Javacard Virtual Machine on MultiApp V4.0.1 Platform

Common Criteria / ISO 15408
Security Target – Public version
EAL7

TABLE OF CONTENTS

FIGURES

TABLES

1 SECURITY TARGET INTRODUCTION

1.1 SECURITY TARGET REFERENCE

Title :	Javacard Virtual Machine on MultiApp V4.0.1 Platform – Security Target
Version :	1.8p
ST Reference :	D1391107
Origin :	Gemalto and Trusted Labs
Authors :	Maria Christofi, Quang-Huy Nguyen
IT Security Evaluation scheme :	Serma Technologies
IT Security Certification scheme :	Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)

1.2 TOE REFERENCE

Product Name :	MultiApp V4
Security Controllers :	**M7892**
TOE Name :	MultiApp V4 Java Card Virtual Machine
TOE Reference :	4.0.1
TOE documentation :	Guidance [AGD]

The TOE identification is provided by the Tag identity and CPLC data.These data are available by executing a dedicated command. Information and values to identified TOE are described chapter 1.5 of [AGD-OPE] document.

The TOE and the product differ. Actually, the TOE is the Java Card Virtual Machine of the MultiApp V4 product.

1.3 SECURITY TARGET OVERVIEW

The main objectives of this ST are:

- To introduce TOE;
- To describe the TOE components, the components in the TOE environment, the product type, the TOE environment and life cycle and the limits of the TOE;
- To define the scope of the TOE and its security features;
- To describe the security environment of the TOE, including the assets to be protected and the threats to be countered by the TOE and its environment during the product development, production and usage;
- To describe the organizational security policies and the assumptions;
- To describe the security objectives of the TOE and its environment supporting in terms of integrity and confidentiality of application data and programs and of protection of the TOE;
- To specify the security requirements which includes the TOE security functional requirements, the TOE assurance requirements, the TOE security functions and the associated rationales.

1.4 REFERENCES

1.4.1 External References

[CC]	Common Criteria references
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, CCMB-2009-07-001, version 3.1 rev 4, September 2012
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, CCMB-2009-07-002, version 3.1 rev 4, September 2012
[CC-3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, CCMB-2009-07-003, version 3.1 rev 4 September 2012
[CEM]	Common Methodology for Information Technology Security Evaluation Methodology CCMB-2009-07-004, version 3.1 rev 4 September 2012
[PP]	**Protection profiles**
[PP-IC-0084]	Security IC Platform Protection Profile with augmentation Packages– BSI-CC-PP-0084-2014
[PP-JCS-Open]	Java Card System Protection Profile – Open Configuration ANSSI-PP-2010-03M01, Version 3.0, May 2012
[IFX]	**Infineon References**
[ST-IC]	[ST-IC-M7892]
[ST-IC-M7892]	Security Target Common Criteria EAL6 augmented / EAL6+ M7892 Design Steps D11 and G12 Revision 1.7 as of 2016-11-16
[CR-IC]	[CR-IC-M7892]
[CR-IC-M7892]	Certification Report, M7892 D11 & G12 BSI-DSZ-CC-0891-V2-2016
[GP]	**Global Platform references**
[GP221]	GlobalPlatform Card Technology Secure Channel Protocol 03 Card Specification v 2.2 – Amendment D Version 1.0 Public Release April 2009
[GP221 Id Config]	Global Platform – ID Configuration v1.0.
[GP221 Com]	Global Platform – Common Configuration v1.0.
[JCS]	**Javacard references**
[JAVASPEC]	The Java Language Specification. Third Edition, May 2005. Gosling, Joy, Steele and Bracha. ISBN 0-321-24678-0.
[JVM]	The Java Virtual Machine Specification. Lindholm, Yellin. ISBN 0-201-43294-3.
[JCBV]	Java Card Platform, version 2.2 Off-Card Verifier. June 2002. White paper. Published by Sun Microsystems, Inc.
[JCRE222]	Java Card 2.2.2 Runtime Environment (JCRE) Specification – 15 March 2006 - Published by Sun Microsystems, Inc.
[JCVM222]	Java Card 2.2.2 Virtual Machine (JCVM) Specification – 15 March 2006 - Published by Sun Microsystems, Inc.
[JCAPI222]	Java Card 2.2.2 Application Programming Interface - March 2006 - Published by Sun Microsystems, Inc.
[JCRE304]	Java Card 3.0.4 Runtime Environment (JCRE) Specification, Classic Edition – September 2011 – Published by Oracle
[JCVM304]	Java Card 3.0.4 Virtual Machine (JDVM) Specification, Classic Edition-– September 2011 – Published by Oracle
[JCAPI304]	Java Card 3.0.4 Application Programming Interface (API) Specification, Classic Edition- – September 2011 – Published by Oracle

1.4.2 Internal References [IR]

[AGD]	[AGD-PRE], [AGD-OPE], [AGD-USR], [AGD-VERIF], [AGD-GD-DEV]
[AGD-PRE]	MultiApp V4.0.1 – AGD_PRE document – Javacard Platform Ref: D1431347, Version 1.0
[AGD-OPE]	MultiApp V4.0.1 Javacard Platform – AGD_OPE document Ref: D1432683, Version 1.2
[AGD-USR]	MultiApp ID Operating System Reference Manual Ref: D1392687, Revision E
[AGD-VERIF]	Verification process of Gemalto non sensitive applet – Qualification level Ref: D1484874, Version 1.0 Verification process of Third Party non sensitive applet – Qualification level Ref: D1484875, Version 1.2
[AGD-GD-DEV]	Guidance for secure application development on Multiapp platforms Ref: D1390326, Version A01, March 2018 Rules for applications on Multiapp certified product – Qualification level Ref: D1484823, Version 1.2
[ALC-DVS]	SUFFICIENCY OF SECURITY MEASURES (D1402691)
[ST_MultiAppv4]	MultiApp V4.0.1 Javacard Platform – Security Target Ref: D1430789, Version 1.2

1.5 ACRONYMS AND GLOSSARY

APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
CC	Common Criteria
CPU	Central Processing Unit
EAL	Evaluation Assurance Level
EEPROM	Electrically-Erasable Programmable Read-Only Memory
ES	Embedded Software
GP	Global Platform
IC	Integrated Circuit
IT	Information Technology
JCRE	JavaCard Runtime Environment
JCS	JavaCard System
JCVM	JavaCard Virtual Machine
NVM	Non-Volatile Memory
OP	Open Platform
PIN	Personal Identification Number
PP	Protection Profile
RMI	Remote Method Invocation
RNG	Random Number Generator
ROM	Read-Only Memory
SAR	Security Assurance Requirement
SC	Smart Card
SCP	Secure Channel Protocol
SFP	Security Function Policy
SFR	Security Functional Requirement
ST	Security Target
TOE	Target Of Evaluation
TSF	TOE Security Functionality

2 TOE OVERVIEW

2.1 TOE TYPE

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 [JCVM304]. 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.

This security target defines the requirements of the Java Card Virtual Machine as a subset of the Java Card System, and corresponds to an extension of the evaluation of the *full* TOE of the product, described in the security target MultiAppV4: JCS Security Target [ST_MultiAppv4]. This security target restricts the security target [ST_MultiAppv4] to the virtual machine, in charge of the secure execution of the applets after their loading on the card.

More precisely, the TOE in this security target is made of:

- The linker
- The interpreter
- A (native) subset of the JC API

The TOE is a subset of the Java Card System whose configuration is defined in [PP-JCS-Open] Java Card System protection profile Open Configuration.

2.2 PRODUCT ARCHITECTURE

The TOE is part of the *MultiApp V4* smartcard product. This smartcard contains the software dedicated to the operation of:

- The MultiApp V4 Platform, which supports the execution of the personalized applets and provides the smartcard administration services. It is conformant to Java Card 3.0.4 and GP 2.2 standards [GP221]. (id configuration [GP221 Id Config] or GP configuration [GP221 Com])
- The identity applets: IAS V4.4, eTravel 2.2, Pure 2.1, Plug&Play, BioPin Management, MPCOS, OATH, e-ID, e-Sign (These applications could be removed based on the customer needs).
- Additionally, other applets – not determined at the moment of the present evaluation – may be loaded on the smartcard before or after issuance.

Therefore, the architecture of the smartcard software and application data can be represented as follows:

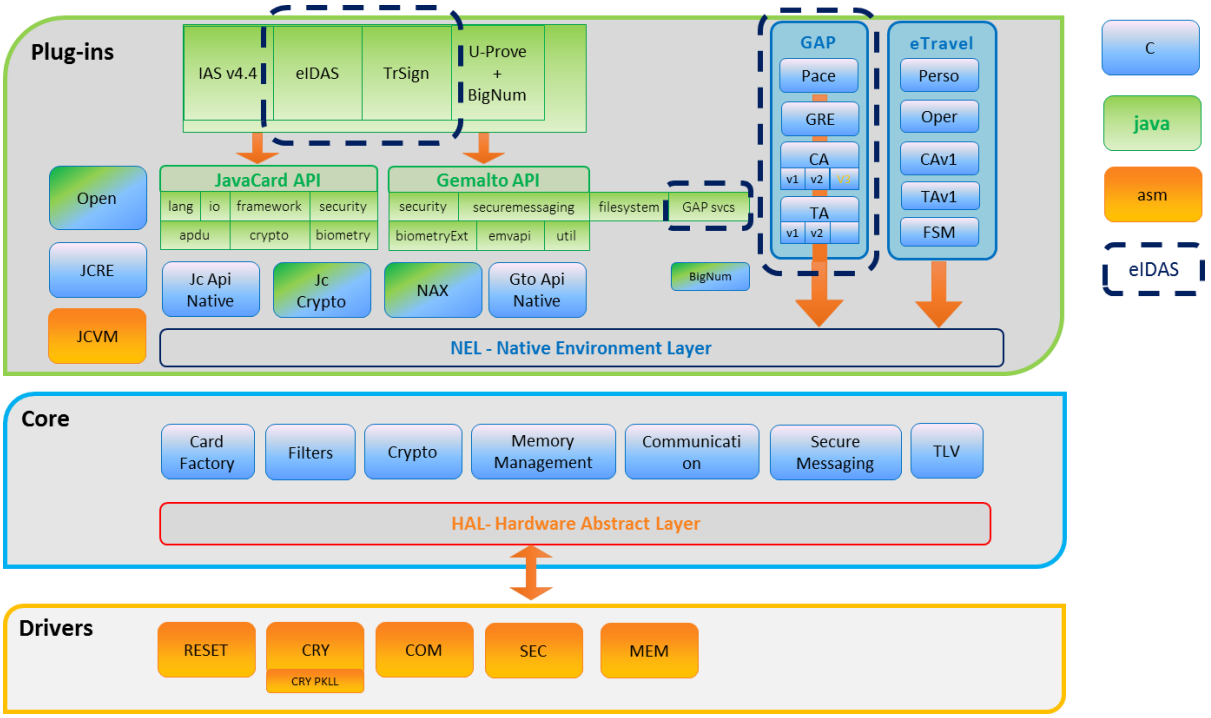

Figure 1: MultiApp V4 smartcard architecture

The applets and the MultiApp v4 Java Card platform, are located in flash code area.
All the data (related to the applets or to the Java Card platform) are located in flash data area. The separation between these data is ensured by the Java Card firewall as specified in [JCRE304].

MultiApp V4 product is a modular product where some features could be removed, based on the customer needs. (See identification and configuration option).

2.3 TOE BOUNDARIES

The Target of Evaluation (TOE) is the Java Card Virtual Machine that is embedded in Smart Card Integrated Circuit in operation and in accordance to its functional specifications. Other smart card product items and other embedded software (such as OS, Secure API, etc) are outside the scope of this evaluation.

Java Card RMI is not implemented in the TOE.

Figure 2 shows the TOE boundaries (presented in the red boxes).

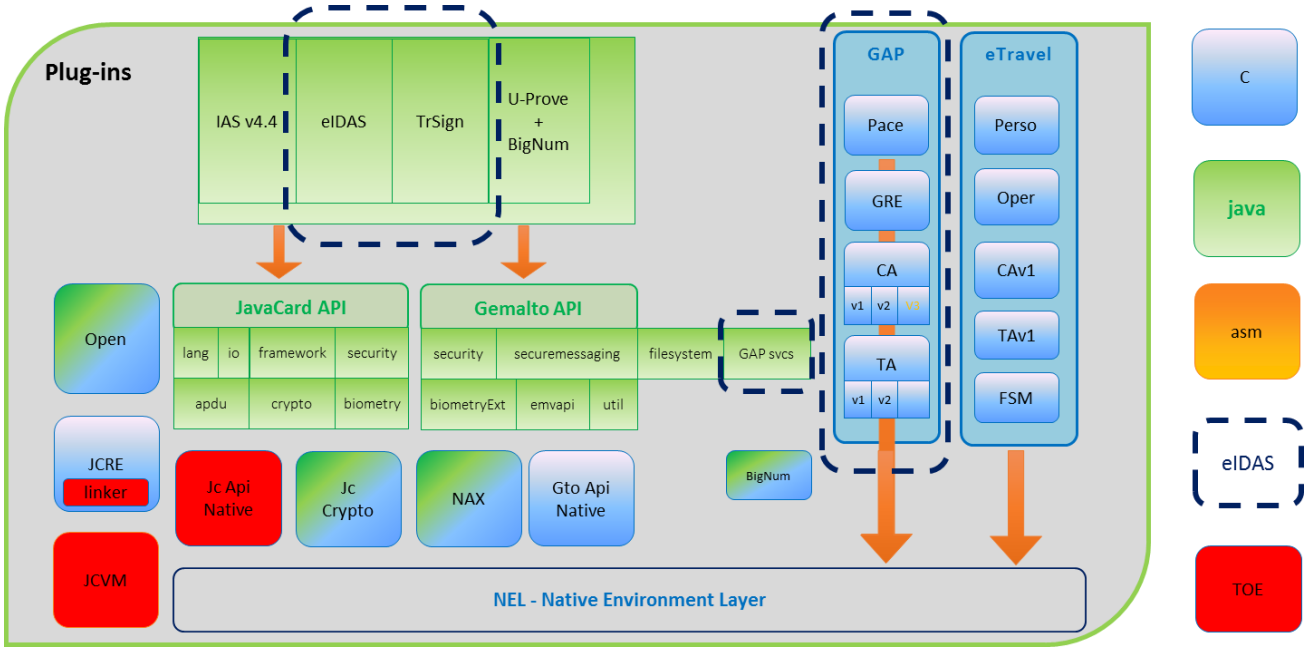

Figure 2: TOE boundaries

2.4 TOE DESCRIPTION

The present TOE is a subset (identified by the red boxes in Figure 2) of the Java Card System. This subset ensures the secure execution of an applet that has been byte code verified and loaded on the product. This execution is processed in two phases:

Phase 1: the (static) link of the loaded CAP file (done once)
Phase 2: the (dynamic) interpretation of the linked byte-codes (done as many as necessary)

The TOE is composed of the following components:

- The **linker** (used for CAP file preparing for interpretation),
- The **interpreter** (used for bytecode interpreting), and
- The **firewall-related native Java Card API** (to ensure that it provides no means to bypass the firewall access control)

All these components have the same version as the embedded software evaluated in [ST_MultiAppv4].

2.4.1 The linker

The linker is in charge of the rearrangement of the data structures contained in the Converted APplet (CAP) file in order to speed up the execution of the applet. The linker first performs a resolution step that is, resolves the external and internal references of a CAP file and replaces them by direct ones. Then it performs the preparation step, allocating the static field image and the static arrays. The later ones are also initialized, thus giving rise to the configuration that will constitute the corresponding initial state of the (JC) interpreter.

The linker contributes to the installation of post-issuance applets. The linker is invoked after the loading process by the JCRE to link the CAP file using the existing packages. Then, the API method `Applet.install` is invoked to instantiate the new application using a fresh AID. The other application management functionalities, such as the load process, the CAD communication, are out of the scope of the TOE.

During its lifetime, an applet can be updated by new packages to be loaded on the TOE. These (Java) updates may replace part of the original applet or extend it. The Java updates are also ensured by the linker.

2.4.2 The interpreter

Once an application is installed, registered and selected, its execution is carried out by the embedded interpreter. The interpreter mainly consists of a loop that computes the next bytecode to be executed and dispatches the appropriate interpretation functions. Such function modifies the runtime data areas of the JCVM (the heap, the static field images, the frame stack, etc) according to the semantics of the byte code interpreted.

2.4.3 The native Java Card API

The byte code interpretation done by the interpreter depends in turn on the behavior of methods of the API. The Java Card APIs consist of a set of customized classes for programming smart card applications according to the ISO 7816 model.

Native API methods are usually written in C and are considered as parts of the Java Card platform. The native methods participate in enforcing several essential TOE security features such as firewall. Consequently, the native methods of the package `javacard.framework` are then included in the TOE. The package `javacard.framework.service` that is mainly used for the JCRMI functionality is not included.

The packages `javacardx.crypto` and `javacard.security` are not included as they are beyond the TOE Life-cycle
The Java Card System (the TOE) life cycle is part of the product life cycle, i.e. the Java Card platform with applications, which goes from product development to its usage by the final user.

The Java Card System (i.e. the TOE) life-cycle itself can be decomposed in four stages:

- Development
- Storage, pre-personalization and testing
- Personalization and testing
- Final usage

The JCS storage is not necessarily a single step in the life cycle since it can be stored in parts. The JCS delivery occurs before storage and may take place more than once if the TOE is delivered in parts.
These four stages map to the product life cycle phases as shown in Figure 6.

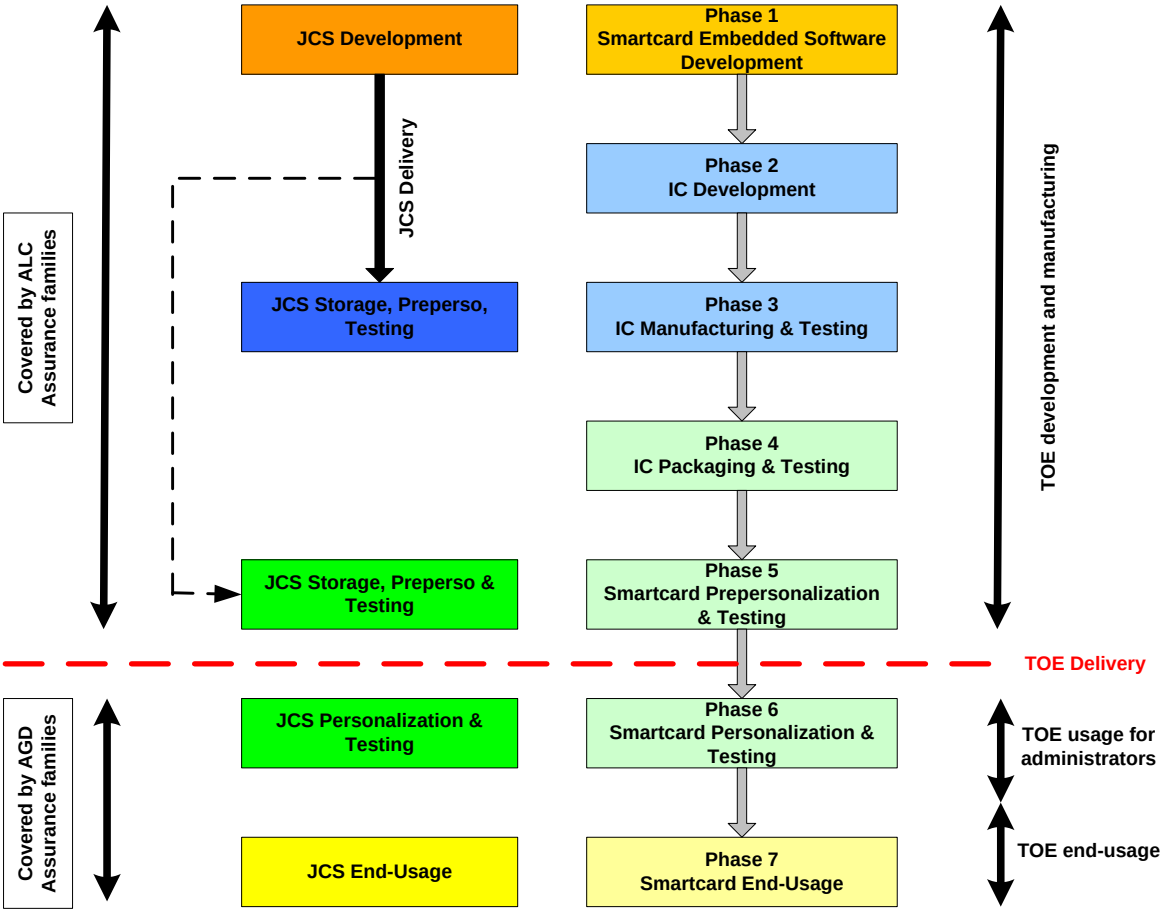

Figure 3: JCS (TOE) Life Cycle within Product Life Cycle

JCS Development is performed during Phase 1. This includes JCS conception, design, implementation, testing and documentation. The JCS development shall fulfill requirements of the final product, including conformance to Java Card Specifications, and recommendations of the SCP user guidance. The JCS development shall occur in a controlled environment that avoids disclosure of source code, data and any critical documentation and that guarantees the integrity of these elements. The present evaluation includes the JCS development environment.

In Phase 3, the IC Manufacturer may store, initialize the JCS and potentially conduct tests on behalf of the JCS developer. The IC Manufacturing environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites. The present evaluation includes the whole IC Manufacturing environment, in particular those locations where the JCS is accessible for installation or testing. As the Security IC has already been certified against [PP-IC-0084] there is no need to perform the evaluation again.

In Phase 5, the SC Pre-Personalizer may store, pre-personalize the JCS and potentially conduct tests on behalf of the JCS developer. The SC Pre-Personalization environment shall protect the integrity and confidentiality of the JCS and of any related material, for instance test suites.

(Part of) JCS storage in Phase 5 implies a TOE delivery after Phase 5. Hence, the present evaluation includes the SC Pre-Personalization environment. The TOE delivery point is placed at the end of Phase 5, since the entire TOE is then built and embedded in the Security IC.

The JCS is personalized in Phase 6, if necessary. The SC Personalization environment is not included in the present evaluation. Appropriate security recommendations are provided to the SC Personalizer through the [AGD] documentation.

The JCS final usage environment is that of the product where the JCS is embedded in. It covers a wide spectrum of situations that cannot be covered by evaluations. The JCS and the product shall provide the full set of security functionalities to avoid abuse of the product by untrusted entities.

Note: Potential applications loaded in pre-issuance will be verified using dedicated evaluated verification process. Applications loaded in post-issuance will need to follow dedicated development rules.

2.5 TOE INTENDED USAGE

The TOE is intended to be used as the Virtual Machine in a Java Card System. The intended usage of the latter is described in [ST_MultiAppv4].

2.6 ACTORS OF THE TOE

In the following table, we can see several entities participating in the system.

Actors	Identification
Integrated Circuit (IC) Developer	IFX
Embedded Software Developer	Gemalto (Meudon, Singapore, Vantaa)
Integrated Circuit (IC) Manufacturer	IFX
Initializer	Gemalto (Gemenos, Singapore, Vantaa, Tczew, Curitiba, Montgomery, PA)
Pre-personalizer	Gemalto (Gemenos, Singapore, Tczew, Curitiba, Vantaa, Montgomery, PA)
Personalization Agent	The agent who is acting on the behalf of the issuing State or Organization and personalize the MRTD for the holder by activities establishing the identity of the holder with biographic data.
Card Holder	The rightful holder of the card for whom the issuer personalizes it.

2.7 TOE SECURITY FEATURES

The principal security feature of the TOE is the correct and secure execution of applications (i.e. Java Card applets).

While the Java Card virtual machine (JCVM) is responsible for ensuring language-level security, the JCRE provides additional security features for the product. The basic runtime security feature imposed by the JCRE enforces isolation of applets using the Java Card **firewall**. It prevents objects created by one applet from being used by another applet without explicit sharing. This prevents unauthorized access to the fields and methods of class instances, as well as the length and contents of arrays.

The firewall is an important security feature which enables complete isolation between applets or controlled communication through additional mechanisms that allow them to share objects when needed. The JCRE allows such sharing using the concept of “shareable interface objects” (SIO) and static public variables. The JCVM should ensure that the only way for applets to access any resources are either through the JCRE or through the native API.
Among the security services provided by the platform to the applications to protect their data and assets, the TOE of this security target is in charge of:

- Confidentiality and integrity of application data among applications. Applications belonging to different contexts are isolated from each other. Application data are not accessible by a normal or abnormal execution of another basic or secure application.
- Application code execution integrity. The Java Card VM and the “applications isolation” property guarantee that the application code is operating as specified in absence of perturbations.

Other security services are ensured by the product and described in [ST_MultiAppv4].

2.8 NON-TOE HW/SW/FW AVAILABLE TO THE TOE

The TOE does not include the following SW components (that are part of the MultiAppv4 smartcard product):

- Part of the JCRE and JCAPI: Applet selection/deletion/loading, Object deletion, Cryptographic API;
- Native environment layer

The TOE does not include the following HW components (that are part of the MultiAppv4 smartcard product):

- Integrated Circuit and the FW (i.e. drivers)

3 CONFORMANCE CLAIMS

3.1 CC CONFORMANCE CLAIM

Common criteria Version:

This ST conforms to CC Version 3.1 [CC-1] [CC-2] [CC-3]

Conformance to CC part 2 and 3:

- CC part 2 conformant
- CC part 3 conformant

The Common Methodology for Information Technology Security Evaluation, Evaluation Methodology; [CEM] has to be taken into account.

The assurance requirement of this security target is **EAL7.**

3.2 PP CLAIM

This security target does not claim conformance to any Protection Profile, but it is based on the SPD and SFRs from the Protection Profile “JavaCard System – Open configuration” [PP-JCS-Open] with adaptations due to the reduced scope.

The security target is a composite security target, including the IC security target [ST-IC]. However the security problem definition, the objectives, and the SFR of the IC are not described in this document.

The TOE is part of the embedded software of the MultiAppV4 product evaluated in the [ST_MultiAppv4] that has a “demonstrable” conformance to [PP-JCS-Open].

Because the TOE is a subset of the reference TOE defined in [PP-JCS-Open], only a subset of its SFRs are enforced in this evaluation. Table 1 explains how the SFRs of PP are refined and used in this ST. Consequently, only a subset of the security objectives defined in PP are satisfied in this ST (because of the limited TOE security functions). The other objectives are put in the environment. Table 2 resumes the modifications done by this ST with respect to the [PP-JCS-Open].

Functional requirements	Refined in [PP-JCS]	Refined in this ST
FDP_ACC.2/FIREWALL	Yes	No
FDP_ACF.1/FIREWALL	Yes	Yes
FDP_IFC.1/JCVM	Yes	No
FDP_IFF.1/JCVM	Yes	Yes
FDP_RIP.1/OBJECTS	Yes	Not used
FMT_MSA.1/JCRE	Yes	Yes
FMT_MSA.1/JCVM	Yes	No
FMT_MSA.2/FIREWALL_JCVM	Yes	No
FMT_MSA.3/FIREWALL	Yes	No
FMT_MSA.3/JCVM	Yes	No
FMT_SMF.1	Yes	No
FMT_SMR.1	Yes	No
FCS_CKM.1	Yes	Not used
FCS_CKM.2	Yes	Not used
FCS_CKM.3	Yes	Not used
FCS_CKM.4	Yes	Not used
FCS_COP.1	Yes	Not used
FDP_RIP.1/ABORT	Yes	Not used
FDP_RIP.1/APDU	Yes	Not used
FDP_RIP.1/bArray	Yes	Not used
FDP_RIP.1/KEYS	Yes	Not used
FDP_RIP.1/TRANSIENT	Yes	Not used
FDP_ROL.1/FIREWALL	Yes	No
FAU_ARP.1	Yes	Yes
FDP_SDI.2	Yes	Not used
FPR_UNO.1	Yes	Not used
FPT_FLS.1	Yes	No
FPT_TDC.1	Yes	No
FIA_ATD.1/AID	Yes	No
FIA_UID.2/AID	Yes	Not used
FIA_USB.1/AID	Yes	Not used
FMT_MTD.1/JCRE	Yes	No
FMT_MTD.3/JCRE	Yes	No
FDP_ITC.2/Installer	Yes	Not used
FMT_SMR.1/Installer	Yes	Not used
FPT_FLS.1/Installer	Yes	Yes
FPT_RCV.3/Installer	Yes	Yes
FDP_ACC.2/ADEL	Yes	Not used
FDP_ACF.1/ADEL	Yes	Not used

Functional requirements	Refined in [PP-JCS]	Refined in this ST
FDP_RIP.1/ADEL	Yes	Not used
FMT_MSA.1/ADEL	Yes	Not used
FMT_MSA.3/ADEL	Yes	Not used
FMT_SMF.1/ADEL	Yes	Not used
FMT_SMR.1/ADEL	Yes	Not used
FPT_FLS.1/ADEL	Yes	Not used
FDP_ACC.2/JCRMI	Yes	Not used
FDP_ACF.1/JCRMI	Yes	Not used
FDP_IFC.1/JCRMI	Yes	Not used
FDP_IFF.1/JCRMI	Yes	Not used
FMT_MSA.1/EXPORT	Yes	Not used
FMT_MSA.1/REM_REFS	Yes	Not used
FMT_MSA.3/JCRMI	Yes	Not used
FMT_REV.1/JCRMI	Yes	Not used
FMT_SMF.1/JCRMI	Yes	Not used
FMT_SMR.1/JCRMI	Yes	Not used
FDP_RIP.1/ODEL	Yes	Not used
FPT_FLS.1/ODEL	Yes	Not used
FCO_NRO.2/CM	Yes	Not used
FDP_IFC.2/CM	Yes	Not used
FDP_IFF.1/CM	Yes	Not used
FDP_UIT.1/CM	Yes	Not used
FIA_UID.1/CM	Yes	Not used
FMT_MSA.1/CM	Yes	Not used
FMT_MSA.3/CM	Yes	Not used
FMT_SMF.1/CM	Yes	Not used
FMT_SMR.1/CM	Yes	Not used
FTP_ITC.1/CM	Yes	Not used

Table 1 - Refinement of SFR of PP JCS Open

PP JCS elements	Modification in ST
Assets	Not changed
Threats	Not changed
Assumptions	Augmented
OSP	Not changed
Security objectives	Reduced
Security objective for the operational environment	Augmented
Security functional requirements	Reduced

Security assurance requirements	Augmented

Table 2 - Compatibility study

3.3 PACKAGE CLAIM

This ST is conforming to assurance package EAL7.

4 SECURITY ASPECTS

This chapter describes the main security issues of the Java Card System and its environment addressed in this ST, called “security aspects”, in a CC-independent way. In addition to this, they also give a semi-formal framework to express the CC security environment and objectives of the TOE. They can be instantiated as assumptions, threats, objectives (for the TOE and the environment) or organizational security policies. For instance, we will define hereafter the following aspect:

#.OPERATE (1) The TOE must ensure continued correct operation of its security functions.
(2) The TOE must also return to a well-defined valid state before a service request in case of failure during its operation.

TSFs must be continuously active in one way or another; this is called “OPERATE”.

4.1 CONFIDENTIALITY

#.CONFID-APPLI-DATA	Application data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain read access to other application’s data.
#.CONFID-JCS-CODE	Java Card System code must be protected against unauthorized disclosure. Knowledge of the Java Card System code may allow bypassing the TSF. This concerns logical attacks at runtime in order to gain a read access to executable code, typically by executing an application that tries to read the memory area where a piece of Java Card System code is stored.
#.CONFID-JCS-DATA	Java Card System data must be protected against unauthorized disclosure. This concerns logical attacks at runtime in order to gain a read access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card platform API classes as well.

4.2 INTEGRITY

#.INTEG-APPLI-CODE	Application code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to the memory zone where executable code is stored. In post-issuance application loading, this threat also concerns the modification of application code in transit to the card.
#.INTEG-APPLI-DATA	Application data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain unauthorized write access to application data. In post-issuance application loading, this threat also concerns the modification of application data contained in a package in transit to the card. For instance, a package contains the values to be used for initializing the static fields of the package.
#.INTEG-JCS-CODE	Java Card System code must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to executable code.
#.INTEG-JCS-DATA	Java Card System data must be protected against unauthorized modification. This concerns logical attacks at runtime in order to gain write access to Java Card System data. Java Card System data includes the data managed by the Java Card RE, the Java Card VM and the internal data of Java Card API classes as well.

4.3 UNAUTHORIZED EXECUTIONS

#.EXE-APPLI-CODE	Application (byte)code must be protected against unauthorized execution. This concerns (1) invoking a method outside the scope of the accessibility rules provided

	by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code.;
#.EXE-JCS-CODE	Java Card System bytecode must be protected against unauthorized execution. Java Card System bytecode includes any code of the Java Card RE or API. This concerns (1) invoking a method outside the scope of the accessibility rules provided by the access modifiers of the Java programming language ([JAVASPEC]§6.6); (2) jumping inside a method fragment or interpreting the contents of a data memory area as if it was executable code. Note that execute access to native code of the Java Card System and applications is the concern of #.NATIVE.
#.FIREWALL	The Firewall shall ensure controlled sharing of class instances, and isolation of their data and code between packages (that is, controlled execution contexts) as well as between packages and the JCRE context. An applet shall neither read, write nor compare a piece of data belonging to an applet that is not in the same context, nor execute one of the methods of an applet in another context without its authorization.
#.NATIVE	Because the execution of native code is outside of the JCS TSF scope, it must be secured so as to not provide ways to bypass the TSFs of the JCS. Loading of native code, which is as well outside the TSFs, is submitted to the same requirements. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

4.4 BYTECODE VERIFICATION

#.VERIFICATION	All bytecode must be verified prior to being executed. Bytecode verification includes (1) how well-formed CAP file is and the verification of the typing constraints on the bytecode, (2) binary compatibility with installed CAP files and the assurance that the export files used to check the CAP file correspond to those that will be present on the card when loading occurs.

4.4.1 CAP file Verification

Bytecode verification includes checking at least the following properties: (1) bytecode instructions represent a legal set of instructions used on the Java Card platform; (2) adequacy of bytecode operands to bytecode semantics; (3) absence of operand stack overflow/underflow; (4) control flow confinement to the current method (that is, no control jumps to outside the method); (5) absence of illegal data conversion and reference forging; (6) enforcement of the private/public access modifiers for class and class members; (7) validity of any kind of reference used in the bytecodes (that is, any pointer to a bytecode, class, method, object, local variable, etc actually points to the beginning of piece of data of the expected kind); (8) enforcement of rules for binary compatibility (full details are given in [JCVM222], [JVM], [JCBV]). The actual set of checks performed by the verifier is implementation-dependent, but shall at least enforce all the “must clauses” imposed in [JCVM222] on the bytecodes and the correctness of the CAP files’ format.
As most of the actual Java Card VMs do not perform all the required checks at runtime, mainly because smart cards lack memory and CPU resources, CAP file verification prior to execution is mandatory. On the other hand, there is no requirement on the precise moment when the verification shall actually take place, as far as it can be ensured that the verified file is not modified thereafter. Therefore, the bytecodes can be verified either before the loading of the file on to the card or before the installation of the file in the card or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. This Security Target assumes bytecode verification is performed off-card.

Another important aspect to be considered about bytecode verification and application downloading is, first, the assurance that every package required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM222] §4.4), second, that the export files used to check and link the loaded applet have the corresponding correct counterpart on the card.

4.4.2 Integrity and Authentication

Verification off-card is useless if the application package is modified afterwards. The usage of cryptographic certifications coupled with the verifier in a secure module is a simple means to prevent any attempt of modification between package verification and package installation.

Once a verification authority has verified the package, it signs it and sends it to the card. Prior to the installation of the package, the card verifies the signature of the package, which authenticates the fact that it has been successfully verified. In addition to this, a secured communication channel is used to communicate it to the card, ensuring that no modification has been performed on it.

Alternatively, the card itself may include a verifier and perform the checks prior to the effective installation of the applet or provide means for the bytecodes to be verified dynamically. On-card bytecode verifier is out of the scope of this Security Target.

4.4.3 Linking and Verification

Beyond functional issues, the installer ensures at least a property that matters for security: the loading order shall guarantee that each newly loaded package references only packages that have been already loaded on the card. The linker can ensure this property because the Java Card platform does not support dynamic downloading of classes.

4.5 CARD MANAGEMENT

#.CARD-MANAGEMENT	(1) The card manager (CM) shall control the access to card management functions such as the installation, update or deletion of applets. (2) The card manager shall implement the card issuer's policy on the card.
#.INSTALL	(1) The TOE must be able to return to a safe and consistent state should the installation of a package or an applet fail or be cancelled (whatever the reasons). (2) Installing an applet must have no effect on the code and data of already installed applets. The installation procedure should not be used to bypass the TSFs. In short, it is an atomic operation, free of harmful effects on the state of the other applets. (3) The procedure of loading and installing a package shall ensure its integrity and authenticity.
#.SID	(1) Users and subjects of the TOE must be identified. (2) The identity of sensitive users and subjects associated with administrative and privileged roles must be particularly protected; this concerns the Java Card RE, the applets registered on the card, and especially the default applet and the currently selected applet (and all other active applets in Java Card System 2.2). A change of identity, especially standing for an administrative role (like an applet impersonating the Java Card RE), is a severe violation of the Security Functional Requirements (SFR). Selection controls the access to any data exchange between the TOE and the CAD and therefore, must be protected as well. The loading of a package or any exchange of data through the APDU buffer (which can be accessed by any applet) can lead to disclosure of keys, application code or data, and so on.
#OBJ-DELETION	(1) Deallocation of objects should not introduce security holes in the form of references pointing to memory zones that are no longer in use, or have been reused for other purposes. Deletion of collection of objects should not be maliciously used to circumvent the TSFs. (2) Erasure, if deemed successful, shall ensure that the deleted class instance is no longer accessible.
#DELETION	(1) Deletion of installed applets (or packages) should not introduce security holes in the form of broken references to garbage collected code or data, nor should they alter integrity or confidentiality of remaining applets. The deletion procedure should not be maliciously used to bypass the TSFs. (2) Erasure, if deemed successful, shall ensure that any data owned by the deleted applet is no longer accessible (shared objects shall either prevent deletion or be made inaccessible). A deleted applet cannot be selected or receive APDU commands. Package deletion shall make the code of the package no longer available for execution. (3) Power failure or other failures during the process shall be taken into account in the implementation so as to preserve the SFRs. This does not mandate, however, the process to be atomic. For instance, an

interrupted deletion may result in the loss of user data, as long as it does not violate the SFRs.

The deletion procedure and its characteristics (whether deletion is either physical or logical, what happens if the deleted application was the default applet, the order to be observed on the deletion steps) are implementation-dependent. The only commitment is that deletion shall not jeopardize the TOE (or its assets) in case of failure (such as power shortage).

Deletion of a single applet instance and deletion of a whole package are functionally different operations and may obey different security rules. For instance, specific packages can be declared to be undeletable (for instance, the Java Card API packages), or the dependency between installed packages may forbid the deletion (like a package using super classes or super interfaces declared in another package).

4.6 SERVICES

#.ALARM	The TOE shall provide appropriate feedback upon detection of a potential security violation. This particularly concerns the type errors detected by the bytecode verifier, the security exceptions thrown by the Java Card VM, or any other security-related event occurring during the execution of a TSF.
#.OPERATE	(1) The TOE must ensure continued correct operation of its security functions. (2) In case of failure during its operation, the TOE must also return to a well-defined valid state before the next service request.
#.RESOURCES	The TOE controls the availability of resources for the applications and enforces quotas and limitations in order to prevent unauthorized denial of service or malfunction of the TSFs. This concerns both execution (dynamic memory allocation) and installation (static memory allocation) of applications and packages.
#.CIPHER	The TOE shall provide a means to the applications for ciphering sensitive data, for instance, through a programming interface to low-level, highly secure cryptographic services. In particular, those services must support cryptographic algorithms consistent with cryptographic usage policies and standards.
#.KEY-MNGT	The TOE shall provide a means to securely manage cryptographic keys. This includes: (1) Keys shall be generated in accordance with specified cryptographic key generation algorithms and specified cryptographic key sizes, (2) Keys must be distributed in accordance with specified cryptographic key distribution methods, (3) Keys must be initialized before being used, (4) Keys shall be destroyed in accordance with specified cryptographic key destruction methods.
#.PIN-MNGT	The TOE shall provide a means to securely manage PIN objects. This includes: (1) Atomic update of PIN value and try counter, (2) No rollback on the PIN-checking function, (3) Keeping the PIN value (once initialized) secret (for instance, no clear-PIN-reading function), (4) Enhanced protection of PIN's security attributes (state, try counter…) in confidentiality and integrity.
#.SCP	The smart card platform must be secure with respect to the SFRs. Then: (1) After a power loss, RF signal loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state. (2) It does not allow the SFRs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the

Java Card API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System. (3) It provides secure low-level cryptographic processing to the Java Card System. (4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism. (5) It allows the Java Card System to store data in “persistent technology memory” or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low–level control accesses (segmentation fault detection). (6) It safely transmits low–level exceptions to the TOE (arithmetic exceptions, checksum errors), when applicable. Finally, it is required that (7) the IC is designed in accordance with a well-defined set of policies and standards (for instance, those specified in [PP-IC-0035]), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

#.TRANSACTION	The TOE must provide a means to execute a set of operations atomically. This mechanism must not jeopardise the execution of the user applications. The transaction status at the beginning of an applet session must be closed (no pending updates).

5 SECURITY PROBLEM DEFINITION

5.1 ASSETS

The assets of the TOE are those defined in [PP-JCS-Open]. The assets of [PP-SC] are studied in [ST-IC].

Assets are security-relevant elements to be directly protected by the TOE. Confidentiality of assets is always intended with respect to un-trusted people or software, as various parties are involved during the first stages of the smart card product life-cycle; details are given in threats hereafter.

Assets may overlap, in the sense that distinct assets may refer (partially or wholly) to the same piece of information or data. For example, a piece of software may be either a piece of source code (one asset) or a piece of compiled code (another asset), and may exist in various formats at different stages of its development (digital supports, printed paper). This separation is motivated by the fact that a threat may concern one form at one stage, but be meaningless for another form at another stage.

The assets to be protected by the TOE are listed below. They are grouped according to whether it is data created by and for the user (User data) or data created by and for the TOE (TSF data). For each asset it is specified the kind of dangers that weigh on it.

5.1.1 User data

D.APP_CODE

The code of the applets 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.APP_KEYs

Cryptographic keys owned by the applets.
To be protected from unauthorized disclosure and modification.

D.PIN

Any end-user's PIN.
To be protected from unauthorized disclosure and modification.

5.1.2 TSF data

D.API_DATA

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

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 Java Card VM, such as, for instance, the frame stack, 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 Java Card RE, like, for instance, the AIDs 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.

5.2 THREATS

This section introduces the threats to the assets against which specific protection within the TOE or its environment is required. The threats are classified in several groups.

5.2.1 Confidentiality

T.CONFID-APPLI-DATA
The attacker executes an application to disclose data belonging to another application. See #.CONFID-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_C_DATA, D.PIN,** and **D.APP_KEYs**.

T.CONFID-JCS-CODE
The attacker executes an application to disclose the Java Card System code. See #.CONFID-JCS-CODE for details.
Directly threatened asset(s): **D.JCS_CODE**.

T.CONFID-JCS-DATA
The attacker executes an application to disclose data belonging to the Java Card System. See #.CONFID-JCS-DATA for details.
Directly threatened asset(s): **D.API_DATA, D.SEC_DATA, D.JCS_DATA,** and **D.CRYPTO**.

5.2.2 Integrity

T.INTEG-APPLI-CODE
The attacker executes an application to alter (part of) its own code or another application's code. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**

T.INTEG-APPLI-CODE.LOAD
The attacker modifies (part of) its own or another application code when an application package is transmitted to the card for installation. See #.INTEG-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.INTEG-APPLI-DATA
The attacker executes an application to alter (part of) another application's data. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_I_DATA, D.PIN,** and **D.APP_KEYs**.

T.INTEG-APPLI-DATA.LOAD
The attacker modifies (part of) the initialization data contained in an application package when the package is transmitted to the card for installation. See #.INTEG-APPLI-DATA for details.
Directly threatened asset(s): **D.APP_I_DATA** and **D_APP_KEYs**.

T.INTEG-JCS-CODE
The attacker executes an application to alter (part of) the Java Card System code. See #.INTEG-JCS-CODE for details.
Directly threatened asset(s): **D.JCS_CODE**.

T.INTEG-JCS-DATA
The attacker executes an application to alter (part of) Java Card System or API data. See #.INTEG-JCS-DATA for details.

Directly threatened asset(s): **D.API_DATA, D.SEC_DATA, D.JCS_DATA,** and **D.CRYPTO**.

Other attacks are in general related to one of the above, and aimed at disclosing or modifying on-card information. Nevertheless, they vary greatly on the employed means and threatened assets, and are thus covered by quite different objectives in the sequel. That is why a more detailed list is given hereafter.

5.2.3 Identity usurpation

T.SID.1

An applet impersonates another application, or even the Java Card RE, in order to gain illegal access to some resources of the card or with respect to the end user or the terminal. See #.SID for details.
Directly threatened asset(s): D.SEC_DATA (other assets may be jeopardized should this attack succeed, for instance, if the identity of the JCRE is usurped), D.PIN and D.APP_KEYs.

T.SID.2

The attacker modifies the TOE's attribution of a privileged role (e.g. default applet and currently selected applet), which allows illegal impersonation of this role. See #.SID for further details.
Directly threatened asset(s): D.SEC_DATA (any other asset may be jeopardized should this attack succeed, depending on whose identity was forged).

5.2.4 Unauthorized execution

T.EXE-CODE.1

An applet performs an unauthorized execution of a method. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.EXE-CODE.2

An applet performs an execution of a method fragment or arbitrary data. See #.EXE-JCS-CODE and #.EXE-APPLI-CODE for details.
Directly threatened asset(s): **D.APP_CODE**.

T.NATIVE

An applet executes a native method to bypass a security function such as the firewall. See #.NATIVE for details.
Directly threatened asset(s): **D.JCS_DATA**.

5.2.5 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. See #.RESOURCES for details.
Directly threatened asset(s): **D.JCS_DATA**.

5.2.6 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). See #.DELETION (p 343) for details).
Directly threatened asset(s): **D.SEC_DATA** and **D.APP_CODE**.

T.INSTALL

The attacker fraudulently installs post-issuance of 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. See #.INSTALL for details.
Directly threatened asset(s): **D.SEC_DATA** (any other asset may be jeopardized should this attack succeed, depending on the virulence of the installed application).

5.2.7 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. See #.OBJ-DELETION for further details.
Directly threatened asset(s): **D.APP_C_DATA, D.APP_I_DATA** and **D.APP_KEYs**.

5.2.8 Miscellaneous

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, and DPA. 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.
This threatens all the identified assets.
This threat refers to the point (7) of the security aspect #.SCP, and all aspects related to confidentiality and integrity of code and data.

5.3 ORGANIZATIONAL SECURITY POLICIES

5.3.1 Java Card System Protection Profile – Open Configuration

This section describes the organizational security policies to be enforced with respect to the TOE environment.

OSP.VERIFICATION

This policy shall ensure the consistency 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. See #.VERIFICATION for details.
If the application development guidance provided by the platform developer contains recommendations related to the isolation property of the platform, this policy shall also ensure that the verification authority checks that these recommendations are applied in the application code.

5.3.2 TOE additional OSP

OSP.SpecificAPI

The TOE must contribute to ensure that application can optimize control on its sensitive operations using a dedicated API provided by TOE. TOE will provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

OSP.RND

This policy shall ensure the entropy of the random numbers provided by the TOE to applet using [JCAPI304] is sufficient. Thus attacker is not able to predict or obtain information on generated numbers.

5.4 ASSUMPTIONS

This section introduces the assumptions made on the environment of the TOE.

5.4.1 Assumptions extracted from [PP-JCS-Open]

A.APPLET

Applets loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM222], §3.3) outside the API.

A.DELETION

Deletion of applets, if available through the card manager, is secure. Refer to #.DELETION for details on this assumption.

The rationale for this latter assumption is that even a Java Card System 2.1.1 TOE could be installed on a product that includes applet deletion features. This assumes that these functions are secure with respect to the SFRs herein.

A.VERIFICATION

All the bytecodes are verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time.

5.4.2 Additional assumptions

A.SID

Any applet and package are uniquely identified.

A.FIREWALL-ENV

The applet selection/deletion/loading, the object deletion, and the JCRMI ensure the controlled sharing of data.

A.GLOBAL_ARRAYS_CONFID

The APDU buffer that is shared by all applications is always cleaned upon applet selection.
The global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

A.OPERATE-ENV

The applet selection/deletion/loading and the object deletion ensure the continued correct operation of the TOE security functions.

A.REALLOCATION

The re-allocation of a memory block for the runtime areas of the Java Card VM does not disclose any information that was previously stored in that block.

A.RESOURCES

The availability of resources for the applications is controlled. See #.RESOURCES for details.

A.ALARM-ENV

The applet deletion/loading and the object deletion ensure the appropriate feedback information upon detection of a potential security violation. See #.ALARM for details.

A.CIPHER

Sensitive data for applications are ciphered in a secure way. See #.CIPHER for details.

A.KEY-MNGT

Cryptographic means are securely managed. This concerns the correct generation, distribution, access and destruction of cryptographic keys. See #.KEY-MNGT.

A.PIN-MNGT

PIN objects are securely managed. See #.PIN-MNGT for details.

A.TRANSACTION

A set of operations are executed atomically. See #.TRANSACTION for details.

A.OBJ-DELETION

The object deletion shall not break references to objects. See #.OBJ-DELETION for further details.

A.DELETION

Both applet and package deletion shall be performed as expected. (See #.DELETION for details).

A.LOAD

The loading of a package into the card is safe. For codes loaded post-issuance, the TOE verifies the integrity and authenticity evidences generated during the verification of the application package by the verification authority. This verification by the TOE occurs during the load or late during the install process.

A.INSTALL-ENV
The applet deletion/loading and the object deletion the installation of an applet to be performed as expected.

5.5 COMPATIBILITY WITH SECURITY ENVIRONMENTS [ST-IC]

5.5.1 Compatibility between threats

T.CONFID-JCS-CODE, T.CONFID-APPLI-DATA, and T.CONFID-JCS-DATA are included in T.Phys-Probing, T.Leak-Inherent and T.Leak-Forced.

T.SID.2 is partly included in T.Phys-Manipulation and T.Malfunction.

T.PHYSICAL is included in T.Phys-Probing, T.Leak-Inherent, T.Phys-Manipulation, T.Malfunction and T.Leak-Forced.

T.INTEG-APPLI-CODE T.INTEG-JCS-CODE T.INTEG-APPLI-DATA DATA T.INTEG-JCS-DATA T.INTEG-APPLI-CODE.LOAD T.INTEG-APPLI-DATA.LOAD T.SID.1 T.EXE-CODE.1 T.EXE-CODE.2 T.NATIVE T.RESOURCES T.INSTALL T.DELETION T.OBJ-DELETION are threats specific to the Java Card platform and they do no conflict with the threats of [ST-IC].

5.5.2 Compatibility between OSP

OSP.VERIFICATION is an OSP specific to the Java Card platform and it does no conflict with the OSP of [ST-IC].

OSP.SpecificAPI has been added to this ST in order to manage Specific API and it does no conflict with the OSP of [ST-IC].

OSP.RND has been added to this ST in order to manage RNG and it does no conflict with the OSP of [ST-IC].

We can therefore conclude that the OSP for the environment of this ST and [ST-IC] are consistent.

5.5.3 Compatibility between assumptions

A.VERIFICATION, A.DELETION, and A.APPLET are assumptions specific to the Java Card platform and they do no conflict with the assumptions of [ST-IC].

A.SID, A.FIREWALL-ENV, A.GLOBAL_ARRAYS_CONFID, A.OPERATE-ENV, A.REALLOCATION, A.RESOURCES, A.ALARM-ENV, A.CIPHER, A.KEY-MNGT, A.PIN-MNGT, A.TRANSACTION, A.OBJ-DELETION, A.DELETION, A.LOAD and A.INSTALL-ENV are assumptions added in order to manage the corresponding OSP.

We can therefore conclude that the assumptions for the environment of this ST and [ST-IC] are consistent.

Regarding the assumptions of [ST-IC], the TOE is part of the JCS defined in [ST_MultiAppV4] which is composed with the IC defined in [ST-IC]. The compatibility is hence justified in [ST_MultiAppV4].

6 SECURITY OBJECTIVES

6.1 SECURITY OBJECTIVES FOR THE TOE

This section defines the security objectives to be achieved by the TOE.

6.1.1 Execution

O.FIREWALL

The TOE shall ensure controlled sharing of data containers owned by applets of different packages, or the JCRE and between applets and the TSFs. See #.FIREWALL for details.

Application note:

With respect to [PP-JCS-Open], this objective is for the components of the JCS included in this TOE, i.e. interpreter, linker and Native API. This objective is not for the following out-of-TOE components of the JCS:

- Applet selection/deletion/loading
- Object deletion
- JCRMI

that are part of the TOE environment.

O.GLOBAL_ARRAYS_INTEG

The TOE shall ensure that only the currently selected applications may have a write access to the APDU buffer and the global byte array used for the invocation of the install method of the selected applet.

O.NATIVE

The only means that the Java Card VM shall provide for an application to execute native code is the invocation of a method of the Java Card API, or any additional API. See #.NATIVE for details.

O.OPERATE

The card manager must ensure continued correct operation of its security functions. See #.OPERATE for details.

Application note:

With respect to [PP-JCS-Open], this objective is for the components of the JCS included in this TOE, i.e. interpreter, linker and Native API. This objective is not for the following out-of-TOE components of the JCS:

- Applet selection/deletion/loading
- Object deletion

that are part of the TOE environment.

6.1.2 Services

O.ALARM

The TOE shall provide appropriate feedback information upon detection of a potential security violation. See #.ALARM for details.

Application note:

With respect to [PP-JCS], this objective is for the components of the JCS included in this TOE, i.e. interpreter, linker and Native API. This objective is not for the following out-of-TOE components of the JCS:

- Applet selection/deletion/loading
- Object deletion

that are part of the TOE environment.

6.1.3 Applet Management

O.INSTALL

The TOE shall ensure that the installation of an applet performs as expected. See #.INSTALL (in [PP-JCS]) for details.

Application note:

With respect to [PP-JCS], this objective is for the components of the JCS included in this TOE, i.e. interpreter, linker and Native API. This objective is not for the following out-of-TOE components of the JCS:

- Applet selection/deletion/loading
- Object deletion

that are part of the TOE environment.

6.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

6.2.1 Objectives for the operational environment extracted from [PP-JCS-Open]

This section introduces the security objectives to be achieved by the environment and extracted from [PP-JCS-Open].

OE.VERIFICATION

All the bytecodes shall be verified at least once, before the loading, before the installation or before the execution, depending on the card capabilities, in order to ensure that each bytecode is valid at execution time. See #.VERIFICATION for details.

Additionally the applet shall follow all recommendations, if any, mandated in the platform guidance for maintaining the isolation property of the platform.

Application Note:

Constraints to maintain the isolation property of the platform are provided by the platform developer in application development guidance. The constraints apply to all application code loaded in the platform.

OE.APPLET

No applet loaded post-issuance shall contain native methods.

OE.CODE-EVIDENCE

For application code loaded pre-issuance, evaluated technical measures implemented by the TOE or audited organizational measures must ensure that loaded application has not been changed since the code verifications required in OE.VERIFICATION.

For application code loaded post-issuance and verified off-card according to the requirements of OE.VERIFICATION, the verification authority shall provide digital evidence to the TOE that the application code has not been modified after the code verification and that he is the actor who performed code verification.

For application code loaded post-issuance and partially or entirely verified on-card, technical measures must ensure that the verification required in OE.VERIFICATION are performed. On-card bytecode verifier is out of the scope of this Protection Profile.

Application Note:

For application code loaded post-issuance and verified off-card, the integrity and authenticity evidence can be achieved by electronic signature of the application code, after code verification, by the actor who performed verification.

OE.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.

This security objective for the environment refers to the security aspect #.SCP.1: The smart card platform must be secure with respect to the SFRs. Then after a power loss or sudden card removal prior to completion of some communication protocol, the SCP will allow the TOE on the next power up to either complete the interrupted operation or revert to a secure state.

OE.SCP.SUPPORT

This security objective for the environment refers to the security aspect #.SCP.2-5:

(2) It does not allow the TSFs to be bypassed or altered and does not allow access to other low-level functions than those made available by the packages of the API. That includes the protection of its private data and code (against disclosure or modification) from the Java Card System.
(3) It provides secure low-level cryptographic processing to the Java Card System.
(4) It supports the needs for any update to a single persistent object or class field to be atomic, and possibly a low-level transaction mechanism.
(5) It allows the Java Card System to store data in "persistent technology memory" or in volatile memory, depending on its needs (for instance, transient objects must not be stored in non-volatile memory). The memory model is structured and allows for low-level control accesses (segmentation fault detection).

OE.SCP.IC

The SCP shall provide all IC security features against physical attacks.
This security objective for the environment refers to the point (7) of the security aspect #.SCP:
It is required that the IC is designed in accordance with a well-defined set of policies and Standards (likely specified in another protection profile), and will be tamper resistant to actually prevent an attacker from extracting or altering security data (like cryptographic keys) by using commonly employed techniques (physical probing and sophisticated analysis of the chip). This especially matters to the management (storage and operation) of cryptographic keys.

OE.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.
The card manager is an application with specific rights, which is responsible for the administration of the smart card. This component will in practice be tightly connected with the TOE, which in turn shall very likely rely on the card manager for the effective enforcing of some of its security functions. Typically the card manager shall be in charge of the life cycle of the whole card, as well as that of the installed applications (applets). 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.

6.2.2 Additional security objectives for the operational environment

6.2.2.1 Identification

OE.SID

The Applet selection component shall uniquely identify every subject (applet, or package) before granting it access to any service.

6.2.2.2 Execution

OE.FIREWALL-ENV

The controlled sharing of data containers owned by applets of different packages, or the JCRE and between applets and the TSFs shall be ensured by the following components:

- Applet selection/deletion/loading
- Object deletion
- JCRMI

OE.GLOBAL_ARRAYS_CONFID

The Applet selection/deselection component shall ensure that the APDU buffer that is shared by all applications is always cleaned upon applet selection.
The TOE shall ensure that the global byte array used for the invocation of the install method of the selected applet is always cleaned after the return from the install method.

OE.OPERATE-ENV

The continued correct operation of the TOE security functions shall be ensured in the following (out-of-TOE) components:

- Applet selection/deletion/loading
- Object deletion

OE.REALLOCATION

The corresponding part of the JCS shall ensure that the re-allocation of a memory block for the runtime areas of the Java Card VM 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 [JCVM222].

OE.RESOURCES

The JCS shall control the availability of resources for the applications. See #.RESOURCES for details.

6.2.2.3 Services

OE.ALARM-ENV

The appropriate feedback information upon detection of a potential security violation shall be provided in the following (out-of-TOE) components:

- Applet deletion/loading
- Object deletion

OE.CIPHER

The cryptographic API shall provide a 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. See #.CIPHER for details.

OE.KEY-MNGT

The cryptographic API shall provide a means to securely manage cryptographic keys. This concerns the correct generation, distribution, access and destruction of cryptographic keys. See #.KEY-MNGT.

Application note:
OE.KEY-MNGT, OE.PIN-MNGT, OE.TRANSACTION and OE.CIPHER are actually provided to applets in the form of Java Card APIs. Vendor-specific libraries can also be present on the card and made available to applets; those may be built on top of the Java Card API or independently. Depending on whether they contain native code or not, these proprietary libraries will need to be evaluated together with the TOE or not (see #.NATIVE). In any case, they are not included in the Java Card System for the purpose of the present document.

OE.PIN-MNGT

The JCS shall provide a means to securely manage PIN objects. See #.PIN-MNGT for details.
Application note:
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 sensitive as that of the PIN.

OE.TRANSACTION

The JCS must provide a means to execute a set of operations atomically. See #.TRANSACTION for details.

6.2.2.4 Object deletion

OE.OBJ-DELETION

The Object Deletion component shall ensure the object deletion shall not break references to objects. See #.OBJ-DELETION for further details.

6.2.2.5 Applet management

OE.DELETION

The Applet deletion component shall ensure that both applet and package deletion perform as expected. (See #.DELETION for details).

OE.LOAD

The Applet loading component shall ensure that the loading of a package into the card is safe.

Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application package by the verification authority. This verification by the TOE shall occur during the load or late during the install process.

OE.INSTALL-ENV

The installation of an applet shall be performed as expected in the following (out-of-TOE) components:

- Applet deletion/loading
- Object deletion

6.2.2.6 Additional objectives

OE.SpecificAPI

The JCS shall provide to application a specific API means to optimize control on sensitive operations performed by application.
JCS shall provide services for secure array management and to detect loss of data integrity and inconsistent execution flow and react against tearing or fault induction.

OE.RND

The JCS must contribute to ensure that random numbers shall not be predictable and shall have sufficient entropy.

6.3 SECURITY OBJECTIVES RATIONALE

	O.FIREWALL	O.GLOBAL_ARRAYS_INTEG	O.NATIVE	O.OPERATE	O.ALARM	O.INSTALL	OE.VERIFICATION	OE.APPLET	OE;CODE_EVIDENCE	OE.SCP.RECOVERY	OE.SCP.SUPPORT	OE.SCP.IC	OE.CARD_MANAGEMENT	OE.SID	OE.FIREWALL-ENV	OE.GLOBAL_ARRAYS_CONFID	OE.OPERATE-ENV	OE.REALLOCATION	OE.RESOURCES	OE.ALARM-ENV	OE.CIPHER	OE.KEY-MNGT	OE.PIN-MNGT	OE.TRANSACTION	OE.OBJ-DELETION	OE.DELETION	OE.LOAD	OE.INSTALL-ENV	OE.SpecificAPI	OE.RND
T.CONFID-JCS-CODE			X				X						X																	
T.CONFID-APPLI-DATA	X			X	X		X			X	X		X	X		X		X			X	X	X	X						
T.CONFID-JCS-DATA	X			X	X		X			X	X		X	X																
T.INTEG-APPLI-CODE			X				X		X				X																	
T.INTEG-JCS-CODE			X				X		X				X																	
T.INTEG-APPLI-DATA	X	X		X	X		X		X	X	X		X	X				X	X			X	X	X						
T.INTEG-JCS-DATA	X			X	X		X		X	X	X		X	X																
T.INTEG-APPLI-CODE.LOAD									X				X														X			
T.INTEG-APPLI-DATA.LOAD									X				X														X			
T.SID.1	X	X				X							X	X		X														
T.SID.2	X			X		X				X	X			X																
T.EXE-CODE.1	X						X																							
T.EXE-CODE.2							X																							
T.NATIVE			X				X	X																						
T.RESOURCES				X		X				X	X								X											
T.INSTALL						X							X														X			
T.DELETION													X													X				

	O.FIREWALL	O.GLOBAL_ARRAYS_INTEG	O.NATIVE	O.OPERATE	O.ALARM	O.INSTALL	OE.VERIFICATION	OE.APPLET	OE;CODE_EVIDENCE	OE.SCP.RECOVERY	OE.SCP.SUPPORT	OE.SCP.IC	OE.CARD_MANAGEMENT	OE.SID	OE.FIREWALL-ENV	OE.GLOBAL_ARRAYS_CONFID	OE.OPERATE-ENV	OE.REALLOCATION	OE.RESOURCES	OE.ALARM-ENV	OE.CIPHER	OE.KEY-MNGT	OE.PIN-MNGT	OE.TRANSACTION	OE.OBJ-DELETION	OE.DELETION	OE.LOAD	OE.INSTALL-ENV	OE.SpecificAPI	OE.RND
T.OBJ-DELETION																									X					
T.PHYSICAL												X																		
OSP.VERIFICATION							X																				X			
OSP.SpecificAPI																												X		
OSP.RND																														X
A.APPLET								X																						
A.DELETION													X																	
A.VERIFICATION							X		X																					
A.SID														X																
A.FIREWALL-ENV															X															
A.GLOBAL_ARRAYS_CONFID																X														
A.OPERATE-ENV																	X													
A.REALLOCATION																		X												
A.RESOURCES																			X											
A.ALARM-ENV																				X										
A.CIPHER																					X									
A.KEY-MNGT																						X								
A.PIN-MNGT																							X							
A.TRANSACTION																								X						
A.OBJ-DELETION																									X					
A.DELETION																										X				
A.LOAD																											X			
A.INSTALL-ENV																												X		

6.3.1 Threats

6.3.1.1 Confidentiality

T.CONFID-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of those instructions enables reading a piece of code, no Java Card applet can therefore be executed to disclose a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to disclose a piece of code.
The (#.VERIFICATION) security aspect is addressed in this ST by the objective for the environment OE.VERIFICATION.
The environmental objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.

T.CONFID-APPLI-DATA This threat is countered by the security objective for the operational environment regarding bytecode verification (OE.VERIFICATION). It is also covered by the isolation commitments stated in the (O.FIREWALL) objective. It relies in its turn on the correct identification of applets stated in (OE.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The environmental objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The environmental objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
As applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (OE.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys, PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (OE.KEY-MNGT, OE.PIN-MNGT, OE.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) shall contribute in covering this threat by controlling the sharing of the global PIN between the applets.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The disclosure of such data is prevented by the (OE.GLOBAL_ARRAYS_CONFID) environmental security objective.
Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the OE.REALLOCATION environmental objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.CONFID-JCS-DATA This threat is covered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) security objective. This latter objective also relies in its turn on the correct identification of applets stated in (OE.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The environmental objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The environmental objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

6.3.1.2 Integrity

T.INTEG-APPLI-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the environmental objective (O.NATIVE), so no application can be run to modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.
The environmental objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that integrity and authenticity evidences exist for the application code loaded into the platform.

T.INTEG-JCS-CODE This threat is countered by the list of properties described in the (#.VERIFICATION) security aspect. Bytecode verification ensures that each of the instructions used on the Java Card platform is used for its intended purpose and in the intended scope of accessibility. As none of these instructions enables modifying a piece of code, no Java Card applet can therefore be executed to modify a piece of code. Native applications are also harmless because of the objective (O.NATIVE), so no application can be run to disclose or modify a piece of code.
The (#.VERIFICATION) security aspect is addressed in this configuration by the objective for the environment OE.VERIFICATION.

The objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.

T.INTEG-APPLI-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (OE.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. The objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
Concerning the confidentiality and integrity of application sensitive data, as applets may need to share some data or communicate with the CAD, cryptographic functions are required to actually protect the exchanged information (OE.CIPHER). Remark that even if the TOE shall provide access to the appropriate TSFs, it is still the responsibility of the applets to use them. Keys and PIN's are particular cases of an application's sensitive data (the Java Card System may possess keys as well) that ask for appropriate management (OE.KEY-MNGT, OE.PIN-MNGT, OE.TRANSACTION). If the PIN class of the Java Card API is used, the objective (O.FIREWALL) is also concerned.
Other application data that is sent to the applet as clear text arrives to the APDU buffer, which is a resource shared by all applications. The integrity of the information stored in that buffer is ensured by the (OE.GLOBAL_ARRAYS_INTEG) objective.
Finally, any attempt to read a piece of information that was previously used by an application but has been logically deleted is countered by the OE.REALLOCATION objective. That objective states that any information that was formerly stored in a memory block shall be cleared before the block is reused.

T.INTEG-JCS-DATA This threat is countered by bytecode verification (OE.VERIFICATION) and the isolation commitments stated in the (O.FIREWALL) objective. This latter objective also relies in its turn on the correct identification of applets stated in (OE.SID). Moreover, as the firewall is dynamically enforced, it shall never stop operating, as stated in the (O.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective O.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives OE.CARD_MANAGEMENT and OE.VERIFICATION contribute to cover this threat by controlling the access to card management functions and by checking the bytecode, respectively.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity.
The objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE and O.ALARM objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

T.INTEG-APPLI-CODE.LOAD This threat is countered by the security objective OE.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of packages code.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity. By controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD_MANAGEMENT contributes to cover this threat.

T.INTEG-APPLI-DATA.LOAD This threat is countered by the security objective OE.LOAD which ensures that the loading of packages is done securely and thus preserves the integrity of applications data.
The objective OE.CODE-EVIDENCE contributes to cover this threat by ensuring that the application code loaded into the platform has not been changed after code verification, which ensures code integrity and authenticity by controlling the access to card management functions such as the installation, update or deletion of applets the objective OE.CARD_MANAGEMENT contributes to cover this threat.

6.3.1.3 *Identity usurpation*

T.SID.1 As impersonation is usually the result of successfully disclosing and modifying some assets, this threat is mainly countered by the objectives concerning the isolation of application data (like PINs), ensured by the (O.FIREWALL). Uniqueness of subject-identity (OE.SID) also participates to face this threat. It should be noticed that the AIDs, which are used for applet identification, are TSF data.
In this configuration, usurpation of identity resulting from a malicious installation of an applet on the card is covered by the objective O.INSTALL.
The installation parameters of an applet (like its name) are loaded into a global array that is also shared by all the applications. The disclosure of those parameters (which could be used to impersonate the applet) is countered by the objective (OE.GLOBAL_ARRAYS_CONFID) and (O.GLOBAL_ARRAYS_INTEG).
The objective OE.CARD_MANAGEMENT contributes, by preventing usurpation of identity resulting from a malicious installation of an applet on the card, to counter this threat.

T.SID.2 This is covered by integrity of TSF data, subject-identification (OE.SID), the firewall (O.FIREWALL) and its good working order (O.OPERATE).
The objective O.INSTALL contributes to counter this threat by ensuring that installing an applet has no effect on the state of other applets and thus can't change the TOE's attribution of privileged roles.
The objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE objective of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

6.3.1.4 *Unauthorized execution*

T.EXE-CODE.1 Unauthorized execution of a method is prevented by the objective OE.VERIFICATION. This threat particularly concerns the point (8) of the security aspect #VERIFICATION (access modifiers and scope of accessibility for classes, fields and methods). The O.FIREWALL objective is also concerned, because it prevents the execution of non-shareable methods of a class instance by any subject apart from the class instance owner.

T.EXE-CODE.2 Unauthorized execution of a method fragment or arbitrary data is prevented by the objective OE.VERIFICATION. This threat particularly concerns those points of the security aspect related to control flow confinement and the validity of the method references used in the bytecodes.

T.NATIVE This threat is countered by O.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through an API. OE.APPLET also covers this threat by ensuring that no native applets shall be loaded in post-issuance. In addition to this, the bytecode verifier also prevents the program counter of an applet to jump into a piece of native code by confining the control flow to the currently executed method (OE.VERIFICATION).

6.3.1.5 *Denial of service*

T.RESOURCES This threat is directly countered by objectives on resource-management (OE.RESOURCES) for runtime purposes and good working order (O.OPERATE) in a general manner.
Consumption of resources during installation and other card management operations are covered, in case of failure, by O.INSTALL.
It should be noticed that, for what relates to CPU usage, the Java Card platform is single-threaded and it is possible for an ill-formed application (either native or not) to monopolize the CPU. However, a smart card can be physically interrupted (card removal or hardware reset) and most CADs implement a timeout policy that prevent them from being blocked should a card fails to answer. That point is out of scope of this Security Target, though.
Finally, the objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the O.OPERATE and OE.RESOURCES objectives of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

6.3.1.6 *Card management*

T.INSTALL This threat is covered by the security objective O.INSTALL which ensures that the installation of an applet performs as expected and the security objectives OE.LOAD which ensures that the loading of a package into the card is safe.
The objective OE.CARD_MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

T.DELETION This threat is covered by the OE.DELETION security objective which ensures that both applet and package deletion perform as expected.
The objective OE.CARD_MANAGEMENT controls the access to card management functions and thus contributes to cover this threat.

6.3.1.7 Services

T.OBJ-DELETION This threat is covered by the OE.OBJ-DELETION security objective which ensures that object deletion shall not break references to objects.

6.3.1.8 Miscellaneous

T.PHYSICAL Covered by OE.SCP.IC. Physical protections rely on the underlying platform and are therefore an environmental issue.

6.3.2 Organizational Security Policies

6.3.2.1 Java Card System Protection Profile – Open Configuration

OSP.VERIFICATION This policy is upheld by the security objective of the environment OE.VERIFICATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.

This policy is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

6.3.2.2 Additional

OSP.SpecificAPI This OSP is enforced by the TOE security objective OE.SpecificAPI.

OSP.RND This OSP is enforced by the TOE security objective OE.RND.

6.3.3 Assumptions

6.3.3.1 Java Card System Protection Profile – Open Configuration

A.APPLET This assumption is upheld by the security objective for the operational environment OE.APPLET which ensures that no applet loaded post-issuance shall contain native methods.

A.DELETION This assumption is upheld by the environmental objective OE.CARD_MANAGEMENT which controls the access to card management functions such as deletion of applets.

A.VERIFICATION This assumption is upheld by the security objective on the operational environment OE.VERIFICATION which guarantees that all the bytecodes shall be verified at least once, before the loading, before the installation or before the execution in order to ensure that each bytecode is valid at execution time.
This assumption is also upheld by the security objective of the environment OE.CODE-EVIDENCE which ensures that evidences exist that the application code has been verified and not changed after verification.

6.3.3.2 Additional

A.SID This assumption is upheld by the environmental objective OE.SID.

A.FIREWALL-ENV This assumption is upheld by OE.FIREWALL-ENV.

A.GLOBAL_ARRAYS_CONFID This assumption is upheld by the environmental objective OE.GLOBAL_ARRAYS_CONFID.

A.OPERATE-ENV This assumption is upheld by the environmental objective OE.OPERATE-ENV.

A.REALLOCATION This assumption is upheld by the environmental objective OE.REALLOCATION.

A.RESOURCES This assumption is upheld by the environmental objective OE.RESOURCES.

A.ALARM-ENV This assumption is upheld by the environmental objective OE.ALARM-ENV.

A.CIPHER This assumption is upheld by the environmental objective OE.CIPHER.

A.KEY-MNGT This assumption is upheld by the environmental objective OE.KEY-MNGT.

A.PIN-MNGT This assumption is upheld by the environmental objective OE.PIN-MNGT.

A.TRANSACTION This assumption is upheld by the environmental objective OE.TRANSACTION.

A.OBJ-DELETION This assumption is upheld by the environmental objective OE.OBJ-DELETION.

A.DELETION This assumption is upheld by the environmental objective OE.DELETION.

A.LOAD This assumption is upheld by the environmental objective OE.LOAD.

A.INSTALL-ENV This assumption is upheld by the environmental objective OE.INSTALL-ENV.

6.3.4 Compatibility with the objectives of [ST-IC]

6.3.4.1 Compatibility between objectives for the TOE

O.SID, O.OPERATE, O.RESOURCES, O.FIREWALL, O.NATIVE, O.REALLOCATION, O.GLOBAL_ARRAYS_CONFID, O.GLOBAL_ARRAYS_INTEG, O.ALARM; O.TRANSACTION, O.PIN-MNGT, O.KEY-MNGT, O.OBJ-DELETION, O.INSTALL, O.LOAD, O.DELETION, O.CARD-MANAGEMENT, and O.SCP.RECOVERY are objectives specific to the Java Card platform and they do no conflict with the objectives of [ST-IC].
O.SpecificAPI is objective added to this platform it does no conflict with the objectives of [ST-IC].
O.RND added to this platform is included in the following objectives of [ST-IC]: O.RND
O.CIPHER is included in the following objectives of [ST-IC]: O.RND and O.Add-Functions.
O.SCP.SUPPORT is partially included in the following objectives of [ST-IC]: O.RND and O.Add-Functions.
O.SCP.IC is included in the following objectives of [ST-IC]: O.Phys-Manipulation, O.Phys-Probing, O.Malfunction O.Leak-Inherent O.Leak-Forced O.Abuse-Func.

We can therefore conclude that the objectives for the TOE of this ST and [ST-IC] are consistent.

6.3.4.2 Compatibility between objectives for the environment

OE.VERIFICATION, OE.CODE-EVIDENCE and OE.Applet are objectives specific to the Java Card platform and they do no conflict with the objectives of [ST-IC].
We can therefore conclude that the objectives for the environment of this ST and [ST-IC] are consistent

7 SECURITY REQUIREMENTS

7.1 SECURITY FUNCTIONAL REQUIREMENTS

This section states the security functional requirements for the TOE.

Group	Description
Core with Logical Channels (CoreG_LC)	The CoreG_LC contains the requirements concerning the runtime environment of the Java Card System implementing logical channels. This includes the firewall policy and the requirements related to the Java Card API. Logical channels are a Java Card specification version 2.2 feature. This group is the union of requirements from the Core (CoreG) and the Logical channels (LCG) groups defined in [PP/0305]. (cf Java Card Sysytem Protection Profile Collection [PP JCS]).
Installation (InstG)	The InstG contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution.

The SFRs refer to all potentially applicable subjects, objects, information, operations and security attributes.

Subjects are active components of the TOE that (essentially) act on the behalf of users. The users of the TOE include people or institutions (like the applet developer, the card issuer, the verification authority), hardware (like the CAD where the card is inserted or the PCD) and software components (like the application packages installed on the card). Some of the users may just be aliases for other users. For instance, the verification authority in charge of the bytecode verification of the applications may be just an alias for the card issuer.
Subjects (prefixed with an "S") are described in the following table:

Subject	Description
S.JCRE	
S.JCVM	The bytecode interpreter that enforces the firewall at runtime.
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.
S.PACKAGE	A package is a namespace within the Java programming language that may contain classes and interfaces, and in the context of Java Card technology, it defines either a user library, or one or several applets.
S.APPLET	Any applet instance.

Objects (prefixed with an "O") are described in the following table:

Object	Description
O.APPLET	Any installed applet, its code and data.
O.JAVAOBJECT	Java class instance or array. It should be noticed that KEYS, PIN, arrays and applet instances are specific objects in the Java programming language.

Information (prefixed with an "I") is described in the following table:

Information	Description
I.DATA	JCVM Reference Data: objectref addresses of APDU buffer, JCRE-owned instances of APDU class and byte array for install method

Security attributes linked to these subjects, objects and information are described in the following table with their values (used in enforcing the SFRs):

Security attribute	Description/Value
Active Applets	The set of the active applets' AIDs. An active applet is an applet that is selected on at least one of the logical channels.

Security attribute	Description/Value
Applet Selection Status	"Selected" or "Deselected"
Applet's version number	The version number of an applet (package) indicated in the export file
Class	Identifies the implementation class of the remote object.
Context	Package AID, or "Java Card RE"
Currently Active Context	Package AID, or "Java Card RE"
Dependent package AID	Allows the retrieval of the Package AID and Applet's version number ([JCVM222], §4.5.2).
ExportedInfo	Boolean (Indicates whether the remote object is exportable or not).
Identifier	The Identifier of a remote object or method is a number that uniquely identifies a remote object or method, respectively.
LC Selection Status	Multiselectable, Non-multiselectable or "None".
LifeTime	CLEAR_ON_DESELECT or PERSISTENT (*).
Owner	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). The owner of a remote object is the applet instance that created the object.
Package AID	The AID of each package indicated in the export file
Registered applets	The set of AID of the applet instance registered on the card
ResidentPackages	The set of AIDs of the packages already loaded on the card
Selected Applet Context	Package AID, or "None"
Sharing	Standards, SIO, Java Card RE entry point, or global array
Static References	Static fields of a package may contain references to objects. The Static References attribute records those references.

(*) Transient objects of type CLEAR_ON_RESET behave like persistent objects in that they can be accessed only when the Currently Active Context is the object's context.
Operations (prefixed with "OP") 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.JAVA(...)	Any access in the sense of [JCRE222], §6.2.8. It stands for one of the operations OP.ARRAY_ACCESS, OP.INSTANCE_FIELD, OP.INVK_VIRTUAL, OP.INVK_INTERFACE, OP.THROW, OP.TYPE_ACCESS.
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.
OP.THROW(O.JAVAOBJECT)	Throwing of an object (athrow, see [JCRE222],§6.2.8.7)
OP.TYPE_ACCESS(O.JAVAOBJECT, class)	Invoke checkcast or instanceof on an object in order to access to classes (standard or shareable interfaces objects).

Operation	Description
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient call).
OP.PUT(S1,S2,I)	Transfer a piece of information I from S1 to S2.

(*) For this operation, there is no accessed object. This rule enforces that shareable transient objects are not allowed. For instance, during the creation of an object, the JavaCardClass attribute's value is chosen by the creator.

7.1.1 CoreG_LC Security Functional Requirements

This group is focused on the main security policy of the Java Card System, known as the firewall. This policy essentially concerns the security of installed applets. The policy focuses on the execution of bytecodes.

7.1.1.1 Firewall Policy

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, S.JCVM, O.JAVAOBJECT** and all operations among subjects and objects covered by the SFP.

Refinement:
The operations involved in the policy are:
- OP.CREATE,
- OP.INVK_INTERFACE,
- OP.INVK_VIRTUAL,
- OP.JAVA,
- OP.THROW,
- OP.TYPE_ACCESS.

FDP_ACC.2.2/FIREWALL The TSF shall ensure that all operations between any subject controlled by the TSF and any object controlled by the TSF are covered by an access control SFP.
Application note:
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_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:

Subject/Object	Attributes
S.PACKAGE	LC Applet Selection Status
S.JCVM	ActiveApplets, Currently Active Context
S.JCRE	Selected Applet Context
O.JAVAOBJECT	Sharing, Context, LifeTime

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:

- R.JAVA.1 ([JCRE222], §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 ([JCRE222], §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 ([JCRE222]§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 ([JCRE222], §6.2.8.6,) An S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if the invoked interface method extends the Shareable interface and one of the following applies:

  (a) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable»,

  (b) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable», and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets.
- R.JAVA.5 An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".

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

1) **The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the exception given in FDP_ACF.1.4/FIREWALL, provided it is the Currently Active Context.**
2) **The only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method (through OP.INVK_INTERFACE or OP.INVK_VIRTUAL).**

FDP_ACF.1.4/FIREWALL The TSF shall explicitly deny access of subjects to objects based on the following additional 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 attempting to create an object by the means of OP.CREATE and a "CLEAR_ON_DESELECT" LifeTime parameter if the active context is not the same as the Selected Applet Context**.

Application note:
In the case of an array type, 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 ([JCRE222], §6.1.3). 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).

([JCRE222], Glossary) Selected Applet Context. The Java Card RE keeps track of the currently selected Java Card applet. Upon receiving a SELECT command with this applet's AID, the Java Card RE makes

this applet the Selected Applet Context. The Java Card RE sends all APDU commands to the Selected Applet Context.
While the expression "Selected Applet Context" refers to a specific installed applet, the relevant aspect to the policy is the context (package AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected package.
([JCRE222], §6.1.2.1) At any point in time, there is only one active context within the Java Card VM (this is called the Currently Active Context).

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 to in this case.

The Java Card platform, version 2.2.x introduces the possibility for an applet instance to be selected on multiple logical channels at the same time, or accepting other applets belonging to the same package being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same package are either all multiselectable or not ([JCVM222], §2.2.5). Therefore, the selection mode can be regarded as an attribute of packages. No selection mode is defined for a library package.

An applet instance will be considered an active applet instance if it is currently selected in at least one logical channel. An applet instance is the currently selected applet instance only if it is processing the current command. There can only be one currently selected applet instance at a given time. ([JCRE222], §4).

FDP_IFC.1/JCVM Subset information flow control

FDP_IFC.1.1/JCVM The TSF shall enforce the **JCVM information flow control SFP** on **S.JCVM, S.LOCAL, S.MEMBER, I.DATA and OP.PUT (S1, S2, I)**.

Application note:
References of temporary Java Card RE entry points, which cannot be stored in class variables, instance variables or array components, are transferred from the internal memory of the Java Card RE (TSF data) to some stack through specific APIs (Java Card RE owned exceptions) or Java Card RE invoked methods (such as the process (APDU apdu)); these are causes of OP.PUT (S1, S2, 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:

Subject / Information	Description
S.JCVM	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 (S1, S.MEMBER, I) is allowed if and only if the active context is "Java Card RE";**
- **Other OP.PUT operations are allowed regardless of the Currently Active Context's value**.

FDP_IFF.1.3/JCVM The TSF shall enforce **no additional information flow control SFP rules**.

FDP_IFF.1.4/JCVM The TSF shall explicitly authorize an information flow based on the following rules: **no additional information flow control SFP rules**.

FDP_IFF.1.5/JCVM The TSF shall explicitly deny an information flow based on the following rules: **no additional information flow control SFP rules**.

Application note:

The storage of temporary Java Card RE-owned objects references is runtime-enforced ([JCRE22], §6.2.8.1-3).

It should be noticed that this policy essentially applies to the execution of bytecode. Native methods, the Java Card RE itself and possibly some API methods can be granted specific rights or limitations through the FDP_IFF.1.3/JCVM to FDP_IFF.1.5/JCVM elements. The way the Java Card 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 through an internal register prior to being pushed on the stack of the invoker. The returned bytecode would cause more than one OP.PUT operation under this scheme.

FMT_MSA.1/JCRE Management of security attributes

FMT_MSA.1.1/JCRE The TSF shall enforce the **FIREWALL access control SFP** to restrict the ability to **modify** the security attributes **the selected applet Context security attribute** to **the Java Card RE (S.JCRE)**.
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE222], §4 and [JCVM222], §3.4.

FMT_MSA.1/JCVM Management of security attributes

FMT_MSA.1.1/JCVM 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 currently active context and the Active Applets security attributes** to **the Java Card VM (S.JCVM)**.
Application note:
The modification of the Selected Applet Context is performed in accordance with the rules given in [JCRE222], §4 and [JCVM222], §3.4.

FMT_MSA.2/FIREWALL_JCVM Secure security attributes

FMT_MSA.2.1/FIREWALL_JCVM The TSF shall ensure that only secure values are accepted for **all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP**.

Application note:

The following rules are given as examples only. For instance, the last two rules are motivated by the fact that the Java Card API defines only transient arrays factory methods. Future versions may allow the creation of transient objects belonging to arbitrary classes; such evolution will naturally change the range of "secure values" for this component.

- The Context attribute of an O.JAVAOBJECT must correspond to that of an installed applet or be "Java Card RE".
- An O.JAVAOBJECT whose Sharing attribute is a Java Card RE entry point or a global array necessarily has "Java Card RE" as the value for its Context security attribute.
- An O.JAVAOBJECT whose Sharing attribute value is a global array necessarily has "array of primitive type" as a 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 initialization

FMT_MSA.3.1/FIREWALL The TSF shall enforce the **FIREWALL access 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 not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

Application note:

FMT_MSA.3.1/FIREWALL

Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable. At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".

The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_MSA.3/JCVM Static attribute initialization

FMT_MSA.3.1/JCVM The TSF shall enforce 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/JCVM The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

Application note:

FMT_MSA.3.1/FIREWALL

Objects' security attributes of the access control policy are created and initialized at the creation of the object or the subject. Afterwards, these attributes are no longer mutable. At the creation of an object (OP.CREATE), the newly created object, assuming that the FIREWALL access control SFP permits the operation, gets its Lifetime and Sharing attributes from the parameters of the operation; on the contrary, its Context attribute has a default value, which is its creator's Context attribute and AID respectively ([JCRE22], §6.1.3). There is one default value for the Selected Applet Context that is the default applet identifier's Context, and one default value for the Currently Active Context that is "Java Card RE".

The knowledge of which reference corresponds to a temporary entry point object or a global array and which does not is solely available to the Java Card RE (and the Java Card virtual machine).

FMT_MSA.3.2/FIREWALL

The intent is that none of the identified roles has privileges with regard to the default values of the security attributes. It should be noticed that creation of objects is an operation controlled by the FIREWALL access control SFP. The operation shall fail anyway if the created object would have had security attributes whose value violates FMT_MSA.2.1/FIREWALL_JCVM.

FMT_SMR.1/JCRE Security roles

FMT_SMR.1.1/JCRE The TSF shall maintain the roles**:**

- **the Java Card RE (JCRE)**.
- **the Java Card VM (JCVM)**.

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

FMT_SMF.1/CORE_LC Specification of Management Functions

FMT_SMF.1.1/Core_LC The TSF shall be capable of performing the following management functions:
modify the Currently Active Context, the Selected Applet Context, and the Active Applets

7.1.1.2 Application Programming Interface

The following SFRs are related to the Java Card API.
The execution of the additional native code is not within the TSF. Nevertheless, access to API native methods from the Java Card System is controlled by TSF because there is no difference between native and interpreted methods in the interface or the invocation mechanism.

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 **operations OP.JAVA and 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(), install() or uninstall() call, notwithstanding the restrictions given in [JCRE222], §7.7, within the bounds of the Commit Capacity ([JCRE222], §7.8), and those described in [JCAPI222]**.

Application note:

FDP_ROL.1.2/FIREWALL Transactions are a service offered by the APIs to applets. It is also used by some APIs to guarantee the atomicity of some operation. This mechanism is either implemented in Java Card platform or relies on the transaction mechanism offered by the underlying platform. Some operations of the API are not conditionally updated, as documented in [JCAPI22] (see for instance, PIN-blocking, PIN-checking, update of Transient objects). It should be noticed that the rollback within the scope of the uninstall() method only applies to Java Card platform, version 2.2.1 compliant TOEs.

7.1.1.3 Card Security Management

FAU_ARP.1 Security alarms

FAU_ARP.1.1 The TSF shall take **the following actions:**

- **throw an exception,**
- **or lock the card session**
- **or reinitialize the Java Card System and its data**

upon detection of a potential security violation.

Refinement:
The TOE detects the following potential security violation:

- Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI222] and ([JCRE222], §7.6.2)
- Violation of the Firewall or JCVM SFPs
- Unavailability of resources
- Array overflow

Application note:

The developer shall provide the exhaustive list of actual potential security violations the TOE reacts to. For instance, other runtime errors related to applet's failure like uncaught exceptions.

The bytecode verification defines a large set of rules used to detect a "potential security violation". The actual monitoring of these "events" within the TOE only makes sense when the bytecode verification is performed on-card.

Depending on the context of use and the required security level, there are cases where the card manager and the TOE must work in cooperation to detect and appropriately react in case of potential security violation. This behavior must be described in this component. It shall detail the nature of the feedback information provided to the card manager (like the identity of the offending application) and the conditions under which the feedback will occur (any occurrence of the java.lang.SecurityException exception).

The "locking of the card session" may not appear in the policy of the card manager. Such measure should only be taken in case of severe violation detection; the same holds for the re-initialization of the Java Card System. Moreover, the locking should occur when "clean" re-initialization seems to be impossible.

The locking may be implemented at the level of the Java Card System as a denial of service (through some systematic "fatal error" message or return value) that lasts up to the next "RESET" event, without affecting other components of the card (such as the card manager). Finally, because the installation of applets is a sensitive process, security alerts in this case should also be carefully considered herein.

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**.

Application note:

- The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE222], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE222]). Behavior of the TOE on power loss and reset is described in [JCRE222], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE222], §3.6.2.

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, the bytecode and its data argument,** when shared between the TSF and another trusted IT product.

FPT_TDC.1.2 The TSF shall use

- The rules defined in [JCVM222] specification;
- The API tokens defined in the export files of reference implementation
- **The rules defined in ISO 7816-6**
- **The rules defined in [GP221] specification**

when interpreting the TSF data from another trusted IT product.

Application note:

FPT_TDC.1.1:

Concerning the interpretation of data between the TOE and the underlying Java Card platform, it is assumed that the TOE is developed consistently with the SCP functions, namely concerning memory management, I/O functions, cryptographic functions, and so on.

7.1.1.4 AID Management

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:

- **package AID**
- **Applet's version number**
- **registered applet's AID**
- **applet selection status ([JCVM222], §6.5)**.

Application note:

- "Individual users" stands for applets.

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' AIDs** to **the JCRE**.

Application note:

The installer and the Java Card RE manage other TSF data such as the applet life cycle or CAP files, but this management is implementation specific. Objects in the Java programming language may also try to query AIDs of installed applets through the lookupAID(...) API method.

The installer may be granted the right to modify the list of registered applets' AIDs in specific implementations (possibly needed for installation and deletion; see #.DELETION and #.INSTALL).

FMT_MTD.3/JCRE Secure TSF data

FMT_MTD.3.1/JCRE The TSF shall ensure that only secure values are accepted for **the AIDs of registered applets**.

7.1.2 INSTG Security Functional Requirements

This group combines the SFRs related to the installation of the applets, which addresses security aspects outside the runtime. The installation of applets is a critical phase, which lies partially out of the boundaries of the firewall, and therefore requires specific treatment. In this ST, loading a package or installing an applet modeled as an importation of user data (that is, user application's data) with its security attributes (such as the parameters of the applet used in the firewall rules).

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** in the following cases:

- the applet package as identified by the package AID is already resident on the card.
- the applet package contains an applet with the same Java Card name as that of another applet already resident on the card.
- the applet package references a package that is not resident on the card.

The other error cases mentioned in [JCRE222] §11.1.4 are not in the scope of the TOE.

Application note:

The TOE may provide additional feedback information to the card manager in case of potential security violations (see FAU_ARP.1).

FPT_RCV.3/Installer Automated recovery without undue loss

FPT_RCV.3.1/Installer When automated recovery from **[none]** is not possible, the TSF shall enter a maintenance mode where the ability to return to a secure state is provided.
Application note:

- The TOE has no maintenance mode.

FPT_RCV.3.2/Installer For **[Failure during applet loading, installation and deletion; sensitive data loading]**, the TSF shall ensure the return of the TOE to a secure state using automated procedures.
FPT_RCV.3.3/Installer The functions provided by the TSF to recover from failure or service discontinuity shall ensure that the secure initial state is restored without exceeding **[none]** for loss of TSF data or objects under the control of the TSF.

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

Application note:

FPT_RCV.3.1/Installer:

This element is not within the scope of the Java Card specification, which only mandates the behavior of the Java Card System in good working order. Further details on the "maintenance mode" shall be provided in specific implementations. The following is an excerpt from [CC-2], p298: In this maintenance mode normal operation might be impossible or severely restricted, as otherwise insecure situations might occur. Typically, only authorized users should be allowed access to this mode but the real details of who can access this mode is a function of FMT: Security management. If FMT: Security management does not put any controls on who can access this mode, then it may be acceptable to allow any user to restore the system if the TOE enters such a state. However, in practice, this is probably not desirable as the user restoring the system has an opportunity to configure the TOE in such a way as to violate the SFRs.

FPT_RCV.3.2/Installer:

Should the installer fail during loading/installation of a package/applet, it has to revert to a "consistent and secure state". The Java Card RE has some clean up duties as well; see [JCRE22], §11.1.5 for possible scenarios. Precise behavior is left to implementers. This component shall include among the listed failures the deletion of a package/applet. See ([JCRE22], 11.3.4) for possible scenarios. Precise behavior is left to implementers.

Other events such as the unexpected tearing of the card, power loss, and so on, are partially handled by the underlying hardware platform (see [PP0035]) and, from the TOE's side, by events "that clear transient objects" and transactional features.

FPT_RCV.3.3/Installer:

The quantification is implementation dependent, but some facts can be recalled here. First, the SCP ensures the atomicity of updates for fields and objects, and a power-failure during a transaction or the normal runtime does not create the loss of otherwise-permanent data, in the sense that memory on a smart card is essentially persistent with this respect (EEPROM). Data stored on the RAM and subject to such failure is intended to have a limited lifetime anyway (runtime data on the stack, transient objects' contents). According to this, the loss of data within the TSF scope should be limited to the same restrictions of the transaction mechanism.

7.2 SECURITY ASSURANCE REQUIREMENTS

The security assurance requirement level is EAL7.

7.3 SECURITY REQUIREMENTS RATIONALE

7.3.1 Objectives

	O.OPERATE	O.FIREWALL	O.NATIVE	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.INSTALL
FDP_ACC.2/FIREWALL	X	X				
FDP_ACF.1/FIREWALL	X	X	X			

	O.OPERATE	O.FIREWALL	O.NATIVE	O.GLOBAL_ARRAYS_INTEG	O.ALARM	O.INSTALL
FDP_IFC.1/JCVM		X		X		
FDP_IFF.1/JCVM		X		X		
FMT_MSA.1/JCRE		X				
FMT_MSA.1/JCVM		X				
FMT_MSA.2/FIREWALL_JCVM		X				
FMT_MSA.3/FIREWALL		X				
FMT_MSA.3/JCVM		X				
FMT_SMR.1/JCRE		X				
FMT_SMF.1/CORE_LC		X				
FDP_ROL.1/FIREWALL	X					
FAU_ARP.1	X				X	
FPT_FLS.1/JCS	X				X	
FPT_TDC.1	X					
FIA_ATD.1/AID	X					
FMT_MTD.1/JCRE		X				
FMT_MTD.3/JCRE		X				
FPT_FLS.1/Installer	X				X	X
FPT_RCV.3/Installer	X					X

Table 3: Objective vs. SFR

7.3.1.1 Security objectives for the TOE

7.3.1.1.1 EXECUTION

O.FIREWALL This objective is met by the FIREWALL access control policy (FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL), the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM). The functional requirements of the class FMT (FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, and FMT_MSA.1/JCRE) also indirectly contribute to meet this objective.

O.GLOBAL_ARRAYS_CONFID Only arrays can be designated as global, and the only global arrays required in the Java Card API are the APDU buffer and the byte array input parameter (bArray) to an applet's install method. The JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from keeping a pointer to a shared buffer, which could be used to read its contents when the buffer is being used by another application.

O.NATIVE This security objective is covered FDP_ACF.1/FIREWALL that the only means that the subject S.JCVM shall provide for an application to execute native code is the invocation of a Java Card API method. This objective mainly relies on the environmental OE.APPLET, which upholds the assumption A.APPLET.

OE.OPERATE The TOE is protected in various ways against applets' actions (FPT_TDC.1), the FIREWALL access control policy (FDP_ACC.2/FIREWALL and FDP_ACF.1/FIREWALL), and is able to detect and block various failures or security violations during usual working (FPT_FLS.1/Installer, FAU_ARP.1). Its security-critical parts and procedures are also protected: safe recovery from failure is ensured

(FPT_RCV.3/Installer), applets' installation may be cleanly aborted (FDP_ROL.1/FIREWALL), FIA_ATD.1/AID) to prevent alteration of TSF data (also protected by components of the FPT class).

Almost every objective and/or functional requirement indirectly contributes to this one too.

7.3.1.1.2 SERVICES

O.ALARM This security objective is met by FPT_FLS.1/Installer, FPT_FLS.1 which guarantee that a secure state is preserved by the TSF when failures occur, and FAU_ARP.1 which defines TSF reaction upon detection of a potential security violation.

7.3.1.1.3 APPLET MANAGEMENT

O.INSTALL This security objective specifies that installation of applets must be secure. In partocular, the TSFs are protected against possible failures of the installer (FPT_FLS.1/Installer, FPT_RCV.3/Installer). Note that the security of the applet loading/deleting is not included in this objective.

7.3.2 Dependencies

7.3.2.1 SFRs DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
FAU_ARP.1	FAU_SAA.1	Unsupported
FDP_ACC.2/FIREWALL	FDP_ACF.1	FDP_ACF.1/FIREWALL
FDP_ACF.1/FIREWALL	FDP_ACC.1, FMT_MSA.3	FDP_ACC.2/FIREWALL, FMT_MSA.3/FIREWALL
FDP_IFC.1/JCVM	FDP_IFF.1	FDP_IFF.1/JCVM
FDP_IFF.1/JCVM	FDP_IFC.1, FMT_MSA.3	FDP_IFC.1/JCVM, FMT_MSA.3/JCVM
FDP_ROL.1/FIREWALL	(FDP_ACC.1 or FDP_IFC.1)	FDP_ACC.2/FIREWALL, FDP_IFF.1/JCVM
FIA_ATD.1/AID	none	
FMT_MSA.1/JCVM	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_SMF.1/CORE_LC, FMT_SMR.1/JCRE
FMT_MSA.2/FIREWALL_JCVM	(FDP_ACC.1 or FDP_IFC.1), FMT_MSA.1, FMT_SMR.1	FDP_ACC.2/FIREWALL, FDP_IFC.1/JCVM, FMT_MSA.1/JCRE, FMT_SMR.1/JCRE
FMT_MSA.3/FIREWALL	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_SMR.1/JCRE
FMT_MSA.3/JCVM	FMT_MSA.1, FMT_SMR.1	FMT_MSA.1/JCVM, FMT_SMR.1/JCRE
FMT_MTD.1/JCRE	FMT_SMF.1, FMT_SMR.1	FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC
FMT_MTD.3/JCRE	FMT_MTD.1	FMT_MTD.1/JCRE
FMT_SMR.1/JCRE	FIA_UID.1	FIA_UID.2/AID
FMT_SMF.1/CORE_LC	none	
FPT_FLS.1/JCS	none	
FPT_FLS.1/Installer	none	
FPT_RCV.3/Installer	AGD_OPE.1	AGD_OPE.1
FPT_TDC.1	none	

7.3.2.1.1 RATIONALE FOR THE EXCLUSION OF DEPENDENCIES

The dependency FAU_SAA.1 of FAU_ARP.1 is unsupported. Potential violation analysis is used to specify the set of auditable events whose occurrence or accumulated occurrence held to indicate a potential violation of the SFRs, and any rules to be used to perform the violation analysis. The dependency of FAU_ARP.1 on this functional requirement assumes that a "potential security violation" is an audit event indicated by the FAU_SAA.1 component. The events listed in FAU_ARP.1 are, on the contrary, merely self-contained ones (arithmetic exception, ill-formed bytecodes, access failure) and ask for a straightforward reaction of the TSFs on their occurrence at runtime. The JCVM or other components of the TOE detect these events during their usual working order. Thus, in principle there would be no applicable audit recording in this framework. Moreover, no specification of one such recording is provided elsewhere. Therefore no set of auditable events could possibly be defined.

7.3.2.2 SARs DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
ADV_ARC.1	ADV_FSP.1; ADV_TDS.1	ADV_FSP.6; ADV_TDS.6
ADV_SPM.1	ADV_FSP.4	ADV_FSP.6
ADV_FSP.6	ADV_TDS.1	ADV_TDS.6
ADV_IMP.2	ADV_TDS.3; ALC_TAT.1 and ALC_CMC.5	ADV_TDS.6; ALC_TAT.3, ALC_CMC.5
ADV_INT.3	ADV_IMP.1; ADV_TDS.3; ALC_TAT.1	ADV_IMP.2; ADV_TDS.6; ALC_TAT.3
ADV_TDS.6	ADV_FSP.6	ADV_FSP.6
AGD_OPE.1	ADV_FSP.1	ADV_FSP.6
AGD_PRE.1	No dependencies	N/A
ALC_CMC.5	ALC_CMS.1; ALC_DVS.2; ALC_LCD.1	ALC_CMS.5; ALC_DVS.2; ALC_LCD.1
ALC_CMS.5	No dependencies	N/A
ALC_DEL.1	No dependencies	N/A
ALC_DVS.2	No dependencies	N/A
ALC_LCD.2	No dependencies	N/A
ALC_TAT.3	ADV_IMP.1	ADV_IMP.2
ASE_CCL.1	(ASE_ECD.1) and (ASE_INT.1) and (ASE_REQ.1)	ASE_ECD.1, ASE_INT.1, ASE_REQ.2
ASE_ECD.1	No dependencies	N/A
ASE_INT.1	No dependencies	N/A
ASE_OBJ.2	(ASE_SPD.1)	ASE_SPD.1
ASE_REQ.2	(ASE_ECD.1) and (ASE_OBJ.2)	ASE_ECD.1, ASE_OBJ.2
ASE_SPD.1	No dependencies	
ASE_TSS.1	(ADV_FSP.1) and (ASE_INT.1) and (ASE_REQ.1)	ADV_FSP.6, ASE_INT.1, ASE_REQ.2
ATE_COV.3	ADV_FSP.2; ATE_FUN.1	ADV_FSP.6; ATE_FUN.2
ATE_DPT.4	ADV_ARC.1; ADV_TDS.4; ADV_IMP.1; ATE_FUN.1	ADV_ARC.1; ADV_TDS.4; ADV_IMP.2; ATE_FUN.1
ATE_FUN.2	ATE_COV.1	ATE_COV.3
ATE_IND.3	ADV_FSP.4; AGD_OPE.1; AGD_PRE.1; ATE_COV.1; ATE_FUN.1	ADV_FSP.6; AGD_OPE.1; AGD_PRE.1; ATE_COV.3; ATE_FUN.2
AVA_VAN.5	ADV_ARC.1; ADV_FSP.4; ADV_TDS.3; ADV_IMP.1; AGD_OPE.1; AGD_PRE.1; ATE_DPT.1	ADV_ARC.1; ADV_FSP.6; ADV_TDS.6; ADV_IMP.2; AGD_OPE.1; AGD_PRE.1; ATE_DPT.4

7.3.3 Rationale for the security assurance requirements

7.3.3.1 EAL7: Formally verified design and tested

EAL7 is required for this type of TOE and product since it is intended to defend against sophisticated attacks. This evaluation assurance level allows a developer to gain maximum assurance from positive security engineering based on good practices. In order to provide a meaningful level of assurance that the TOE and its embedding product provide an adequate level of defense against such attacks: the evaluators should have

access to the low level design and source code. Additionally the formal model of select TOE security policies and the semiformal presentation of the functional specification and TOE design, provided by EAL7, gives a more structured presentation of the implementation and a thus more assurance on the TOE.

This product is intended to be used in an open environment where sensitive and non-sensitive but hostile applications will co-exist on the product. One of the most sensitive functions of the embedded software of this product, providing the property of isolation between applications is **the firewall**. To provide assurance on the correct behavior of this security function, this security target provides formal assurances on its development from the EAL7 level. The formal assurances provide evidence that this function has been implemented correctly with respect to the specification.

7.3.3.2 ADV_SPM.1 Formal TOE security policy model

The formally modelled security policies consist of:

1. Firewall security policy that controls the sharing of data containers owned by applets of different packages, or the JCRE, and between applets and the TSFs (cf. O.FIREWALL objective);
2. Typing security policy that supposes that all the bytecodes are verified before being loaded/installed/executed on the TSF (cf. A.VERIFICATION assumption);
3. (Static) information access security policy that ensures that an external reference is accessible in the current package only of this reference is exported by its package and that package is imported by the current package (part of O.INSTALL objective).

The SPM component is fulfilled by the following evident elements.

Requirement	Title	Type
TOE Security Policy	Interpreter	Document
	Linker (static access control)	Document
	BCV/Typing	Document
ADV_SPM.1.1C	SPM formal model	Coq code
	Model of the JC virtual machine	Document
	Model of the Firewall policy	Document
	Model of the Typing policy	Document
	Model of Java Card API	Document
	Correspondence between ISP and TSP model	Document
ADV_SPM.1.2C	FWVM state machine and JCVM/FWVM proof	Coq code
	Proof of O.Firewall: confidentiality	Document
	TYVM state machine and JCVM/TYVM proof	Coq code
	Proof of the safe execution of a CAP file	Document
	Proof of OT.GLOBAL_ARRAYS_INTEG	Coq code
	Proof of O.FIREWALL: integrity	Coq code
ADV_SPM.1.3C ADV_SPM.1.4C ADV_SPM.1.5C	Correspondence proof between the FSP and the TSP formal models (linker, interpreter, API)	Coq code
	Linker: Correspondence between FSP and TSP models	Document
	Interpreter: Correspondence between FSP and TSP models	Document
	Native API: Correspondence between the FSP and the TSP formal models	Document

It depends on ADV_FSP.6 that is satisfied by this evaluation.

7.3.3.3 ADV_FSP.6 Complete semi-formal functional specification with additional formal specification

This requirement is fulfilled by the following evident elements.

Requirement	Title	Type
ADV_FSP.6	Linker: FSP model	Coq code

	Interpreter: FSP model	
	API (native): FSP model	
	Functional Spec of the Linker	Document
	Functional Spec of the Interpreter	Document
	Functional Spec of the native API	Document
	Correspondence between ST (SFRs) and FSP model	Document

It depends on ADV_TDS.6 that is satisfied by this evaluation.

7.3.3.4 *ADV_TDS.6 Complete semi-formal modular design with formal high-level design presentation*

This requirement is fulfilled by the following evident elements.

Requirement	Title	Type
ADV_TDS.6 (subsystem)	Linker: HLD model	Coq code
	Embedded interpreter: HLD model	
	Native API: HLD model	
	Linker: High-Level Design	Document
	Interpreter: High-Level Design	Document
	Native API: High-Level Design	Document
ADV_TDS.6 (proofs for subsystems)	Correspondence between the HLD and the FSP formal models (linker, interpreter, API)	Coq code
	Linker: FSP-HLD correspondence	Document
	Interpreter: FSP-HLD correspondence	Document
	Native API: FSP-HLD correspondence	Document
ADV_TDS.6 (modules)	Interpreter and API: LLD model	Coq code
	Linker: Low-Level Design	Document
	Interpreter: Low-Level Design	Document
	Native API: Low-Level Design	Document
ADV_TDS.6 (proofs for module)	Correspondence between the LLD and the HLD formal models (interpreter, API)	Coq code
	Correspondence between the LLD and the HLD formal models (interpreter)	Document
	Correspondence between the LLD and the HDL formal models (Native API)	Document

It depends on the ADV_FSP.6 that is satisfied by this evaluation.

7.3.3.5 *ADV_IMP.2 Complete mapping of the implementation representation of the TSF*

This requirement is fulfilled by the following evident elements.

Title	Type
TSF source code (in HTML format)	HTML

It depends on ADV_TDS.6, ALC_TAT.3, ALC_CMC.5 that are satisfied by this evaluation

7.3.3.6 *ADV_INT.3 Minimally complex internals*

This requirement is fulfilled by the following evident elements.

Requirement	Title	Type
ADV_INT.3	Minimally complex internals	Document

It depends on ADV_IMP.2, ADV_TDS.6 and ALC_TAT.3. that are all satisfied by this evaluation.

7.3.3.7 *ATE_DPT.4. Testing: implementation representation*

It depends on ADV_ARC.1, ADV_TDS.4, ADV_IMP.1 and ATE_FUN.1 that are all satisfied by this evaluation.
This component is fulfilled by the following evident elements.

Requirement	Title	Type
ATE_DPT.4	Additional Rationale on Tests	Document

7.3.3.8 *ATE_COV.3. Rigorous analysis of coverage*

It depends on ADV_FSP.2 and ATE_FUN.1 that are all satisfied by this evaluation.
This component is fulfilled by the following evident elements.

Requirement	Title	Type
ATE_COV.3	Additional Rationale on Tests	Document

7.3.3.9 *ATE_FUN.2. Ordered Functional Testing*

It depends on ATE_COV.1 that is satisfied by this evaluation.
This component is fulfilled by the following evident elements.

Requirement	Title	Type
ATE_FUN.2	Additional Rational on Tests	Document

7.3.3.10 *ALC_CMC.5 Advanced support*

This requirement is fulfilled by the following evident elements:

Requirement	Title	Type
ALC_CMC.5	Evidence elements for ADV_CMC.5, ALC_LCD.2 and ALC_TAT.3	Document

It depends on ALC_CMS.1, ALC_DVS.2 and ALC_LCD.1 that are all satisfied by this evaluation.

7.3.3.11 *ALC_LCD.2 Measurable life-cycle model*

This component has no dependency and is fulfilled by the following evident elements.

Requirement	Title	Type
ADV_LCD.2	Evidence elements for ADV_CMC.5, ALC_LCD.2 and ALC_TAT.3	Document

7.3.3.12 *ALC_TAT.3 Compliance with implementation standards – all parts*

It depends on ADV_IMP.1 that is satisfied by this evaluation.
This component is fulfilled by the following evident elements.

Requirement	Title	Type
ADV_TAT.3	Evidence elements for ADV_CMC.5, ALC_LCD.2 and ALC_TAT.3	Document

7.3.3.13 *AVA_VAN.5 Advanced methodical vulnerability analysis*

The TOE is intended to operate in hostile environments. AVA_VAN.5 "Advanced methodical vulnerability analysis" is considered as the expected level for Java Card technology-based products hosting sensitive applications, in particular in payment and identity areas.

7.3.3.14 ALC_DVS.2 Sufficiency of security measures

Development security is concerned with physical, procedural, personnel and other technical measures that may be used in the development environment to protect the TOE and the embedding product.

7.3.3.15 Other Security Assurance Requirements

The other EAL7 security assurance requirements satisfied by this evaluation are fulfilled by evidences provided in the EAL5+ evaluation of the MultiApp V4 JCS [ST_MultiAppv4].

7.3.4 Compatibility with SFR of [ST-IC]

FDP_IFC.1/JCVM, FDP_IFF.1/JCVM, FDP_RIP.1/OBJECTS, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC, FCS_CKM.2, FCS_CKM.3, FCS_CKM.4, FDP_RIP.1/APDU, FDP_RIP.1/bArray, FDP_RIP.1/ABORT, FDP_RIP.1/KEYS, FDP_ROL.1/FIREWALL, FAU_ARP.1, FPT_TDC.1, FPT_FLS.1/JCS, FPR_UNO.1, FMT_MTD.1/JCRE, FMT_MTD.3/JCRE, FIA_ATD.1/AID, FIA_UID.2/AID, FIA_USB.1/AID, FDP_ITC.2/Installer, FMT_SMR.1/Installer, FPT_FLS.1/Installer, FPT_RCV.3/Installer, FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FDP_ACC.2/ADEL, FDP_ACF.1/ADEL, FDP_RIP.1/ADEL, FPT_FLS.1/ADEL, FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL, FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FDP_RIP.1/TRANSIENT, FDP_RIP.1/ODEL, FPT_FLS.1/ODEL, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_SMF.1/CM, FCO_NRO.2/CM, FIA_UAU.1/CM, FIA_UID.1/CM, FDP_IFC.2/CM, FDP_IFF.1/CM, FDP_UIT.1/CM, FTP_ITC.1/CM, FPT_TST.1/SCP, FPT_RCV.4/SCP, FDP_ACC.1/CMGR, FDP_ACF.1/CMGR, FMT_MSA.1/CMGR, and FMT_MSA.3/CMGR are SFR specific to the Java Card platform and they do no conflict with the SFR of [ST-IC].
FPT_FLS.1/SpecificAPI, FPT_ITT.1/SpecificAPI, FPR_UNO.1/SpecificAPI are SFR added to this ST regarding API security and they do no conflict with the SFR of [ST-IC].
FCS_CKM.1, FCS_COP.1 and FCS_RND.1 of this ST are supported by FCS_CKM.1, FCS_COP.1 of [ST-IC].
FDP_SDI.2 of this ST is supported by FDP_SDI.1 and FDP_SDI.2 of [ST-IC].
FPT_PHP.3/SCP of this ST is supported by FPT_PHP.3 of [ST-IC].
We can therefore conclude that the SFR of this ST and [ST-IC] are consistent.

8 TOE SUMMARY SPECIFICATION

8.1 TOE SECURITY FUNCTIONS

TOE Security Functions are provided by the TOE embedded software (including the optional NVM ES) and by the chip.

8.1.1 Security functions provided by MultiApp V4 platform

8.1.1.1 SF.FW: 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* 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*.

An operation OP.PUT (S1, 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_IFC.1/JCVM FDP_IFF.1/JCVM
Only the S.JCRE can modify the security attributes the active context, the selected applet context security attributes.	FMT_MSA.1/JCRE
Only the S.JCVM can modify the security attributes the active context, the currently active Context and the Active Applets security attributes.	FMT_MSA.1/JCVM
only secure values are accepted for all the security attributes of subjects and objects defined in the FIREWALL access control SFP and the JCVM information flow control SFP.	FMT_MSA.2/FIREWALL_ JCVM
provide restrictive default values for security attributes that are used to enforce the SFP.	FMT_MSA.3/FIREWALL
The TSF shall maintain the roles: the Java Card RE, the Java Card VM. The TSF shall be able to associate users with roles.	FMT_SMR.1/JCRE
The TSF shall be capable of performing the following management functions: • Modify the active context and the SELECTed applet Context. • Modify the list of registered applets' AID	FMT_SMF.1/CORE_LC
([JCRE222]§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".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE222]§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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
([JCRE222]§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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
• ([JCRE222], §6.2.8.6,) An S.PACKAGE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT whose Sharing attribute has the value "SIO", and whose Context attribute has the value "Package AID", only if one of the following applies: (c) The value of the attribute Selection Status of the package whose AID is "Package AID" is "Multiselectable", (d) The value of the attribute Selection Status of the package whose AID is "Package AID' is "Non-multiselectable", and either "Package AID" is the value of the currently selected applet or otherwise "Package AID" does not occur in the attribute ActiveApplets,	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL

and in either of the cases above the invoked interface method extends the Shareable interface	
An S.PACKAGE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The subject S.JCRE can freely perform OP.JAVA(...) and OP.CREATE, with the following two exceptions: 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.	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL
The TSF shall permit the rollback of the operations OP.JAVA and OP.CREATE on the O.JAVAOBJECTs.	FDP_ROL.1/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 [JCRE222], §7.7, within the bounds of the Commit Capacity ([JCRE222], §7.8), and those described in [JCAPI222].	FDP_ROL.1/FIREWALL
Only updates to persistent objects participate in the transaction. Updates to transient objects and global arrays are never undone, regardless of whether or not they were “inside a transaction.” [JCRE222], §7.7	FDP_ROL.1/FIREWALL
A TransactionException is thrown if the commit capacity is exceeded during a transaction. [JCRE222], §7.8	FDP_ROL.1/FIREWALL
Transaction & PIN: When comparing a PIN, even if a transaction is in progress, update of internal state - the try counter, the validated flag, and the blocking state, do not participate in the transaction. [JCAPI222]	FDP_ROL.1/FIREWALL

8.1.1.2 SF.CSM: Card Security Management

The TSF shall take the following actions: • throw an exception, • or lock the card session • or reinitialize the Java Card System and its data upon detection of a potential security violation.	FAU_ARP.1
The TOE detects the following potential security violation: • Abortion of a transaction in an unexpected context (see abortTransaction(), [JCAPI222] and ([JCRE222], §7.6.2) • Violation of the Firewall or JCVM SFPs • Unavailability of resources • Array overflow	FAU_ARP.1
In order to consistently interpret **the CAP files, the bytecode and its data argument,** when shared between the TSF and another trusted IT product, the TSF shall use: • The rules defined in [JCVM222] specification; • The API tokens defined in the export files of reference implementation • **The rules defined in ISO 7816-6** • **The rules defined in [GP221] specification**	FPT_TDC.1
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.** The Java Card RE Context is the Current context when the Java Card VM begins running after a card reset ([JCRE222], §6.2.3) or after a proximity card (PICC) activation sequence ([JCRE222] §4.1.2). Behavior of the TOE on power loss and reset is described in [JCRE222], §3.6, and §7.1. Behavior of the TOE on RF signal loss is described in [JCRE222], §3.6.2	FPT_FLS.1/JCS

8.1.1.3 SF.AID: AID Management

Only **the JCRE** can **modify** the **list of registered applets' AIDs**.	FMT_MTD.1/JCRE
Only secure values are accepted for **the AIDs of registered applets**.	FMT_MTD.3/JCRE
The TSF shall maintain the following list of security attributes belonging to individual users: • **package AID** • **Applet's version number** • **registered applet's AID** • **applet selection status ([JCVM222], §6.5)**	FIA_ATD.1/AID

8.1.1.4 SF.INST: 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 [JCRE222] §11.1.4**	FPT_FLS.1/Installer
After **Failure during applet loading, installation and deletion; sensitive data loading**, the TSF ensures the return of the TOE to a secure state using automated procedures. The TSF provides the capability to determine the objects that were or were not capable of being recovered.	FPT_RCV.3/Installer

8.2 TOE SUMMARY SPECIFICATION RATIONALE

8.2.1 TOE security functions rationale

	SF.FW	SF.CSM	SF.AID	SF.INST
FDP_ACC.2/FIREWALL	X			
FDP_ACF.1/FIREWALL	X			
FDP_IFC.1/JCVM	X			
FDP_IFF.1/JCVM	X			
FMT_MSA.1/JCRE	X			
FMT_MSA.1/JCVM	X			
FMT_MSA.2/FIREWALL_JCVM	X			
FMT_MSA.3/FIREWALL	X			
FMT_MSA.3/JCVM	X			
FMT_SMR.1/JCRE	X			
FMT_SMF.1/CORE_LC	X			
FDP_ROL.1/FIREWALL	X			
FAU_ARP.1		X		
FPT_FLS.1/JCS		X		
FPT_TDC.1		X		
FIA_ATD.1/AID			X	
FMT_MTD.1/JCRE			X	
FMT_MTD.3/JCRE			X	

	SF.FW	SF.CSM	SF.AID	SF.INST
FPT_FLS.1/Installer				X
FPT_RCV.3/Installer				X

Table 4: Rationale table of functional requirements and security functions

END OF DOCUMENT