the **FIREWALL access control SFP** when importing user data, controlled under the SFP, from outside of the TSC.

FDP_ITC.2.2/Installer The TSF shall use the security attributes associated with the imported user data.

FDP_ITC.2.3/Installer The TSF shall ensure that the protocol used provides for the unambiguous association between the security attributes and the user data received.

FDP_ITC.2.4/Installer The TSF shall ensure that interpretation of the security attributes of the imported user data is as intended by the source of the user data.

FDP_ITC.2.5/Installer The TSF shall enforce the following rules when importing user data controlled under the SFP from outside the TSC:

A package may depend on (import or use data from) other packages already installed. This dependency is explicitly stated in the loaded package in the form of a list of package AIDs. The loading is allowed only if, for each dependent package, its AID attribute is equal to a resident package AID attribute, the major (minor) Version attribute associated to the former is equal (less than or equal) to the major (minor) Version attribute associated to the latter ([JCVM221], §4.5.2).

The intent of this rule is to ensure the binary compatibility of the package with those already on the card ([JCVM221], §4.4).

FPT_FLS.1/Installer Failure with preservation of secure state

FPT_FLS.1.1/Installer The TSF shall preserve a secure state when the following types of failures occur: **the installer fails to load/install a *package* / *applet* as described in [JCRE221] §11.1.5**.

FRU_RSA.1/Installer Maximum quotas

FRU_RSA.1.1/Installer The TSF shall enforce maximum quotas of the following resources: **imported packages and declared classes, methods and fields** that **subjects** can use **simultaneously**.

Global refinement:

Subjects here are the packages.

Application note:

1. A *package* may import at most 128 packages.

2. A *package* may declare at most 255 classes and interfaces.

3. A *class* can implement a maximum of 128 public or protected instance methods, and a maximum of 128 instance methods with *package* visibility. These limits include inherited methods.

4. A *class* instance can contain a maximum of 255 fields, where an int data type is counted as occupying two fields ([JCVM221], §2.2.4.2).

5.1.2.5 Applet deletion (ADELG)

This group was added with respect to the Standard 2.1.1 configuration [PP/JCS]. It was extracted from the Standard 2.2 configuration of the same JCS PP collection.

It includes the Security Functional Requirements as described by the PP, except *FMT_SMR.1/ADEL*. All Security Roles have been merged into *FMT_SMR.1/JCS*.

Applet Deletion Manager Policy

FDP_ACC.2/ADEL Complete access control

FDP_ACC.2.1/ADEL The TSF shall enforce the **ADEL access control SFP** on **S.ADEL, O.JAVAOBJECT, O.APPLET and O.CODE_PKG** and all operations among subjects and objects covered by the SFP.

Non editorial refinement:

Subjects (prefixed with an " *S* ") and objects (prefixed with an " *O* ") covered by this policy are:

S.ADEL The *applet deletion manager*. *This subject is unique.*

O.CODE_PKG The code of a *package*, including all linking information. On the Java Card platform, a package is the installation unit.

O.APPLET Any installed *applet*, its code and data.

O.JAVAOBJECT Java class instance or array.

Operations (prefixed with " *OP* ") of this policy are described in the following table:

Operation	Description
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed *applet* and its objects, either logically or physically.
OP.DELETE_PCKG(O.CODE_PKG,...)	Delete a *package*, either logically or physically
OP.DELETE_PCKG_APPLET(O.CODE_PKG,...)	Delete a *package* and its installed *applets*, either logically or physically.

FDP_ACC.2.2/ADEL The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

FDP_ACF.1/ADEL Security attribute based access control

FDP_ACF.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to objects based on the following: **(1) the security attributes of the covered subjects and objects, (2) the list of AIDs of the applet instances registered on the card, (3) the attribute ResidentPackages, which journals the list of AIDs of the packages already loaded on the card and (4) the attribute ActiveApplets, which is a list of the active applets' AIDs**.

Non editorial refinement:

The following table presents the security attributes associated to the subjects/objects under control of the policy:

Subject/Object	Attributes
O.CODE_PKG	*package's AID*, dependent packages' *AID* s, Static References
O.APPLET	Selection state
O.JAVAOBJECT	Owner

The package's *AID* identifies the package defined in the CAP file.

When an export file is used during preparation of a CAP file, the version numbers and *AIDs* indicated in the export file are recorded in the CAP files ([JCVM221], §4.5.2): the dependent packages AIDs attribute allows the retrieval of those identifications.

Static fields of a package may contain references to objects. The Static References attribute records those references.

An applet instance can be in two different selection states: selected or deselected. If the applet is selected (in some logical channel), then in turn it could either be *currently selected* or just *active*. At any time there could be up to four active applet instances, but only one currently selected. This latter is the one that is processing the current command ([JCRE221], §4).

The Owner of an object is either the applet instance that created the object or the package (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the package).

Finally, there are needed security attributes that are not attached to any object or subject of the TSP: (1) the ResidentPackages Versions (or Resident Image, [JCVM221],§4.5) and

AID s. They describe the packages that are already on the card, (2) the list of registered applet instances and (3) the ActiveApplets security attribute. They are all attributes internal to the VM, that is, not attached to any specific object or subject of the SPM. These attributes are TSF data that play a role in the SPM.

FDP_ACF.1.2/ADEL The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: **by the ADEL SFP: see corresponding refinement**.

Non editorial refinement:

The subject of this policy is *S.ADEL*.

Some basic common specifications are required in order to allow Java Card *applet*s and *package*s to be deleted without knowing the implementation details of a particular deletion manager. In particular, this policy introduces a notion of **reachability**, which provides a general means to describe objects that are referenced from a certain *applet* instance or *package*.

In the context of this policy, an object O is reachable if and only if either: (1) the owner of O is a registered *applet* instance A (O is reachable from A), (2) a static field of a loaded *package* P contains a reference to O (O is reachable from P), (3) (option not applicable, remote reachable objects not supported), and (4) there is an object O' that is reachable according to either (1) or (2) or (3) above and O' contains a reference to O (the reachability status of O is that of O').

The following access control rules determine when an operation among controlled subjects and objects is allowed by the policy:

R.JAVA.14 ([JCRE221], §11.3.4.1, **Applet Instance Deletion**).Deletion). The *S.ADEL* may perform *OP.DELETE_APPLET* upon an *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) *O.APPLET* is deselected and (3) there is no *O.JAVAOBJECT* owned by *O.APPLET* such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from O.APPLET, or *O.JAVAOBJECT* is reachable from a package P.

R.JAVA.15 ([JCRE221],§11.3.4.1, **Multiple Applet Instance Deletion**). The *S.ADEL* may perform *OP.DELETE_APPLET* upon several *O.APPLET* only if, (1) *S.ADEL* is currently selected, (2) every *O.APPLET* being deleted is deselected and (3) there is no *O.JAVAOBJECT* owned by any of the *O.APPLET* being deleted such that either *O.JAVAOBJECT* is reachable from an applet instance distinct from any of those *O.APPLET*, or *O.JAVAOBJECT* is reachable from a package P.

R.JAVA.16 ([JCRE221], §11.3.4.2, **Applet/Library Package Deletion**). The *S.ADEL* may perform *OP.DELETE_PCKG* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG* that is an instance of a class that belongs to *O.CODE_PCKG* exists on the card and (3) there is no package loaded on the card that depends on *O.CODE_PCKG*.

R.JAVA.17 ([JCRE221], §11.3.4.3, **Applet Package and Contained Instances Deletion**). The *S.ADEL* may perform *OP.DELETE_PCKG_APPLET* upon an *O.CODE_PCKG* only if, (1) *S.ADEL* is currently selected, (2) no reachable *O.JAVAOBJECT*, from a package distinct from *O.CODE_PCKG*, which is an instance of a class that belongs to *O.CODE_PCKG* exists on the card, (3) there is no package loaded on the card that depends on *O.CODE_PCKG* and (4) for every *O.APPLET* of those being deleted it holds that: (i) *O.APPLET* is deselected and (ii) there is no *O.JAVAOBJECT* owned by *O.APPLET*

such that either *O.JAVAOBJECT* is reachable from an applet instance not being deleted, or *O.JAVAOBJECT* is reachable from a package not being deleted.

FDP_ACF.1.3/ADEL The TSF shall explicitly authorise access of subjects to objects based on the following additional rules: **none**.

FDP_ACF.1.4/ADEL The TSF shall explicitly deny access of subjects to objects based on the**: any subject but the *S.ADEL* to *O.CODE_PKG* or *O.APPLET* must not have access for the purpose of deleting it from the card**.

Application note:

As the product doesn't support RMI, references to Remote objects have been removed from *FDP_ACF.1.1* and *FDP_ACF.1.2*.

FMT_MSA.1/ADEL Management of security attributes

FMT_MSA.1.1/ADEL The TSF shall enforce the **ADEL access control SFP** to restrict the ability to **modify** the security attributes **ActiveApplets** to **the JCRE**.

FMT_MSA.3/ADEL Static attribute initialisation

FMT_MSA.3.1/ADEL The TSF shall enforce the **ADEL access control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/ADEL The TSF shall allow the **following roles: none** to specify alternative initial values to override the default values when an object or information is created.

Additional Deletion Requirements

FDP_RIP.1/ADEL Subset residual information protection

FDP_RIP.1.1/ADEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **applet instances and/or packages when one of the deletion operations in FDP_ACC.2.1/ADEL is performed on them**.

Application note:

Requirements on de-allocation during *applet* / *package* deletion are described in [JCRE221] §11.3.4.1, §11.3.4.2 and §11.3.4.3.

FPT_FLS.1/ADEL Failure with preservation of secure state

FPT_FLS.1.1/ADEL The TSF shall preserve a secure state when the following types of failures occur: **the *applet deletion manager* fails to delete a *package* / *applet* as described in [JCRE221], §11.3.4**.

5.1.2.6 Object deletion (ODELG)

This group was added additionally to Standard 2.1.1 configuration [PP/JCS]. It was extracted from the Standard 2.2 configuration of the same JCS PP collection.

FDP_RIP.1/ODEL Subset residual information protection

FDP_RIP.1.1/ODEL The TSF shall ensure that any previous information content of a resource is made unavailable upon the **deallocation of the resource from** the following objects: **the objects owned by the context of an applet instance which triggered the execution of the method javacard.framework.JCSystem.requestObjectDeletion()**.

Application note:

Freed data resources resulting from the invocation of the method javacard.framework.JCSystem.requestObjectDeletion() may be reused. Requirements on *de-allocation* after the invocation of the method are described in [JCAPI221].

FPT_FLS.1/ODEL Failure with preservation of secure state

FPT_FLS.1.1/ODEL The TSF shall preserve a secure state when the following types of failures occur: **the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method**.

5.1.2.7 Secure carrier (CarG)

This group includes the Security Functional Requirements as described by the PP, except:

- *FMT_SMR.1/CM*. All Security Roles have been merged into *FMT_SMR.1/JCS*.
- *FIA_UID.1/CM*; the actions on behalf of the user have been merged into *FIA_UID.1/JCS*.

FCO_NRO.2/CM Enforced proof of origin

FCO_NRO.2.1/CM The TSF shall enforce the generation of evidence of origin for transmitted **application packages** at all times.

FCO_NRO.2.2/CM The TSF shall be able to relate the **identity** of the originator of the information, and the **application package** of the information to which the evidence applies.

FCO_NRO.2.3/CM [Editorially Refined] The TSF shall provide a capability to verify the evidence of origin of information to the **recipient** given **that the related cryptographic keys have been loaded using a secure process**.

Non editorial refinement:

The related cryptographic keys are usually TDES keys that serve to load *applets* / *packages* through a secure channel.

Application note:

If a new application package is received by the card for installation, the card manager shall first check that it actually comes from the verification authority. The verification authority is the entity responsible for bytecode verification (*S.BCV*).

If there are library packages, they are considered to be included into application packages.

FDP_IFC.2/CM Complete information flow control

FDP_IFC.2.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** on **S.CRD, S.BCV, S.SPY** and all operations that cause that information to flow to and from subjects covered by the SFP.

Non editorial refinement:

Subjects (prefixed with an " *S* ") covered by this policy are those involved in the reception of an application package by the card through a potentially unsafe communication channel:

Subject	Description
S.BCV	The subject representing who is in charge of the bytecode verification of the packages (also known as the verification authority).
S.CRD	The on-card entity in charge of package downloading.
S.SPY	Any other subject that may potentially intercept, modify, or permute the messages exchanged between the former two subjects.

The operations (prefixed with *"OP"*) that make information to flow between the subjects are those enabling to send a message through and to receive a message from the communication channel linking the card to the outside world. It is assumed that any message sent through the channel as clear text can be read by the attacker. Moreover,

the attacker may capture any message sent through the communication channel and send its own messages to the other subjects.

Operation	Description
OP.SEND(M)	A subject sends a message M through the communication channel.
OP.RECEIVE(M)	A subject receives a message M from the communication channel.

The information (prefixed with an *"I"*) controlled by the typing policy is the APDUs exchanged by the subjects through the communication channel linking the card and the *CAD*. Each of those messages contain part of an application package that is required to be loaded on the card (either S.BCV or S.SPY), as well as any control information used by the subjects in the communication protocol.

Information	Description
I.APDU	Any APDU sent to or from the card through the communication channel.

FDP_IFC.2.2/CM The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

FDP_IFF.1/CM Simple security attributes

FDP_IFF.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** based on the following types of subject and information security attributes: **S.BCV, S.CRD, S.SPY and I.APDU:**

Subject / Information	Attribute	Value
S.BCV	**DAPKey, OPKeys**	**Valid**
S.CRD	**DAPKey, OPKeys**	**Valid**
S.SPY	**none**	**none**
I.APDU	**SecureLevel**	**None, MAC, ENC**

Subject / Object	Attribute	Value
user	**role**	**Administrator (Operator, Signatory), None**
applet	**checked**	**Boolean**
DAP Key	**OK**	**Boolean**

.

FDP_IFF.1.2/CM The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold:

- **The user with the security attribute role set to Administrator can load an applet.**

- **Only applets with the security attribute Checked set to YES can be transferred (loaded).**
- **The DAP key OK security attribute must be set to TRUE to check the integrity and the origin of the applet**.

FDP_IFF.1.3/CM The TSF shall enforce the **none**.

FDP_IFF.1.4/CM The TSF shall provide the following **none**.

FDP_IFF.1.5/CM The TSF shall explicitly authorise an information flow based on the following rules:

- **The user with the security attribute role set to Administrator can load an applet.**
- **The DAP key OK security attribute must be set to TRUE to check the integrity and the origin of the applet**.

FDP_IFF.1.6/CM The TSF shall explicitly deny an information flow based on the following rules: **No user can load an applet with the security attribute Checked set to NO**.

Global refinement:

S.SPY A subject without the correct DAP attributes cannot modify the transmitted information.

FDP_UIT.1/CM Data exchange integrity

FDP_UIT.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to be able to **receive** user data in a manner protected from **replay, deletion, modification and insertion** errors.

FDP_UIT.1.2/CM The TSF shall be able to determine on receipt of user data, whether **replay, insertion, deletion and modification** has occurred.

Non editorial refinement:

The modification, deletion, insertion, replay apply for some of the pieces of the application sent by the CAD.

FMT_MSA.1/CM Management of security attributes

FMT_MSA.1.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to restrict the ability to **modify** the security attributes **AIDs** to **none**.

FMT_MSA.3/CM Static attribute initialisation

FMT_MSA.3.1/CM The TSF shall enforce the **PACKAGE LOADING information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/CM [Editorially Refined] The TSF shall allow **none** to specify alternative initial values to override the default values when an object or information is created.

FTP_ITC.1/CM Inter-TSF trusted channel

FTP_ITC.1.1/CM The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2/CM [Editorially Refined] The TSF shall permit **the CAD placed in the card issuer secured environment** to initiate communication via the trusted channel.

FTP_ITC.1.3/CM The TSF shall initiate communication via the trusted channel for **installing a new application package on the card**.

Application note:

There is no dynamic package loading on the Java Card platform. New packages can be installed on the card only on demand of the Card issuer.

5.2 TOE security assurance requirements

The [PP/BSI-0002] requirements for the IC evaluation are consistent with these requirements.

The security assurance requirement level is EAL4. The EAL is augmented with AVA_MSU.3, AVA_VLA.4, ADV_IMP.2 and ALC_DVS.2.

5.2.1 ACM Configuration management

5.2.1.1 ACM_AUT CM automation

ACM_AUT.1 Partial CM automation

ACM_AUT.1.1D The developer shall use a CM system.

ACM_AUT.1.2D The developer shall provide a CM plan.

ACM_AUT.1.1C The CM system shall provide an automated means by which only authorised changes are made to the TOE implementation representation.

ACM_AUT.1.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.1.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.1.4C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.2 ACM_CAP CM capabilities

ACM_CAP.4 Generation support and acceptance procedures

ACM_CAP.4.1D The developer shall provide a reference for the TOE.

ACM_CAP.4.2D The developer shall use a CM system.

ACM_CAP.4.3D The developer shall provide CM documentation.

ACM_CAP.4.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.4.2C The TOE shall be labelled with its reference.

ACM_CAP.4.3C The CM documentation shall include a configuration list, a CM plan, and an acceptance plan.

ACM_CAP.4.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.4.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.4.6C The CM documentation shall describe the method used to uniquely identify the configuration items.

ACM_CAP.4.7C The CM system shall uniquely identify all configuration items.

ACM_CAP.4.8C The CM plan shall describe how the CM system is used.

ACM_CAP.4.9C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.4.10C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.4.11C The CM system shall provide measures such that only authorised changes are made to the configuration items.

ACM_CAP.4.12C The CM system shall support the generation of the TOE.

ACM_CAP.4.13C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

ACM_CAP.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.3 ACM_SCP CM scope

ACM_SCP.2 Problem tracking CM coverage

ACM_SCP.2.1D The developer shall provide a list of configuration items for the TOE.

ACM_SCP.2.1C The list of configuration items shall include the following: implementation representation; security flaws; and the evaluation evidence required by the assurance components in the ST.

ACM_SCP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2 ADO Delivery and operation

5.2.2.1 ADO_DEL Delivery

ADO_DEL.2 Detection of modification

ADO_DEL.2.1D The developer shall document procedures for delivery of the TOE or parts of it to the user.

ADO_DEL.2.2D The developer shall use the delivery procedures.

ADO_DEL.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to a user's site.

ADO_DEL.2.2C The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

ADO_DEL.2.3C The delivery documentation shall describe how the various procedures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

ADO_DEL.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 ADO_IGS Installation, generation and start-up

ADO_IGS.1 Installation, generation, and start-up procedures

ADO_IGS.1.1D The developer shall document procedures necessary for the secure installation, generation, and start-up of the TOE.

ADO_IGS.1.1C The installation, generation and start-up documentation shall describe all the steps necessary for secure installation, generation and start-up of the TOE.

ADO_IGS.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADO_IGS.1.2E The evaluator shall determine that the installation, generation, and start-up procedures result in a secure configuration.

5.2.3 ADV Development

5.2.3.1 ADV_FSP Functional specification

ADV_FSP.2 Fully defined external interfaces

ADV_FSP.2.1D The developer shall provide a functional specification.

ADV_FSP.2.1C The functional specification shall describe the TSF and its external interfaces using an informal style.

ADV_FSP.2.2C The functional specification shall be internally consistent.

ADV_FSP.2.3C The functional specification shall describe the purpose and method of use of all external TSF interfaces, providing complete details of all effects, exceptions and error messages.

ADV_FSP.2.4C The functional specification shall completely represent the TSF.

ADV_FSP.2.5C The functional specification shall include rationale that the TSF is completely represented.

ADV_FSP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_FSP.2.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.2 ADV_HLD High-level design

ADV_HLD.2 Security enforcing high-level design

ADV_HLD.2.1D The developer shall provide the high-level design of the TSF.

ADV_HLD.2.1C The presentation of the high-level design shall be informal.

ADV_HLD.2.2C The high-level design shall be internally consistent.

ADV_HLD.2.3C The high-level design shall describe the structure of the TSF in terms of subsystems.

ADV_HLD.2.4C The high-level design shall describe the security functionality provided by each subsystem of the TSF.

ADV_HLD.2.5C The high-level design shall identify any underlying hardware, firmware, and/or software required by the TSF with a presentation of the functions provided by the supporting protection mechanisms implemented in that hardware, firmware, or software.

ADV_HLD.2.6C The high-level design shall identify all interfaces to the subsystems of the TSF.

ADV_HLD.2.7C The high-level design shall identify which of the interfaces to the subsystems of the TSF are externally visible.

ADV_HLD.2.8C The high-level design shall describe the purpose and method of use of all interfaces to the subsystems of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_HLD.2.9C The high-level design shall describe the separation of the TOE into TSP-enforcing and other subsystems.

ADV_HLD.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_HLD.2.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.3 ADV_IMP Implementation representation

ADV_IMP.2 Implementation of the TSF

ADV_IMP.2.1D The developer shall provide the implementation representation for the entire TSF.

ADV_IMP.2.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP.2.2C The implementation representation shall be internally consistent.

ADV_IMP.2.3C The implementation representation shall describe the relationships between all portions of the implementation.

ADV_IMP.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_IMP.2.2E The evaluator shall determine that the implementation representation is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.4 ADV_LLD Low-level design

ADV_LLD.1 Descriptive low-level design

ADV_LLD.1.1D The developer shall provide the low-level design of the TSF.

ADV_LLD.1.1C The presentation of the low-level design shall be informal.

ADV_LLD.1.2C The low-level design shall be internally consistent.

ADV_LLD.1.3C The low-level design shall describe the TSF in terms of modules.

ADV_LLD.1.4C The low-level design shall describe the purpose of each module.

ADV_LLD.1.5C The low-level design shall define the interrelationships between the modules in terms of provided security functionality and dependencies on other modules.

ADV_LLD.1.6C The low-level design shall describe how each TSP-enforcing function is provided.

ADV_LLD.1.7C The low-level design shall identify all interfaces to the modules of the TSF.

ADV_LLD.1.8C The low-level design shall identify which of the interfaces to the modules of the TSF are externally visible.

ADV_LLD.1.9C The low-level design shall describe the purpose and method of use of all interfaces to the modules of the TSF, providing details of effects, exceptions and error messages, as appropriate.

ADV_LLD.1.10C The low-level design shall describe the separation of the TOE into TSP-enforcing and other modules.

ADV_LLD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ADV_LLD.1.2E The evaluator shall determine that the low-level design is an accurate and complete instantiation of the TOE security functional requirements.

5.2.3.5 ADV_RCR Representation correspondence

ADV_RCR.1 Informal correspondence demonstration

ADV_RCR.1.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.1.1C For each adjacent pair of provided TSF representations, the analysis shall demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3.6 ADV_SPM Security policy modeling

ADV_SPM.1 Informal TOE security policy model

ADV_SPM.1.1D The developer shall provide a TSP model.

ADV_SPM.1.2D The developer shall demonstrate correspondence between the functional specification and the TSP model.

ADV_SPM.1.1C The TSP model shall be informal.

ADV_SPM.1.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.1.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.1.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 AGD Guidance documents

5.2.4.1 AGD_ADM Administrator guidance

AGD_ADM.1 Administrator guidance

AGD_ADM.1.1D The developer shall provide administrator guidance addressed to system administrative personnel.

AGD_ADM.1.1C The administrator guidance shall describe the administrative functions and interfaces available to the administrator of the TOE.

AGD_ADM.1.2C The administrator guidance shall describe how to administer the TOE in a secure manner.

AGD_ADM.1.3C The administrator guidance shall contain warnings about functions and privileges that should be controlled in a secure processing environment.

AGD_ADM.1.4C The administrator guidance shall describe all assumptions regarding user behaviour that are relevant to secure operation of the TOE.

AGD_ADM.1.5C The administrator guidance shall describe all security parameters under the control of the administrator, indicating secure values as appropriate.

AGD_ADM.1.6C The administrator guidance shall describe each type of security-relevant event relative to the administrative functions that need to be performed, including changing the security characteristics of entities under the control of the TSF.

AGD_ADM.1.7C The administrator guidance shall be consistent with all other documentation supplied for evaluation.

AGD_ADM.1.8C The administrator guidance shall describe all security requirements for the IT environment that are relevant to the administrator.

AGD_ADM.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4.2 AGD_USR User guidance

AGD_USR.1 User guidance

AGD_USR.1.1D The developer shall provide user guidance.

AGD_USR.1.1C The user guidance shall describe the functions and interfaces available to the non-administrative users of the TOE.

AGD_USR.1.2C The user guidance shall describe the use of user-accessible security functions provided by the TOE.

AGD_USR.1.3C The user guidance shall contain warnings about user-accessible functions and privileges that should be controlled in a secure processing environment.

AGD_USR.1.4C The user guidance shall clearly present all user responsibilities necessary for secure operation of the TOE, including those related to assumptions regarding user behaviour found in the statement of TOE security environment.

AGD_USR.1.5C The user guidance shall be consistent with all other documentation supplied for evaluation.

AGD_USR.1.6C The user guidance shall describe all security requirements for the IT environment that are relevant to the user.

AGD_USR.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5 ALC Life cycle support

5.2.5.1 ALC_DVS Development security

ALC_DVS.2 Sufficiency of security measures

ALC_DVS.2.1D The developer shall produce development security documentation.

ALC_DVS.2.1C The development security documentation shall describe all the physical, procedural, personnel, and other security measures that are necessary to protect the confidentiality and integrity of the TOE design and implementation in its development environment.

ALC_DVS.2.2C The development security documentation shall provide evidence that these security measures are followed during the development and maintenance of the TOE.

ALC_DVS.2.3C The evidence shall justify that the security measures provide the necessary level of protection to maintain the confidentiality and integrity of the TOE.

ALC_DVS.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ALC_DVS.2.2E The evaluator shall confirm that the security measures are being applied.

5.2.5.2 ALC_LCD Life cycle definition

ALC_LCD.1 Developer defined life-cycle model

ALC_LCD.1.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.1.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.1.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.1.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.5.3 ALC_TAT Tools and techniques

ALC_TAT.1 Well-defined development tools

ALC_TAT.1.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.1.2D The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.1.1C All development tools used for implementation shall be well-defined.

ALC_TAT.1.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.1.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

ALC_TAT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6 ATE Tests

5.2.6.1 ATE_COV Coverage

ATE_COV.2 Analysis of coverage

ATE_COV.2.1D The developer shall provide an analysis of the test coverage.

ATE_COV.2.1C The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.2.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

ATE_COV.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.2 ATE_DPT Depth

ATE_DPT.1 Testing: high-level design

ATE_DPT.1.1D The developer shall provide the analysis of the depth of testing.

ATE_DPT.1.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design.

ATE_DPT.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.3 ATE_FUN Functional tests

ATE_FUN.1 Functional testing

ATE_FUN.1.1D The developer shall test the TSF and document the results.

ATE_FUN.1.2D The developer shall provide test documentation.

ATE_FUN.1.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.1.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.1.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.1.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.1.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

ATE_FUN.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.6.4 ATE_IND Independent testing

ATE_IND.2 Independent testing - sample

ATE_IND.2.1D The developer shall provide the TOE for testing.

ATE_IND.2.1C The TOE shall be suitable for testing.

ATE_IND.2.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

ATE_IND.2.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

ATE_IND.2.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.2.3E The evaluator shall execute a sample of tests in the test documentation to verify the developer test results.

5.2.7 AVA Vulnerability assessment

5.2.7.1 AVA_MSU Misuse

AVA_MSU.3 Analysis and testing for insecure states

AVA_MSU.3.1D The developer shall provide guidance documentation.

AVA_MSU.3.2D The developer shall document an analysis of the guidance documentation.

AVA_MSU.3.1C The guidance documentation shall identify all possible modes of operation of the TOE (including operation following failure or operational error), their consequences and implications for maintaining secure operation.

AVA_MSU.3.2C The guidance documentation shall be complete, clear, consistent and reasonable.

AVA_MSU.3.3C The guidance documentation shall list all assumptions about the intended environment.

AVA_MSU.3.4C The guidance documentation shall list all requirements for external security measures (including external procedural, physical and personnel controls).

AVA_MSU.3.5C The analysis documentation shall demonstrate that the guidance documentation is complete.

AVA_MSU.3.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_MSU.3.2E The evaluator shall repeat all configuration and installation procedures, and other procedures selectively, to confirm that the TOE can be configured and used securely using only the supplied guidance documentation.

AVA_MSU.3.3E The evaluator shall determine that the use of the guidance documentation allows all insecure states to be detected.

AVA_MSU.3.4E The evaluator shall confirm that the analysis documentation shows that guidance is provided for secure operation in all modes of operation of the TOE.

AVA_MSU.3.5E The evaluator shall perform independent testing to determine that an administrator or user, with an understanding of the guidance documentation, would reasonably be able to determine if the TOE is configured and operating in a manner that is insecure.

5.2.7.2 AVA_SOF Strength of TOE security functions

AVA_SOF.1 Strength of TOE security function evaluation

AVA_SOF.1.1D The developer shall perform a strength of TOE security function analysis for each mechanism identified in the ST as having a strength of TOE security function claim.

AVA_SOF.1.1C For each mechanism with a strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the minimum strength level defined in the PP/ST.

AVA_SOF.1.2C For each mechanism with a specific strength of TOE security function claim the strength of TOE security function analysis shall show that it meets or exceeds the specific strength of function metric defined in the PP/ST.

AVA_SOF.1.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_SOF.1.2E The evaluator shall confirm that the strength claims are correct.

5.2.7.3 AVA_VLA Vulnerability analysis

AVA_VLA.4 Highly resistant

AVA_VLA.4.1D The developer shall perform a vulnerability analysis.

AVA_VLA.4.2D The developer shall provide vulnerability analysis documentation.

AVA_VLA.4.1C The vulnerability analysis documentation shall describe the analysis of the TOE deliverables performed to search for ways in which a user can violate the TSP.

AVA_VLA.4.2C The vulnerability analysis documentation shall describe the disposition of identified vulnerabilities.

AVA_VLA.4.3C The vulnerability analysis documentation shall show, for all identified vulnerabilities, that the vulnerability cannot be exploited in the intended environment for the TOE.

AVA_VLA.4.4C The vulnerability analysis documentation shall justify that the TOE, with the identified vulnerabilities, is resistant to obvious penetration attacks.

AVA_VLA.4.5C The vulnerability analysis documentation shall show that the search for vulnerabilities is systematic.

AVA_VLA.4.6C The vulnerability analysis documentation shall provide a justification that the analysis completely addresses the TOE deliverables.

AVA_VLA.4.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

AVA_VLA.4.2E The evaluator shall conduct penetration testing, building on the developer vulnerability analysis, to ensure the identified vulnerabilities have been addressed.

AVA_VLA.4.3E The evaluator shall perform an independent vulnerability analysis.

AVA_VLA.4.4E The evaluator shall perform independent penetration testing, based on the independent vulnerability analysis, to determine the exploitability of additional identified vulnerabilities in the intended environment.

AVA_VLA.4.5E The evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.

5.3 Security requirements for the IT environment

5.3.1 IT environment functional requirements

5.3.1.1 Java Card

Bytecode verification (BCVG)

FDP_IFC.2/BCV Complete information flow control

FDP_IFC.2.1/BCV The TSF shall enforce the **TYPING information flow control SFP** on **S.LOCVAR, S.STCKPOS, S.FLD, S.MTHD** and all operations that cause that information to flow to and from subjects covered by the SFP.

Non editorial refinement:

Subjects (prefixed with an " *S* ") covered by this policy are:

Subject	Description
S.LOCVAR	Any local variable of the currently executed method.
S.STCKPOS	Any operand stack position of the currently executed method.
S.FLD	Any field declared in a package loaded on the card.
S.MTHD	Any method declared in a package loaded on the card.

The operations (prefixed with " *OP* ") that make information flow between the subjects are all bytecodes. For instance, the *aload_0* bytecode causes information to flow from the local variable 0 to the top of the operand stack; the bytecode *putfield(x)* makes information flow from the top of the operand stack to the field x; and the *return_a* bytecode makes information flow out of the currently executed method.

Operation	Description
OP.BYTECODE(BYTCD)	Any bytecode for the Java Card platform ("Java Card bytecode").

The information (prefixed with an " *I* ") controlled by the typing policy are the bytes, shorts, integers, references and return addresses contained in the different storage units of the JCVM (local variables, operand stack, static fields, instance fields and array positions).

Information	Description
I.BYTE(BY)	Any piece of information that can be encoded in a byte.
I.SHORT(SH)	Any piece of information that can be encoded in a short value.
I.INT(W1,W2)	Any piece of information that can be encoded in an integer value, which in turn is encoded in two words w1 and w2.
I.REFERENCE(RF)	Any reference to a class instance or an array.
I.ADDRESS(ADRS)	Any return address of a subroutine.

FDP_IFC.2.2/BCV The TSF shall ensure that all operations that cause any information in the TSC to flow to and from any subject in the TSC are covered by an information flow control SFP.

FDP_IFF.2/BCV Hierarchical security attributes

FDP_IFF.2.1/BCV The TSF shall enforce the **TYPING information flow control SFP** based on the following types of subject and information security attributes: **(1) type attribute of the information, (2) type attribute of the storage units of the JCVM, (3) class attribute of the fields and methods, (4) bounds attribute of the methods**.

Non editorial refinement:

The following table describes which security attributes are attached to which subject/information of our policy.

Subject / Information	Attributes
S.LOCVAR	*TYPE*
S.STCKPOS	*TYPE*
S.FLD	*TYPE, CLASS*
S.MTHD	*TYPE, CLASS, BOUNDS*
I.BYTE(BY)	*TYPE*
I.SHORT(SH)	*TYPE*
I.INT(W1,W2)	*TYPE*
I.REFERENCE(RF)	*TYPE*
I.ADDRESS(ADRS)	*TYPE*

The following table describes the security attributes.

Attribute Name	Description
TYPE	Either the type attached to the information, or the type held or declared by the subject.
CLASS	The class where a field or method is declared.
BOUNDS	The start and end of the method code inside the method component of the CAP file where it is declared.

The *TYPE* security attribute attached to local variables and operand stack positions is the type of information they currently hold. The *TYPE* attribute of the fields and the methods is the type declared for them by the programmer.

The *BOUNDS* attribute of a method is used to prevent control flow to jump outside the currently executed method.

The following table describes the possible values for each security attribute.

Name	Description
TYPE	byte, short, int_1, int_2, any class name *C*, *T*[] with *T* any type in the Java Card platform ("Java Card type"), T_0 (T_1 x_1,... T_n x_n) with T_0,... T_n any Java Card type, RetAddrs(adrs), Top, Null, ⊥.
CLASS	The name of a class, represented as a reference into the class Component of one of the packages loaded on the card.
BOUNDS	Two integers marking a rank into the method component of a package loaded on the card.

Byte values have type ***byte*** and short values have type ***short***. The first and second halves of an integer value has respectively type ***int_1***, and ***int_2***. The type of a reference to an instance of the class *C* is ***C*** itself. A reference to an array of elements of type *T* has type ***T[]***. From the previous basic types it is possible to build the type T_0 (T_1 x_1,... T_n x_n) of a method. A return address *adrs* of a subroutine has type ***RetAddrss*** *(adrs)*. Finally, the former Java Card types are extended with three extra types **Top**, **Null** and ⊥, so that the domain of types forms a complete lattice. **Top** is the type of any piece of data, that is, the maximum of the lattice. **Null** is the type of the default value null of all the reference types (classes and arrays). ⊥ is the type of an element that belongs to all types (for instance the value 0, provided that null is represented as zero).

FDP_IFF.2.2/BCV The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules, based on the ordering relationships between security attributes hold:

The following rules constitute a synthetic formulation of the information flow control:

R.JAVA.6 If the bytecode pushes values from the operand stack, then there are a sufficient number of values on the stack and the values of the attribute *TYPE* of the top positions of the stack is appropriate with respect to the ones expected by the bytecode.

R.JAVA.7 If the bytecode pushes values onto the operand stack, then there is sufficient room on the operand stack for the new values. The values, with the appropriate attribute *TYPE* value are added to the top of the operand stack.

R.JAVA.8 If the bytecode modifies a local variable with a value with attribute *TYPE* T, it must be recorded that the local variable now contains a value of that type. In addition, the variable shall be among the local variables of the method.

R.JAVA.9 If the bytecode reads a local variable, it must be ensured that the specified local variable contains a value with the attribute *TYPE* specified by the bytecode.

R.JAVA.10 If the bytecode uses a field, it must be ensured that its value is of an appropriate type. This type is indicated by the *CLASS* attribute of the field.

R.JAVA.11 If the bytecode modifies a field, then it must be ensured that the value to be assigned is of an appropriate type. This type is indicated by the *CLASS* attribute of the field

R.JAVA.12 If the bytecode is a method invocation, it must be ensured that it is invoked with arguments of the appropriate type. These types are indicated by the *TYPE* and *CLASS* attributes of the method.

R.JAVA.13 If the bytecode is a branching instruction, then the bytecode target must be defined within the *BOUNDS* of the method in which the branching instruction is defined.

FDP_IFF.2.3/BCV The TSF shall enforce the **(following additional information flow control SFP rules): none**.

FDP_IFF.2.4/BCV The TSF shall provide the following **(list of additional SFP capabilities): none.**

FDP_IFF.2.5/BCV The TSF shall explicitly authorise an information flow based on the following rules: **none**.

FDP_IFF.2.6/BCV The TSF shall explicitly deny an information flow based on the following rules: **none**.

FDP_IFF.2.7/BCV The TSF shall enforce the following relationships for any two valid information flow control security attributes:

a) There exists an ordering function that, given two valid security attributes, determines if the security attributes are equal, if one security attribute is greater than the other, or if the security attributes are incomparable; and

b) There exists a "least upper bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is greater than or equal to the two valid security attributes; and

c) There exists a "greatest lower bound" in the set of security attributes, such that, given any two valid security attributes, there is a valid security attribute that is not greater than the two valid security attributes.

Application note:

FDP_IFF.2.2

The rules described above are strongly inspired in the rules described in section 4.9 of [JVM], Second Edition. The complete set of typing rules can be derived from the "Must" clauses from Chapter 7 of [JCVM221] as instances of the rules defined above.

FDP_IFF.2.7

The order relationship between Java Card types is described, for instance, in the description of the **checkcast** bytecode of [JCVM221]. That relation is with the following rules:

- **Top** is the maximum of all types;
- **Null** is the minimum of all classes and array types;
- ⊥ is the minimum of all types.

These three extra types are introduced in order to satisfy the two last items in requirement *FDP_IFF.2.7*.

FMT_MSA.1/BCV.1 Management of security attributes

FMT_MSA.1.1/BCV.1 The TSF shall enforce the **TYPING information flow control SFP** to restrict the ability to **modify** the security attributes **TYPE security attribute of the fields and methods** to **none**.

FMT_MSA.1/BCV.2 Management of security attributes

FMT_MSA.1.1/BCV.2 The TSF shall enforce the **TYPING information flow control SFP** to restrict the ability to **modify** the security attributes **TYPE security attribute of local variables and operand stack position** to **the role Bytecode Verifier**.

Application note:

1. See *FMT_SMR.1.1* for the roles.

2. *FMT_MSA.1.1/BCV.2*. The TYPE attribute of the local variables and the operand stack positions is identified to the attribute of the information they hold. Therefore, this security attribute is possibly modified as information flows. For instance, the rules of the typing function enable information to flow from a local variable *lv* to the operand stack by the operation *sload*, provided that the value of the type attribute of *lv* is ***short***. This operation hence modifies the type attribute of the top of the stack. The modification of the security attributes should be done according to the typing rules derived from Chapter 7 of [JCVM221].

FMT_MSA.2/BCV Secure security attributes

FMT_MSA.2.1/BCV The TSF shall ensure that only secure values are accepted for security attributes.

Application note:

During the type verification of a method, the bytecode verifier makes intensive use of the information provided in the CAP format like the sub-class relationship between the classes declared in the package, the type and class declared for each method and field, the rank of exceptions associated to each method, and so on. All that information can be thought of as security attributes used by the bytecode verifier, or as information relating security attributes. Moreover, the bytecode verifier relies on several properties about the CAP format. All the properties on the CAP format required by the bytecode verifier could, for instance, be completely described before starting type verifications. Examples of such properties are:

- Correspondences between the different components of the CAP file (for instance, each class in the class component has an entry in the descriptor component).

- Pointer soundness (example: the index argument in a static method invocation always has an entry in the constant pool);
- Absence of hanged pointers (example: each exception handler points to the beginning of some bytecode);
- Redundant information (enabling different ways of searching for it);
- Conformance to the Java Language Specification respecting the access control features mentioned in §2.2 of [JCVM221].
- Packages that are loaded post-issuance can not contain native code.

FMT_MSA.3/BCV Static attribute initialisation

FMT_MSA.3.1/BCV The TSF shall enforce the **TYPING information flow control SFP** to provide **restrictive** default values for security attributes that are used to enforce the SFP.

FMT_MSA.3.2/BCV The TSF shall allow the **none** to specify alternative initial values to override the default values when an object or information is created.

Application note:

FMT_MSA.3.1 The TYPE attribute of the fields and methods is fixed by the application provider and never modified. When a method is invoked, the operand (type) stack is empty. The initial type assigned to those local variables that correspond to the method parameters is the type the application provider declared for those parameters. Any other local variable used in the method is set to the default value Top.

FMT_MSA.3.2 The intent is to have none of the identified roles to have privileges with regards to the default values of the TYPE attributes.

FMT_SMR.1/BCV Security roles

FMT_SMR.1.1/BCV The TSF shall maintain the roles **Bytecode Verifier**.

FMT_SMR.1.2/BCV The TSF shall be able to associate users with roles.

FRU_RSA.1/BCV Maximum quotas

FRU_RSA.1.1/BCV The TSF shall enforce maximum quotas of the following resources: **the operand stack and the local variables** that **individual user** can use **simultaneously**.

Non editorial refinement:

Individual user is **a method** here.

6 TOE summary specification

6.1 TOE security functions

The security functions of the product are those of both the hardware and the software platforms. The Security Functions that reply to the SFR of the IC are described in [ST/Infineon].

The minimum strength for the security functions is SOF-high.

6.1.1 Card Operation

This section contains security functions dedicated to the operating system that may be used by all the applications on the card.

SF.SmartCardPlatform

The TOE runs tests at power on to check it has not been corrupted.

The TOE provides the ability to check the integrity of sensitive data stored in the card.

The TOE hides sensitive data transfers and operations from outside.

The TOE is protected against power and electromagnetic analysis, DFA & timing attacks.

It detects physical tampering and provides automatic response.

The data or operations protection against attacks uses probabilistic or permutational effects and has to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

SF.CardManager

The Card Manager controls the card and applets Life Cycles.

The Card Manager loads applets into the card on behalf of the Byte Code Verifier. The Card Manager locks the loading of applets on the card on behalf of the Issuer.

The Card Manager loads OP keys into the card on behalf of the Operator. The Card Manager controls the flow of APDUs between off card applications and the current application (native or JC applet) and the access to the APDU buffer.

This SF uses probabilistic or permutational effects and has to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

Application note:

The product doesn't support Delegated Management.

SF.TrustedChannel

This SF establishes a trusted channel between the card and a remote IT product. The trusted channel is [GP] compliant. It protects the integrity and/or the confidentiality of data loading into the card.

Additionally, a DAP mechanism protects the integrity of package loading and the Non repudiation of origin.

Keys are protected in Integrity and confidentiality when loaded.

The GP secure channel uses probabilistic or permutational effects and has to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

6.1.2 Common services

Security functions serving both the OS and the JCS.

SF.Transaction

This SF manages transactions in the:

- OS;
- JCVM. It enables to create Java Objects within a transaction. Transaction management includes the rollback of operations according to [JCRE221].

The TSF preserves a secure state when failures occur.

This SF does not use probabilistic or permutational effects.

This function has no strength.

6.1.3 Java Card

SF.Attributes

This SF manages the values of the following security attributes: resident applets, active applets and currently selected applet. It also checks the consistency of applets' life cycle.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Crypto

This SF provides the crypto functions and their Java API. This includes:

- Key generation, Import, Export and Deletion,
- Cryptographic operations

for the following algorithms: TDES 2 keys, RSA 1536, 1792 and 2048 bits, SHA-1, Signature RSA_SHA_PKCS#1 & RSA_SHA_ISO9796-2. RSA is used in the CRT mode.

This SF uses probabilistic or permutational effects and has to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

Application note:

The TOE executes also other algorithms like DES (one key), RSA 1024 and CRC, but these algorithms are not strong enough for SOF high and therefore no FCS_COP functional requirement was created for any of them. These algorithms are considered as services and their usage requires special care (see AGD class).

SF.Erase

This SF ensures that sensitive data cannot be accessed upon and after some types of operations. This SF may be split in:

- Logical deletion of data upon package and/or applet(s) deletion,
- Physical deletion of data upon and after object(s) deletion.

  This SF erases the APDU buffer and the cryptographic buffer.

The hiding operations use probabilistic or permutational effects and have to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

SF.Firewall

The JCRE firewall enforces applet isolation. The *JCRE* shall allocate and manage a context for each *applet* or *package* installed respectively loaded on the card and its own JCRE context. *Applet* s cannot access each other's objects unless they are defined in the same package (they share the same context) or they use the object sharing mechanism supported by *JCRE*.

This SF participates to information confidentiality.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Install

This SF ensures that the package loading is performed safely without loss, substitution, addition, modification, repetition of data or any other integrity failure on the loaded data such as a wrong order in the delivery of data by incoming APDUs.

It also ensures a safe *applet* installation process.

It modifies the CAP files in a safe way and performs coherency checks on the CAP files. It verifies the export references of the packages upon linking them.

This SF ensures that Java Card objects such as classes, *package*s and *applet* instances use limited resources.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.JCRE

The *JCRE* owns JCRE entry points objects of which methods may be accessed by any context. Their fields may only be accessed by the JCRE context.

The reference of temporary JCRE entry point objects and global arrays cannot be stored in class variables, instance fields or array components.

The JCRE is the only one allowed to create JCRE entry point objects and global arrays and to define them as temporary or permanent.

The *JCRE* has access to any objects and methods owned by any context. But it only invokes the methods select(), process(), deselect(), getShareableInterfaceObject() defined in Applet class and MultiSelectable interface.

The *JCRE* is the currently active context when the *JCVM* starts running after a card reset.

It is the only context allowed to register applets.

The *JCRE* supplies transient memory management through JC System services.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Applet

The SF defines the behaviour of each Java Card application through a strictly defined interface named Applet class and to the management of the *AID* s. Each new application inherits its behaviour and the associated constraints from the Applet class model.

This SF realises applet code interpretation. From the interpretation point of view, the *applet* 's code is considered as data to read. There is no way for an *applet* to be executed independently or to access platform resources. Thus the *JCRE* decides to start and to stop the applet execution.

This SF participates to information confidentiality, information integrity and availability.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.Domain

This SF consists in memory management with the *JCRE* scope. It is realized thanks to OS services.

The *applet* needs some different types of storage areas to be executed. As *applet* has no direct access to internal resources in particular to memory, the *JCRE* manages the memory for the applets. For this purpose, some services have been defined. On its side, the *JCRE* has its own execution space in memory that is not accessible to the *applet*s.

This SF does not use probabilistic or permutational effects.

This function has no strength.

SF.PIN

This SF supplies to *applet* a mean to assume a user identification and authentication with the OwnerPin class.

It is supplied through a secure comparison between a PIN stored in the persistent memory and a data received from the *CAD*.

This SF participates to information confidentiality.

This SF uses probabilistic or permutational effects and has to be included in the AVA_SOF analysis with SOF high.

The strength of this function is SOF-high.

7 PP claims

[PP/JCS] Java Card™ System, Protection Profile Collection, Version 1.0b, Standard 2.1.1 configuration.

7.1 PP reference

[PP/JCS] is claimed. The Java Card TOE defined into this PP is extended to the actual TOE; consequently it covers the operating system.

7.2 PP additions

7.2.1 Additional Security Functional Requirements

The following SFR have been added to the ST main groups of SFRs compared to [PP/JCS]:

Java Card part:

- *FMT_SMF.1/CM*. It is required as dependency by *FMT_MSA.1*.
- *ADELG* from Standard 2.2 configuration
- *ODELG* from Standard 2.2 configuration

7.2.2 Operations that have been completed on SFRs from [PP/JCS]

CoreG:

- Firewall Policy: *FDP_IFF.1/JCVM*;
- Application Programming Interface: *FCS_CKM.1*, *FCS_CKM.2*, *FCS_CKM.3*, *FCS_CKM.4* and *FCS_COP.1* have been instantiated with the algorithms available on the card.
- Card Security Management: *FDP_SDI.2*, *FPR_UNO.1*.

Secure carrier (CarG):

- *FCO_NRO.2/CM*, *FDP_IFF.1/CM*, *FMT_MSA.1/CM*.

Smart card platform (SCPG):

- *FPT_FLS.1/SCP*, *FRU_FLT.1/SCP*, *FPT_PHP.3/SCP*.

Card manager (CMGRG):

- *FDP_ACF.1/CMGR*, *FMT_MSA.1/CMGR*, *FMT_MSA.3/CMGR*.

7.2.3 The following SFRs have been merged and refined when necessary, with respect to [PP/JCS]:

Please see the group *Merged SFRs* from the Java Card part.

7.2.4 Additional Assurance requirements

[PP/JCS] requires EAL4 augmented with:

- ADV_IMP.2
- AVA_VLA.3.

Additionally, this ST requires:

- ALC_DVS.2
- AVA_MSU.3
- AVA_VLA.4

8 Rationale

8.1 Environment rationale

Not delivered in public version.

8.2 Security objectives rationale

Not delivered in public version.

8.3 Security requirements rationale

Not delivered in public version.

8.4 TOE summary specification rationale

Not delivered in public version.

Notice

This document has been generated with TL SET version 1.7.1. The Security Editing Tool of Trusted Logic is available at www.trusted-logic.com.

Index

END OF SECURITY TARGET