MultiApp V5.1 Java Card Virtual Machine

Common Criteria / ISO 15408
Security Target – Public version
EAL7

Version 1.3 - October 4th, 2023

CONTENT

FIGURES

TABLES

1 SECURITY TARGET INTRODUCTION

1.1 SECURITY TARGET REFERENCE

Title :	MultiApp V5.1: VM Security Target
Version :	1.3
ST Reference :	D1586135_LITE
Origin :	Thales DIS
IT Security Evaluation Facility :	LETI
IT Security Certification scheme :	Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)

1.2 TOE REFERENCE

The product is available in two configurations:

- Configuration 1: TOE based on the IC AQUARIUS_BA_09- AQUARIUS_v1 called "revision B"
- Configuration 2: TOE based on the IC AQUARIUS_CA_09- AQUARIUS_v2 called "revision C"

Product Technical Name :	MultiApp V5.1 Virtual Machine
Product Commercial Names:	MultiApp V5.1 Virtual Machine

Security Controllers :	For the configuration 1 (Revision B):
	5.1 revision B (0x3055)
	For the configuration 2 (Revision C):
	5.1 revision C (0x3055)

TOE Name :	MultiApp V5.1 Java Card Virtual Machine
TOE Version :	5.1 Revision C (0x3055)
TOE documentation :	Guidance [AGD]
Composition elements:	
Composite TOE identifier:	For the configuration 1: AQUARIUS_BA_09 - AQUARIUS_v1 For the configuration 2: AQUARIUS_CA_09 - AQUARIUS_v2
Composite TOE Version:	For the configuration 1 (Revision B): Hardware revision: B For the configuration 2 (Revision C): Hardware revision: C For both configurations: Platform ROM Firmware Revision: A Platform FLASH Firmware Revision: 09 ➢ *BIOS: Version 1.0-911* ➢ *Loader: Version 2.2*

1.3 TOE IDENTIFICATION

The TOE identification is provided by the Tag identity and CPLC data. These data are available by executing a dedicated command described in [AGD-OPE] and here below:

The TOE can be identified through the Get Data Command response with tag "0103", as follows:

Name	Length	Description	Value	Participate to TOE identification
Thales Family Name	1	Java Card	0xB0	YES
Thales OS name	1	MultiApp	0x85	YES
Thales Mask Number	1	MultiappV5	0x68	YES
Thales Product Name	1		0x6A	YES
Flow id Version	1		0x01	YES
Filter set	1		0x00	YES
Chip Manufacturer	2	Infineon	0x1290	YES
Chip Identifier	2	Identifier		NO
BPU	2	BPU configuration		NO
PDM TP	3			NO
PDM CI	3			NO
Feature FLag – Crypto Config	2	See after		NO
Feature Flag – Feature Config byte 1	1	See after		NO
Feature Flag – Feature Config byte 2	1	See after		NO
Platform Certificates	1		Bit 7 (0x040): CC Configuration	YES (only for byte 7)
APPLI CERTIFICATES byte 1	1		Bit 8 (0x80): eTravel Bit 7 (0x40): IAS Bit 6-1 : Not used (0)	YES (only for byte 8 & 7)
APPLI CERTIFICATES byte 2	1		00h	NO

Note: the eight first fields of this table (from "Thales Family Name" to "Chip Identifier") are used for traceability purpose.

Also, using Get data command with tag 9F7F for product identification :

Name	length	Description	Value	Participate to TOE identification
IC Fabricator	2	Chip fabricator	0x12 0x90	YES
IC Type	2	Chip model number	0x00 0x13	YES
Operating system identifier	2	OS developer	0x19 0x81	YES
Operating system release date	2	Date reference	0x30 0x55	YES
Operating system release level	2	5.1	0x05 0x10	YES

Also, using Get data command with tag 010B for product identification:

Index	Description	IC Revision B Value (Previous)	IC Revision C Value	Part of TOE the identification
0	PRODUCT	0x09	0x09	No
1	HW_REV	0x42	0x43	Yes
2	Not relevant for identification			No
…				No
15				No

The TOE and the product differ, as further explained in Architecture of the product.

- The TOE is the JCS open platform of the MultiApp V5.1 product.
- The MultiApp V5.1 product also includes applets.

Optional features / Field (extract from identity tag)	Crypto features byte A								Crypto features byte B								features byte 1								features byte 2							
bit	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
ECC	x																															
RSA									x																							
RSA-DH												x																				
RSA-OBKG													x																			
RSA-4K															x																	
SHA3																																
HMAC				x																												
PACE DH																		x														
PACE ECC																			x													
File system																				x												
ISM																					x											
Etravel																						x										
EAC/GAP																							x									
Linker																								x								
Biometry Fingerprint																									x							
Biometry Facial																										x						
Biometry IRIS																											x					
(Not used)																																
(Not used)																																
FIPS																															x	

Table 1: MAV 5.1 Features configuration

Note 1: X with value 1 when the feature is available, X with value 0 when the feature is not available.
Note 2: The bits that are not listed in the table 1 are considered as RFU

The TOE and the product differ, as further explained in Architecture of the product

- The TOE is the Java Card Virtual Machine of the MultiApp V5.1 product.

1.4 SECURITY TARGET OVERVIEW

The main objectives of this ST are:

- To introduce 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 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 and TOE security functions.

1.5 REFERENCES

1.5.1 External References

[CC]	**Common Criteria references**
[CC-1]	Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, CCMB-2017-04-001, Version 3.1 Revision 5, April 2017.
[CC-2]	Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, CCMB-2017-04-002, Version 3.1 Revision 5, April 2017.
[CC-3]	Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, CCMB-2017-04-003, Version 3.1 Revision 5, April 2017.
[CEM]	Common Methodology for Information Technology Security Evaluation Methodology CCMB-2017604-001, version 3.1 rev 5 April 2017
[JIL_CPE]	Joint Interpretation Library: Composite product evaluation for Smart Cards and similar devices, Version 1.5.1 May 2018
[PP]	**Protection profiles**
[PP-IC-0084]	Security IC Platform Protection Profile with augmentation Packages– BSI-CC-PP-0084-2014 version 1.0
[PP-JCS-Open]	Java Card System Protection Profile – Open Configuration BSI-CC-PP-0099-V2-2020, Version 3.1, April 2020
[AIS31]	A proposal for: Functionality classes for random number generators Version 2.0 Sept 2011
[THALES DS]	**Thales DIS France SAS References**
[AQU-IC]	[AQU-IC-GDA]
[AQU-IC-GDA]	Security Target for AQUARIUS (Microcontroller AQUARIUS_BA_09 & CA_09 - AQUARIUS_v2) Ref: AQUARIUS_ST Revision: 1.0 – 10/03/2023
[CR-IC]	[CR-IC-GDA]
[CR-IC-GDA]	Certification Report, CERTIFICAT ANSSI-CC-2023/01-M01 AQUARIUS_BA_09 & CA_09 - AQUARIUS_v2 EAL6 Augmenté (ASE_TSS.2, ALC_FLR.2) Date : 21/06/2023 conforme au profil de protection : Security IC Platform Protection Profile with Augmentation Packages, version 1.0 certifié BSI-CC-PP-0084-2014 le 19 février 2014
[GP]	**Global Platform references**
[GP23]	GP2.3.1: GPC_CardSpecification_v2.3.1_PublicRelease_CC.pdf GlobalPlatform Technology Card Specification Version 2.3.1 - March 2018 Reference: GPC_SPE_034
[GP23 Amend D]	Card Technology Secure Channel Protocol '03' Card Specification v2.3 – Amendment D Version 1.1.2 - March 2019 Reference: GPC_SPE_014
[GP23 Amend E]	Card Technology Security Upgrade for Card Content Management Card Specification v2.3 – Amendment E Version 1.1 - October 2016 Reference: GPC_SPE_042
[GP23 Com]	Global Platform – Card Common Implementation Configuration Version v2.1 - July 2018

[GP PF]	GlobalPlatform TechnologyCard Specification – Privacy FrameworkVersion 1.0.1 Public Release November 2019 Document Reference: GPC_SPE_10
[Others]	**Others specification references**

[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.
[JCRE3]	Java Card 3.1 Runtime Environment (JCRE) Specification – November 2019 – Published by Oracle
[JCVM3]	Java Card 3.1 Virtual Machine (JCVM) Specification – November 2019 – Published by Oracle
[JCAPI3]	Java Card 3.1 Application Programming Interface (API) Specification, Classic Edition - November 2019 – Published by Oracle

1.5.2 Internal References [IR]

[ALC]	MulitiApp V5.1 Software – Life Cycle documentation
[ALC-DVS]	MultiApp V5.1: ALC DVS document - Javacard Platform, D1576205
[AGD]	MulitiApp V5.1 Software – Guidance documentation
[AGD-PRE]	Preparation Guidance, D1574816
[AGD-OPE]	Operational Guidance, D1574815
[AGD-Ref]	MultiApp ID Operating System –Reference Manual, D15525385C
[ASE]	MulitiApp V5.1 Software – Security Target
[ST-MAV51]	MulitiApp V5.1 Software – JCS Security Target – D1572544
[MAV51_SPM]	MultiApp V5.1 SPM – Formal Security Policy Model of the Java Card Virtual Machine - D1590147

1.6 ACRONYMS AND GLOSSARY

AES	Advanced Encryption Standard
APDU	Application Protocol Data Unit
API	Application Programming Interface
CAD	Card Acceptance Device
CC	Common Criteria
CPU	Central Processing Unit
DES	Data Encryption Standard
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
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
RSA	Rivest Shamir Adleman
SAR	Security Assurance Requirement
SC	Smart Card
SCP	Secure Channel Protocol
SFP	Security Function Policy
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
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 [JCVM3]. 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 MultiAppV5.1: JCS Security Target [ST-MAV51]. This security target restricts the security target [ST-MAV51] 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 interpreter

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 V5.1* smartcard product. This smartcard contains the software dedicated to the operation of:

- ➢ The MultiApp V5.1 Platform, which supports the execution of the personalized applets and provides the smartcard administration services. It is conformant to Java Card 3.1 and GP 2.3.1 standards [GP23]. (With common configuration 2.1 [GP23 Com]) and with GP Privacy Framework v1.0.1 [GP PF].

  The identity applets: GDP, IAS classic V5.2, eTravel v3.1, BioPin Manager v3.1 (MOCA server/client), MPCOS v4.1, MSFT PnP v1.0, FIDO Authentificator v2.1 applet, LDSv2 v1.1 , PURE DI v3.6.0, TachoG2V2, BelPIC v1.8, Privacy Manager v1.0.

Applet name	Package	Package AID
GDP	com.gemalto.javacardx.gdp	A00000001810020303
LDSV2 v1.1	com/gemalto/javacard/icao/lds2	A000000018300B0201000000000000FE
IAS Classic v5.2	com.gemalto.javacard.iasclassic	A0000000188000000000066240FF
eTravel 3.1	N/A (Natif)	A000000018300B0200000000000000FF
BioPin Manager v3.1: MOC Client	com.gemalto.moc.client	4D4F43415F436C69656E74
BioPin Manager v3.1: MOC Server	com.gemalto.moc.server	4D4F43415F536572766572
MSFT PnP v1.0	com.gemalto.javacard.mspnp	A0000000308000000006DF00FF
Pure DI (version v3.6.0)	com.gemalto.puredi	A000000018320A0100000000000000FF
TachoG2V2	com.gemalto.tacho	A00000003080000000A2800FF
BelPIC v1.8	com.gemalto.belpic	A000000030800000000043417
Privacy Manager v1.0	com.gemalto.javacard.eid	A0000000308000000008DB00FF
MPCOS v4.1	com.gemalto.mpcos	A0000000183003010000000000000FF
FIDO Authentificator 2.1	com.gemalto.javacard.fido.ctap	A00000003080000000A9A00FF

- ➢ Additionally, other applets – not determined at the moment of the present evaluation – may be loaded on the smartcard before or after issuance.

- ➢ A cryptographic library developed by Thales

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

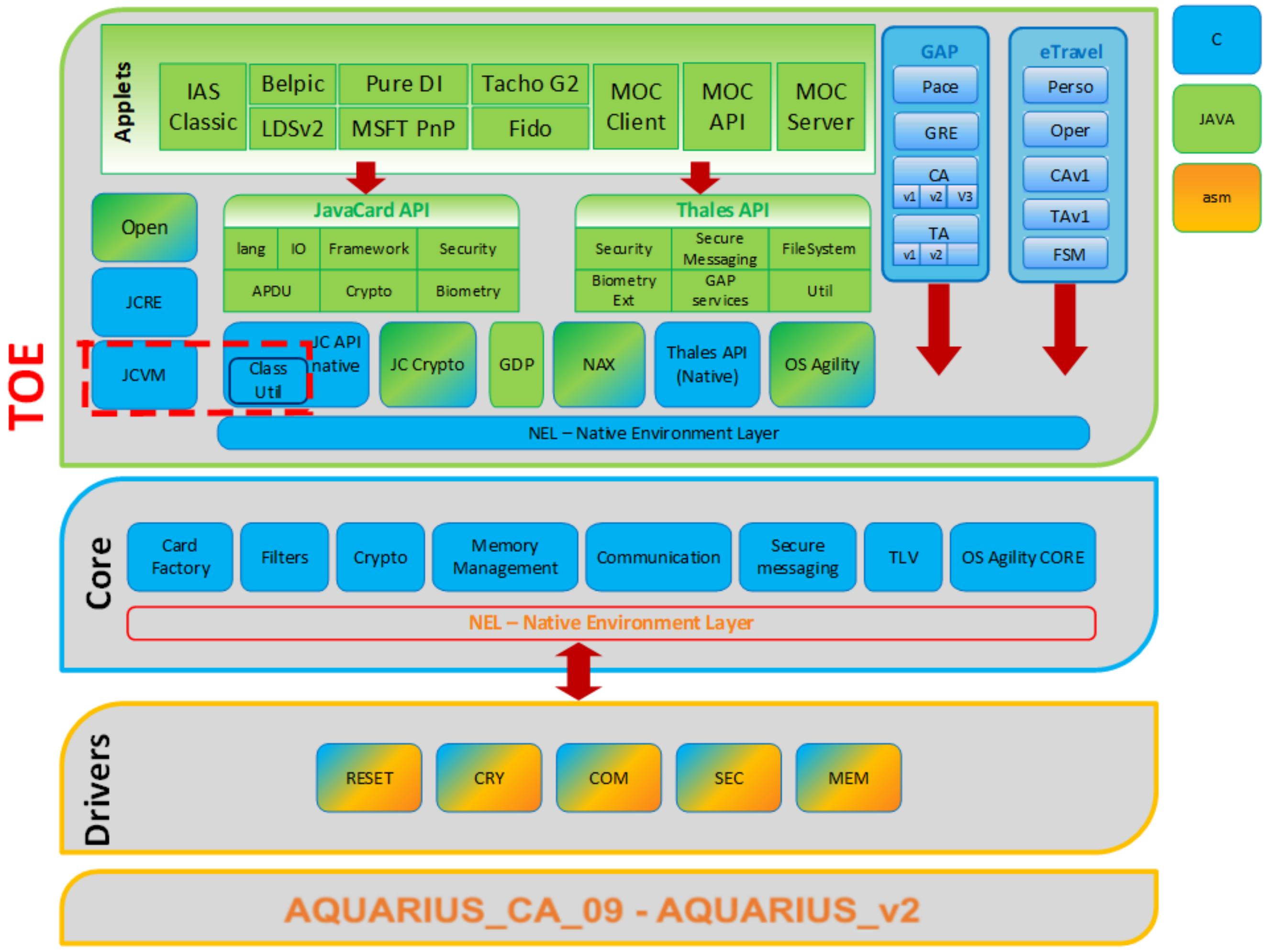

Figure 1: MultiApp V5.1 smartcard architecture

Applets and the MultiApp V5.1 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 [JCRE3].

MultiApp V5.1 products 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 1 shows the TOE boundaries (presented in the red boxes).

2.4 TOE DESCRIPTION

2.4.1 Architecture

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 **interpreter** (used for bytecode interpreting)

All these components have the same version as the embedded software evaluated in [ST-MAV51].

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.

2.5 LIFE-CYCLE

2.5.1 Product Life-cycle

2.5.1.1 Actors

Actors	Identification
Integrated Circuit (IC) Developer	THALES DESIGN SERVICES
Embedded Software Developer (Also named OS developer for the phase 1 of the Life cycle)	Thales DIS (See [ALC-DVS] for details)
Integrated Circuit (IC) Manufacturer	THALES DESIGN SERVICES
Module Manufacturer	THALES (when it is done before the TOE delivery)
Form factor Manufacturer (optional)	THALES (when it is done before the TOE delivery) It can be also an accredited company or the SC Issuer after the TOE delivery
Card manufacturer (Initializer/Pre-personalizer)	THALES (See [ALC-DVS] for details)
Personalization Agent (Personalizer)	The agent who is acting on the behalf of the Issuer (e.g. issuing State or Organization) and personalize the TOE and applicative data (e.g. MRTD for the holder) by activities establishing the identity of the user (e.g. holder with biographic data).
OS Update loader	Agent who is acting on the behalf of the issuer to load the OS patch on the card
Issuer	The Issuer is the actual owner of the SE. As such, no OS Update operation shall be made without his consent. This concept has already been introduced in the SE PP.
Card Holder	The rightful holder of the card for whom the issuer personalizes it.

Table 2: Identification of the actors

2.5.1.2 Life cycle description for

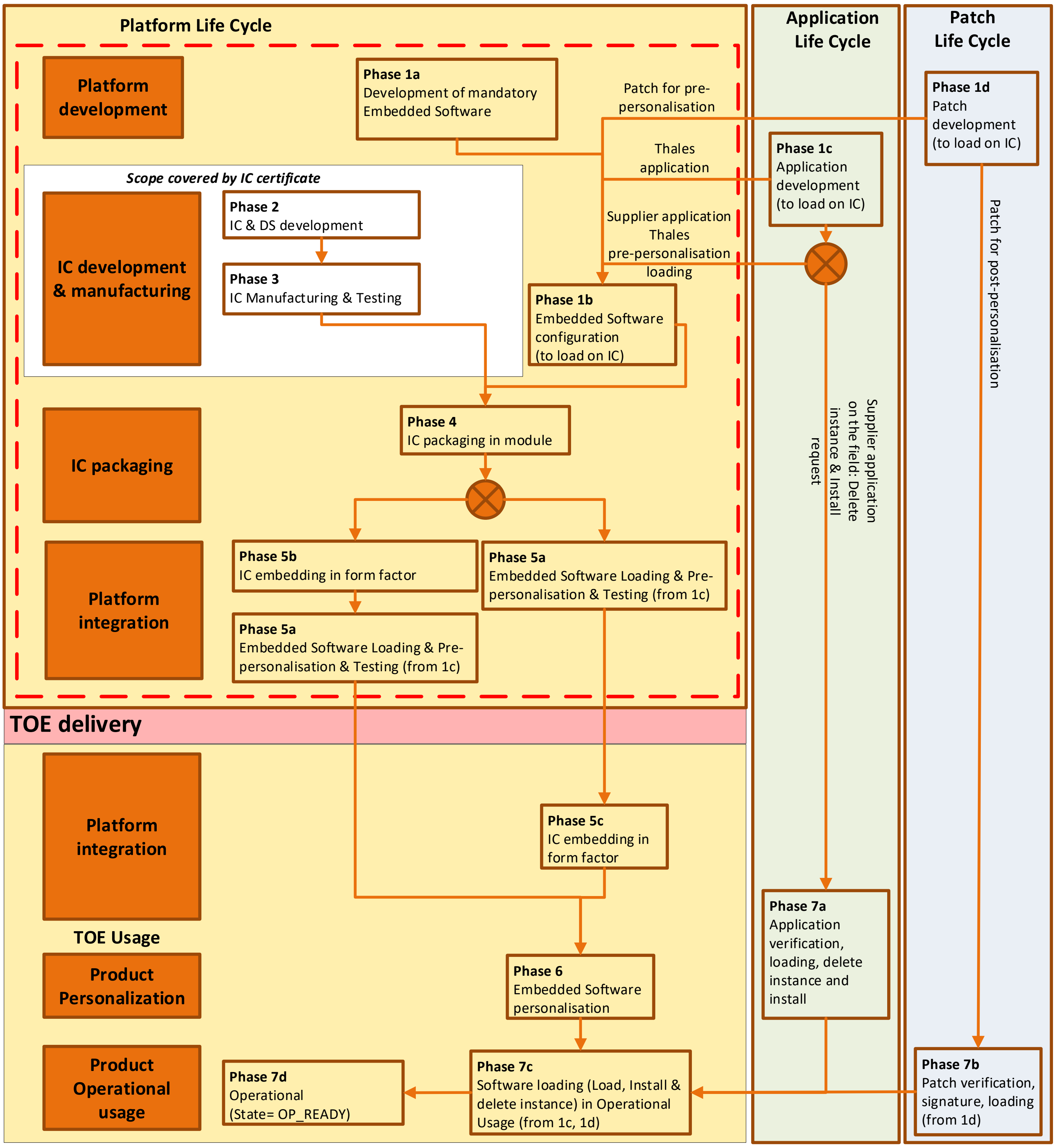
Platform Life Cycle
Application Life Cycle
Patch Life Cycle
Platform development
Phase 1a
Development of mandatory Embedded Software
Patch for pre-personalisation
Phase 1d
Patch development (to load on IC)
Thales application
Phase 1c
Application development (to load on IC)
Scope covered by IC certificate
IC development & manufacturing
Phase 2
IC & DS development
Phase 3
IC Manufacturing & Testing
Supplier application Thales pre-personalisation loading
Phase 1b
Embedded Software configuration (to load on IC)
Patch for post-personalisation
Supplier application on the field: Delete instance & Install request
IC packaging
Phase 4
IC packaging in module
Platform integration
Phase 5b
IC embedding in form factor
Phase 5a
Embedded Software Loading & Pre-personalisation & Testing (from 1c)
Phase 5a
Embedded Software Loading & Pre-personalisation & Testing (from 1c)
TOE delivery
Platform integration
Phase 5c
IC embedding in form factor
TOE Usage
Product Personalization
Phase 6
Embedded Software personalisation
Phase 7a
Application verification, loading, delete instance and install
Product Operational usage
Phase 7d
Operational (State= OP_READY)
Phase 7c
Software loading (Load, Install & delete instance) in Operational Usage (from 1c, 1d)
Phase 7b
Patch verification, signature, loading (from 1d)

Figure 2: Manufacturing phases description

The Life cycle is described on the figure hereunder:

Phase	Description / comments		Who	Where
1	MAV5.1 platform development	Platform development & tests (1.a)	**Thales** R&D team SL Crypto team - secure environment -	Thales Development site (see §2.5.4)
	Thales applets (IAS, eTravel…) development	- Applet Development (1.c) - Applet tests	**Thales** R&D team - secure environment -	Thales Development site (see §2.5.4)
	Patch development	- Patch Development (1.d) - Patch tests	**Thales** R&D team - secure environment -	Thales Development site (see §2.5.4)
	PSE team	- Platform configuration (1.b) - Script development	**Thales Thales Product Engineering**	Thales manufacturing site (see §2.5.4)
2	IC development	Integrated circuits development	**THALES DIS France SAS** - Secure environment -	THALES DIS France SAS development site(s)
3	IC manufacturing & Testing	Manufacturing of virgin integrated circuits embedding the THALES DIS France SAS flash loader, and protected by a dedicated transport key.	**THALES DIS France SAS** - Secure environment -	THALES DIS France SAS development site(s)
4	SC manufacturing: IC packaging & Embedding, also called “assembly”	- IC packaging & testing	**Thales** Production teams - Secure environment -	Thales manufacturing site (see §2.5.4)
5.a	Initialization / Pre-personalization	Loading of the Thales software (platform and applets on top based on script generated)	**Thales** Production teams - Secure environment -	Thales manufacturing site (see §2.5.4)
5.b	Embedding	Put the module on a dedicated form factor (Card, inlay MFF2, other…)		
5c	Embedding	Put the module on a dedicated form factor (Card, inlay MFF2, other…)	SC Issuer or another accredited company	SC Personalizer or Accredited company site
6	SC Personalization	Creation of files and loading of end-user data	SC Issuer or Another accredited company	SC Personalizer or Accredited company site
7	End-usage	Application verification before loading (7.a)	SC Issuer	Field
		Application Loading (Load, Install and delete instance capabilities) (7.c)	SC Issuer	Field
		Patch verification before loading (Signature) (7.b)	Thales	Field
		Patch update (7.b)	Thales	Field
		End-usage for cardholder (7.d)	Cardholder	Field

Figure 3: Life Cycle description

Remark 1: Initialization & pre-personalization operation could be done on module or on other form factor. The form factor does not affect the TOE security.

Remark 2: For initialization/pre-personalization IC flash loader could be used based on the IC manufacturer recommendation.

Remark 3: Embedding (module put on a dedicated form factor) will be done on an audited site if the *Embedding phase (5a & 5b)* is before the TOE delivery.

2.5.2 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.

As a summary description of how the parts of the TOE are delivered to the final customer, the MultiApp V5.1 application is delivered mainly in form of a smart card or inlay. The form factor is packaged on Thales DIS's manufacturing facility and sent to final customer premises.

The different guides accompanying the TOE and parts of the TOE are the ones specified in [AGD] section. They are delivered in form of electronic documents (*.pdf) by Thales DIS's Technical representative.

<u>Note related to patch development</u>

No patch is present within the TOE for the present evaluation. Indeed, should a patch be needed in the future, it would require at least a maintenance of the CC certificate, as required by the CC scheme rules. However, the patch mechanism is part of the TOE and as such its security is assessed within the present evaluation.

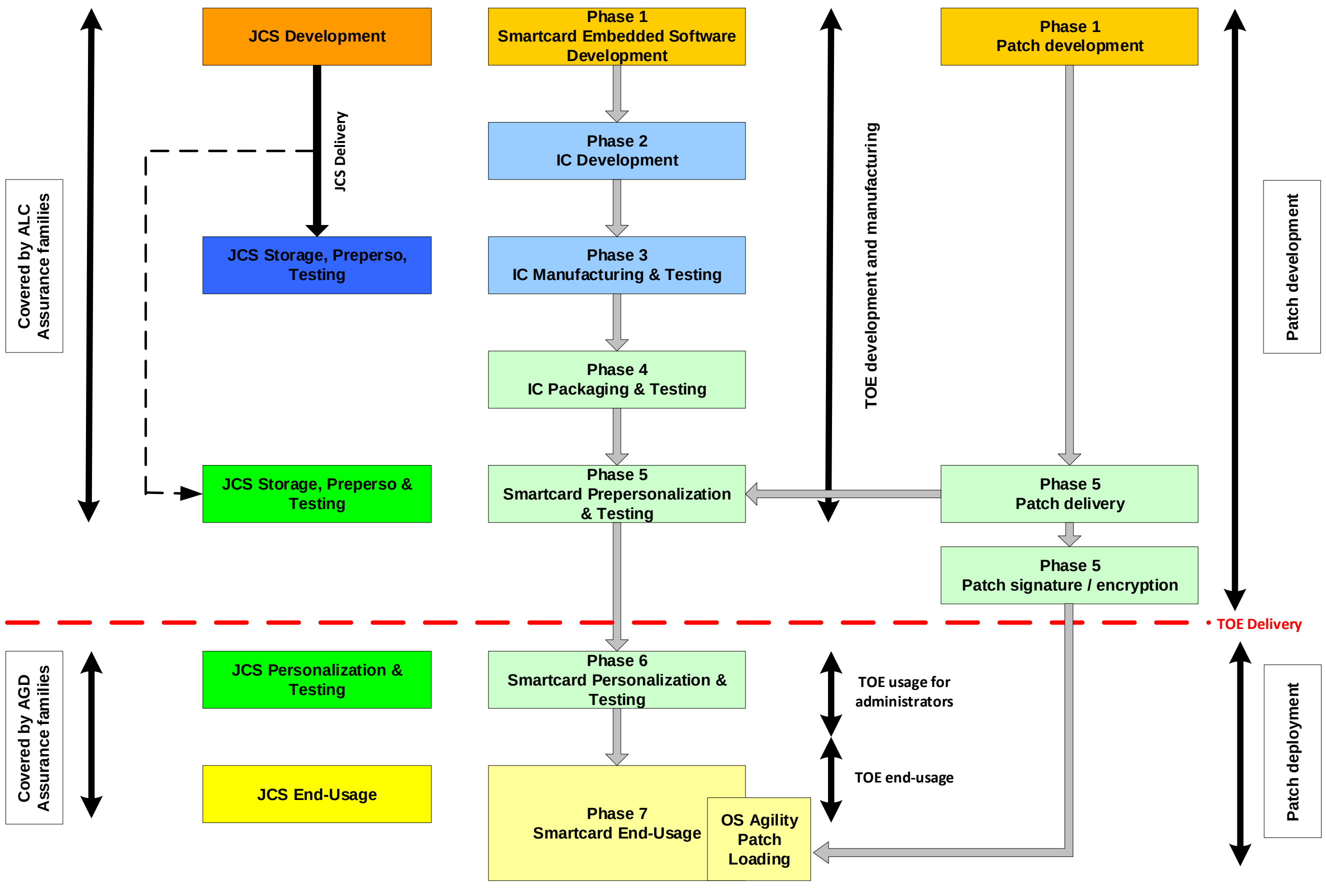
JCS Development
JCS Delivery
JCS Storage, Preperso, Testing
JCS Storage, Preperso & Testing
JCS Personalization & Testing
JCS End-Usage
Covered by ALC Assurance families
Covered by AGD Assurance families
Phase 1 Smartcard Embedded Software Development
Phase 2 IC Development
Phase 3 IC Manufacturing & Testing
Phase 4 IC Packaging & Testing
Phase 5 Smartcard Prepersonalization & Testing
Phase 6 Smartcard Personalization & Testing
Phase 7 Smartcard End-Usage
OS Agility Patch Loading
TOE development and manufacturing
TOE usage for administrators
TOE end-usage
Phase 1 Patch development
Phase 5 Patch delivery
Phase 5 Patch signature / encryption
TOE Delivery
Patch development
Patch deployment

Figure 4: 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.

2.5.3 GP Life-cycle

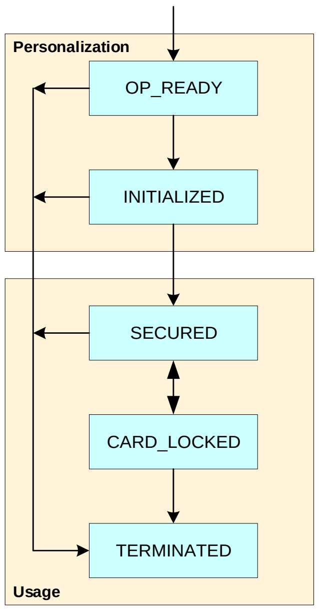

Figure 5: GP Life Cycle

Note that the Patch management (OS-Agility mechanisms) will be available only for the mode:

- OP_READY
- INITIALIZED
- SECURED

2.5.4 Involved Thales-DIS sites

- ❑ **Development and Project Managment**
  - La Ciotat (France)
    - CC project management
  - Singapore
    - Platform & eTravel development
  - Meudon (France)
    - Platform & eTravel development
  - Vantaa
    - Platform & eTravel development support

- ❑ **Manufacturing**
  - Gémenos, Singapore, Vantaa, Tczew, Curitiba, Chanhassen, Pont-Audemer

- ❑ **IT activities**
  - Gémenos, Calamba, Les Clayes, Marcoussi, Pune

2.6 TOE INTENDED USAGE

2.6.1.1 Personalization Phase

During the Personalization Phase the following Administrative Services are available:

- Applet Load
- Applet Install
- Applet Personalization
- Applet Delete
- Applet Extradite
- Applet Management Lock

If the OS Agility is available

- Patch Management

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

2.6.1.2 Usage Phase

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

- Applet Load
- Applet Install
- Applet Personalization
- Applet Delete
- Applet Extradite
- Applet Management Lock

In addition, the following User services are available:

- Applet Selection
- Applet Interface

If the OS Agility is available

- Patch Management

2.6.1.3 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.
- 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-MAV51].

2.6.1.4 NON-TOE HARDWARE/SOFTWARE/FIRMWARE REQUIRED BY THE TOE

The TOE does not include the following components (that are part of the JCRE and JCAPI of the MultiAppv5.1 system):

- Applet selection/deletion/loading
- Object deletion
- Java Card RMI
- Cryptographic API

2.6.1.5 TOE Delivery

As a summary description of how the parts of the TOE are delivered to the final customer, the MultiApp v5.1 embedded software is delivered mainly in form of a smart card, module or wafer. The form factor is packaged on Thales's manufacturing facility and sent to final customer premises or via the wafer init process from the IC Manufacturer premises.

The product is sent to the customer by standard transportation respecting Thales Transport Security Policies.

The different guides accompanying the TOE and parts of the TOE are the ones specified in [AGD] section. They are delivered in form of electronic documents (*.pdf) by Thales's Technical representative via a secure file sharing platform download action.

Item type	Item	Reference/Version	Form of delivery
Software and Hardware	MultiApp v5.1	Refer to paragraph §1.3	Smart card, module or wafer
Document	MultiApp V5.1: AGD_OPE document - Javacard Platform	D1574815 – v1.9 25/08/2023	Electronic document via secure file download
Document	MultiApp V5.1: AGD_PRE document - Javacard Platform	D1574816 – v1.9 25/08/2023	Electronic document via secure file download
Document	MultiApp ID V5 Operating System Reference Manual	D1525385C, December 7th, 2022	Electronic document via secure file download
Document	MultiApp Guidance Document Guidance document for secure development for MultiApp products	D1539156, v1.2 24/03/2023	Electronic document via secure file download

3 CONFORMANCE CLAIMS

3.1 CC CONFORMANCE CLAIM

Common criteria Version:
This ST conforms to CC Version 3.1 revision 5 [CC-1] [CC-2] [CC-3].

Conformance to CC part 2 and 3:
- CC part 2 conformant.
  All the other security requirements have been drawn from the catalogue of requirements in Part 2 [CC-2].
- 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

The MultiApp V5.1 Virtual Machine security target **does not claims strict conformance** to the Protection Profile "JavaCard System – Open configuration", ([PP-JCS-Open]). Also the TOE is part of the embedded software of the MultiAppV5.1 product evaluated in the [ST-MAV51] that has **a "demonstrable" conformance to [PP-JCS-Open]**.

The MultiApp V5.1 JCS security target is a composite security target, including the IC security target [AQU-IC]. However, the security problem definition, the objectives, and the SFR of the IC are not described in this document.

3.3 PACKAGE CLAIM

This ST is conforming to assurance package EAL7.

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-Open]	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	No
FDP_ROL.1/FIREWALL	Yes	Not used
FDP_RIP.1/OBJECTS	Yes	Not used
FMT_MSA.1/JCRE	Yes	Yes
FMT_MSA.1/JCVM	Yes	Yes
FMT_MSA.2/FIREWALL_JCVM	Yes	No
FMT_MSA.3/FIREWALL	Yes	No
FMT_MSA.3/JCVM	Yes	No
FMT_SMF.1/CORE_LC	Yes	Yes
FMT_SMR.1/JCRE	Yes	No

FDP_RIP.1/ABORT	Yes	Not used
FDP_RIP.1/APDU	Yes	Not used
FDP_RIP.1/GlobalArray	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	Not used
FAU_ARP.1	Yes	Not used
FDP_SDI.2/DATA	Yes	Not used
FPR_UNO.1	Yes	Not used
FPT_FLS.1/JCS	Yes	Not used
FPT_TDC.1	Yes	Not used
FIA_ATD.1/AID	Yes	Not used
FIA_UID.2/AID	Yes	Not used
FIA_USB.1/AID	Yes	Not used
FMT_MTD.1/JCRE	Yes	Yes
FMT_MTD.3/JCRE	Yes	Not used
FDP_ITC.2/Installer	Yes	Not used
FMT_SMR.1/Installer	Yes	Not used
FPT_FLS.1/Installer	Yes	Not used
FPT_RCV.3/Installer	Yes	Not used
FDP_ACC.2/ADEL	Yes	Not used
FDP_ACF.1/ADEL	Yes	Not used
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_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_UAU.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
FPT_TST.1/SCP	Yes	Not used
FPT_PHP.3/SCP	Yes	Not used
FPT_RCV.4/SCP	Yes	Not used
FDP_ACC.1/CMGR	Yes	Not used
FDP_ACF.1/CMGR	Yes	Not used
FMT_MSA.1/CMGR	Yes	Not used
FMT_MSA.3/CMGR	Yes	Not used
FPT_FLS.1/SpecificAPI	Yes	Not used
FPT_ITT.1/SpecificAPI	Yes	Not used
FPR_UNO.1/SpecificAPI.	Yes	Not used
FCS_RNG.1	Yes	Not used
FMT_SMR.1/OS-UPDATE	Yes	Not used
FMT_SMF.1/OS-UPDATE	Yes	Not used
FIA_ATD.1/OS-UPDATE	Yes	Not used
FDP_ACC.1/OS-UPDATE	Yes	Not used
FDP_ACF.1/OS-UPDATE	Yes	Not used
FMT_MSA.3/OS-UPDATE	Yes	Not used
FTP_TRP.1/OS-UPDATE	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	Not changed
Security functional requirements	Reduced
Security assurance requirements	Reduced

Table 2 - Compatibility study

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 CAP file in transit to the card. For instance, a CAP file contains the values to be used for initializing the static fields of the CAP file.
#.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 CAP files (that is, controlled execution contexts) as well as between CAP files 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 [JCVM3], [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 [JCVM3] 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 CAP file required by the loaded applet is indeed on the card, in a binary-compatible version (binary compatibility is explained in [JCVM3] §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 CAP files 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 CAP file verification and CAP file installation.

Once a verification authority has verified the CAP file, it signs it and sends it to the card. Prior to the installation of the CAP file, the card verifies the signature of the CAP file, 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 CAP file references only CAP files 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.

5 SECURITY PROBLEM DEFINITION

5.1 ASSETS

The assets of the TOE are part of those defined in [PP-JCS-Open]. The assets of [PP-IC-0084] are studied in [AQU-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, 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 and the PIN security attributes (PIN Try limit, PIN Try counter and State). 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 FROM JAVA CARD SYSTEM PROTECTION PROFILE – OPEN CONFIGURATION

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, ARRAY_VIEWS_CONF, 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 CAP file 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, ARRAY_VIEWS_INT, D.PIN,** and **D.APP_KEYs**.

T.INTEG-APPLI-DATA.LOAD

The attacker modifies (part of) the initialization data contained in an application CAP file when the CAP file 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**.

5.3 Organizational Security Policies

This section describes the organizational security policies to be enforced with respect to the TOE environment. This organizational security policies is coming from the Java Card System Protection Profile – Open Configuration

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.

5.4 ASSUMPTIONS

This section introduces the assumptions made on the environment of the TOE. This assumption is coming from the Java Card System Protection Profile – Open Configuration

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.

A.CAP_FILE

CAP Files loaded post-issuance do not contain native methods. The Java Card specification explicitly "does not include support for native methods" ([JCVM3], §3.3) outside the API.

6 SECURITY OBJECTIVES

6.1 SECURITY OBJECTIVES FOR THE TOE

This section defines the security objective from the Java Card System Protection Profile Open Configuration to be achieved by the TOE.

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

O.ARRAY_VIEWS_CONFID
The TOE shall ensure that no application can read elements of an array view not having array view security attribute ATTR_READABLE_VIEW.
The TOE shall ensure that an application can only read the elements of the array view within the bounds of the array view.

O. ARRAY_VIEWS_INTEG
The TOE shall ensure that no application can write to an array view not having array view security attribute ATTR_WRITABLE_VIEW.
The TOE shall ensure that an application can only write within the bounds of the array view.

O.GLOBAL_ARRAYS_INTEG
The TOE shall ensure that no application can store a reference to the APDU buffer, a global byte array created by the user through makeGlobalArray method and the byte array used for invocation of the install method of the selected applet.

6.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

6.2.1 Security Objectives for the Operational Environment from Java Card System Protection Profile – Open Configuration

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.CAP_FILE
No CAP file 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.

6.2.2 Supplementary security objectives for the operational environment

6.2.2.1 *Identification*

OE.SID
All the subject (applet, or CAP file) should be uniquely identify before granting it access to any service.

6.2.2.2 *Execution*

OE.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.

OE.OPERATE
It should be ensure the continued correct operation of the security functions. See #.OPERATE for details.

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

6.2.2.3 *Applet management*

OE.LOAD
The loading of a CAP file into the card should be safe.
Besides, for codes loaded post-issuance, the TOE shall verify the integrity and authenticity evidences generated during the verification of the application CAP file by the verification authority. This verification by the TOE shall occur during the load or late during the install process.**OE.INSTALL**
The installation of an applet performs should process as expected. (See #.INSTALL for details).
Besides, for codes loaded post-issuance, The integrity and authenticity evidences generated during the verification of the application CAP file should be verify by the verification authority. If it is not performed during the loading process, this verification should occur during the install process.

6.2.2.4 *Services*

OE.ALARM
Appropriate feedback information upon detection of a potential security violation shall be provided. See #.ALARM for details.

6.2.2.5 *CMGR*

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.

6.2.2.6 SCP

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
The SCP shall support the TSFs of the TOE.
This security objective for the environment refers to the security aspects 2, 3, 4 and 5 of #.SCP:
(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 CAP files 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).

6.3 SECURITY OBJECTIVES RATIONALE

6.3.1 Security objectives rationale from JCS Protection Profile – Open Configuration

	O.FIREWALL	O.GLOBAL_ARRAYS_INTEG	O.ARRAY_VIEW_CONFID	O.ARRAY_VIEWS_INTEG	OE.NATIVE	OE.CARD_MANAGEMENT	OE.OPERATE	OE.SID	OE.ALARM	OE.LOAD	OE.SCP.RECOVERY	OE.SCP.SUPPORT	OE.REALLOCATION	OE.INSTALL	OE.VERIFICATION	OE.CAP_FILE	OE.CODE-EVIDENCE
T.CONFID-JCS-CODE					X	X									X		
T.CONFID-APPLI-DATA	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
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			
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	
OSP.VERIFICATION										X					X		X
A.CAP_FILE																X	
A.VERIFICATION															X		X

Table 3: Threats, OSP, Assumptions vs Security Objectives

6.3.1.1 *Threats*

6.3.1.1.1 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 of environment (OE.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 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 (OE.OPERATE) objective of environment.
As the firewall is a software tool automating critical controls, the objective of environment OE.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 objectives of environment OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the OE.OPERATE and OE.ALARM objectives of environments of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
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.
An applet might share data buffer with another applet using array views without the array view security attribute ATTR_READABLE_VIEW. The disclosure of data of the applet creating the array view is prevented by the security object O.ARRAY_VIEWS_CONFID.
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.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 (OE.OPERATE) objective of environment.
As the firewall is a software tool automating critical controls, the objective OE.ALARM asks for it to provide clear warning and error messages, so that the appropriate counter-measure can be taken.
The objectives of environment 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 objectives OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the OE.OPERATE and OE.ALARM objectives of environment of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
Integrity

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 (OE.OPERATE) objective of environment.
As the firewall is a software tool automating critical controls, the objective OE.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 OE.OPERATE and OE.ALARM objectives of environment of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.
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 objective O.GLOBAL_ARRAYS_INTEG.
An applet might share data buffer with another applet using array views without the array view security attribute ATTR_WRITABLE_VIEW. The integrity of data of the applet creating the array view is ensured by the security objective O.ARRAY_VIEWS_INTEG.
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 (OE.OPERATE) objective.
As the firewall is a software tool automating critical controls, the objective OE.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 OE.OPERATE and OE.ALARM objectives of environment of the TOE, so they are indirectly related to the threats that these latter objectives contribute to counter.

6.3.1.1.2 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 objective of environment (OE.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 objectives of environment 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 of environment 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 of environment (OE.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 of environment 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 of environment 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-CODE.LOAD This threat is countered by the security objective of environment OE.LOAD which ensures that the loading of CAP file is done securely and thus preserves the integrity of CAP file 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 of environment 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 CAP file 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.

6.3.1.1.3 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 OE.INSTALL.
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 (OE.OPERATE).
The objective OE.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 of environment OE.SCP.RECOVERY and OE.SCP.SUPPORT are intended to support the OE.OPERATE objective of environment of the TOE, so they are indirectly related to the threats that this latter objective contributes to counter.

6.3.1.1.4 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 OE.NATIVE which ensures that a Java Card applet can only access native methods indirectly that is, through an API. OE.CAP_FILE also covers this threat by ensuring that no CAP files containing native code 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).

6.3.1.2 Organizational Security Policies

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, and by the security objective of the environment OE.LOAD which shall ensure that the loading of a CAP file into the card is safe.

6.3.1.3 *Assumptions*

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

6.3.2 Compatibility between objectives of the TOE and objectives of [AQU-IC]

...
The O.FIREWALL and O.GLOBAL_ARRAYS_INTEG are objectives specific to the Java Card platform and it do no conflict with the objectives of [AQU-IC]. We can therefore conclude that the objectives for the TOE and [AQU-IC] are consistent.

6.3.3 Compatibility between objectives for the environment

OE.VERIFICATION, OE.CAP_FILE, OE.CODE-EVIDENCE, OE.SID, OE.NATIVE, OE.OPERATE, OE.REALLOCATION, OE.LOAD, OE.INSTALL, OE.ALARM, OE.CARD-MANAGEMENT, OE.SCP.RECOVERY, OE.SCP.SUPPORT are objectives specific to the Java Card platform and they do no conflict with the objectives of [AQU-IC].

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

7 SECURITY REQUIREMENTS

7.1 SECURITY FUNCTIONAL REQUIREMENTS

For this section, a presentation choice has been selected. Each SFR may present a table with different type of algorithms treated. For each case, there is no distinction regarding the technical objectives fulfilled by each row on the table (thus algorithm family). The technical objectives are the same disregarding this differentiation.

7.2 SECURITY ASSURANCE REQUIREMENTS

The security assurance requirement level is EAL7.
The list of all the security assurance requirements for this security target is defined in the Table 4: Assurance Level 7 (EAL7).
The entry "EAL7" means that this requirement is defined in the CC part 5
The entry "EAL7/PP" means that requirement is defined in both [CC-3] part and in [PP-JCS-Open] (or linked)
The entry "ST" means that the requirement is defined in this security target.

SAR	Title	Required by
ADV: Development	ADV_ARC.1 Security architecture description	EAL7 / PP
	ADV_FSP.6 Complete semi-formal functional specification with additional formal specification	EAL7
	ADV_IMP.2 Complete mapping of the implementation representation of the TSF	EAL7
	ADV_INT.3 Minimally complex internals	EAL7
	ADV_SPM.1 Formal TOE security policy model	EAL7
	ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation	EAL7
AGD: Guidance documents	AGD_OPE.1 Operational user guidance	EAL7 / PP
	AGD_PRE.1 Preparative procedures	EAL7 / PP
ALC: Life-cycle support	ALC_CMC.5 Advanced support	EAL7
	ALC_CMS.5 Development tools CM coverage	EAL7
	ALC_DEL.1 Delivery procedures	EAL7 / PP
	ALC_DVS.2 Sufficiency of security measures	EAL7
	ALC_LCD.2 Measurable life-cycle model	EAL7 / PP
	ALC_TAT.3 Compliance with implementation standards - all parts	EAL7
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims	EAL7 / PP
	ASE_ECD.1 Extended components definition	EAL7 / PP
	ASE_INT.1 ST introduction	EAL7 / PP
	ASE_OBJ.2 Security objectives	EAL7 / PP
	ASE_REQ.2 Derived security requirements	EAL7 / PP
	ASE_SPD.1 Security problem definition	EAL7 / PP
	ASE_TSS.1 TOE summary specification	**ST**
ATE: Tests	ATE_COV.3 Rigorous analysis of coverage	EAL7
	ATE_DPT.4 Testing: implementation representation	EAL7
	ATE_FUN.2 Ordered functional testing	EAL7
	ATE_IND.3 Independent testing - complete	EAL7 / PP
AVA: Vulnerability assessment	AVA_VAN.5 Advanced methodical vulnerability analysis	EAL7

Table 4: Assurance Level 7 (EAL7)

Among the set of assurance components chosen for EAL7, the assignment appears only in

ADV_SPM.1. The assignment used in ADV_SPM.1 is defined as follows:

ADV_SPM.1	**Formal TOE security policy model** Dependencies: ADV_FSP.4

Developer action elements:

ADV_SPM.1.1D The developer shall provide a formal security policy model for the **Virtual Machine Access Policy**:

- Access Control Policy: FDP_ACC.2/FIREWALL, FDP_ACF.1/FIREWALL
- Flow control: FDP_IFC.1/JCVM, FDP_IFF.1/JCVM
- Security Attributes: FMT_MSA.1/JCRE, FMT_MSA.1/JCVM, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM
- Security roles: FMT_SMR.1/JCRE
- Management Functions: FMT_SMF.1/CORE_LC
- TSF Data: FMT_MTD.1/JCRE

Note: For this formal modelisation, we focus on JCVM opcode processing. The Applet Install, Delete and APIs are out the scope of this modelisation. The initial settings (the Selected Applet Context and the initial active applet) are also out of the scope because done before the JCVM entering (selection of the applet).

Note: For this formal modelisation, the SPM scope will be considering one VM execution

ADV_SPM.1.2D	For each policy covered by the formal security policy model, the model shall identify the relevant portions of the statement of SFRs that make up that policy.
ADV_SPM.1.3D	The developer shall provide a formal proof of correspondence between the model and any formal functional specification.
ADV_SPM.1.4D	The developer shall provide a demonstration of correspondence between the model and the functional specification.

The other SFRs of the MultiApp V5.1 defined in [ST-MAV51] and linked to O.FIREWALL are not linked to this TOE: FMT_MSA.1/ADEL, FMT_MSA.3/ADEL, FMT_SMR.1/ADEL, FMT_SMF.1/ADEL, FMT_MSA.1/CM, FMT_MSA.3/CM, FMT_SMR.1/CM, FMT_SMF.1/CM, FDP_ITC.2/Installer, FMT_SMR.1/Installer, FMT_SMR.1, FMT_SMF.1, are out of the scope of the SPM as they are linked to the applet loading or deletion that is out of scope of the SPM boundaries limited to VM opcodes

The SFR FMT_MTD.3/JCRE is out of scope of the SPM modelisation because AID registry is created during loading phase, which is also out of scope of the SPM (Hypothesis 2 of the SPM document [MAV51_SPM]).

7.2.1 Security Functional Requirements from PP Java Card System – Open configuration

This section states the security functional requirements for the Java Card System – Open configuration.

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-JCS-Open]. (cf Java Card System 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.
Applet deletion (ADELG)	The ADELG contains the security requirements for erasing installed applets from the card, a feature introduced in Java Card specification version 2.2.
Object deletion (ODELG)	The ODELG contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This is a Java Card specification version 2.2 feature.
Secure carrier (CarG)	The CarG group contains minimal requirements for secure downloading of applications on the card. This group contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic bytecodes verification, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.
Smart Card Platform (SCPG)	The SCPG group contains the security requirements for the smart card platform, that is, operating system and chip that the Java Card System is implemented upon.
Card Manager (CMGRG)	The CMGRG group contains the security requirements for the card manager.
Additional SFR (ASFR)	The ASFR group contains security requirements related to specific API and to random generation

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.ADEL	The applet deletion manager which also acts on behalf of the card issuer. It may be an applet ([JCRE22], §11), but its role asks anyway for a specific treatment from the security viewpoint.
S.APPLET	Any applet instance.
S.BCV	The bytecode verifier (BCV), which acts on behalf of the verification authority who is in charge of the bytecode verification of the CAP files.
S.CAD	The CAD represents off-card entity that communicates with the S.INSTALLER.
S.INSTALLER	The installer is the on-card entity which acts on behalf of the card issuer. This subject is involved in the loading of CAP files and installation of applets.
S.JCRE	The runtime environment on which Java programs in a smart card are executed.
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.CAP_FILE	A CAP file may contain multiple Java language packages. A package is a namespace within the Java programming language that may contain classes and interfaces. A CAP file may contain packages that define either a user library, or one or several applets. A CAP file compliant with Java Card Specifications version 3.1 may contain multiple Java language packages. An EXTENDED CAP file as specified in Java Card Specifications version 3.1 may contain only applet packages, only library packages or a combination of library packages. A COMPACT CAP file as specified in Java Card Specifications version 3.1 or CAP files compliant to previous versions of Java Card Specification, MUST contain only a single package representing a library or one or more applets.

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

Object	Description
O.APPLET	Any installed applet, its code and data.
O.CODE_CAP_FILE	The code of a CAP file, including all linking information. On the Java Card platform, a CAP file is the installation unit.
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.APDU	Any APDU sent to or from the card through the communication channel.
I.DATA	JCVM Reference Data: objective 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.
Applet Selection Status	"Selected" or "Deselected"
Applet's version number	The version number of an applet indicated in the export file
Class	Identifies the implementation class of the remote object.
Context	CAP file AID, or "Java Card RE"
Currently Active Context	CAP file AID, or "Java Card RE"
Dependent package AID	Allows the retrieval of the package AID and Applet's version number ([JCVM3], §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 CAP file (library) where it has been defined (these latter objects can only be arrays that initialize static fields of the CAP file). The owner of a remote object is the applet instance that created the object.
CAP File AID	The AID of a CAP file.
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
Resident CAP files	The set of AIDs of the CAP files already loaded on the card.
Selected Applet Context	CAP File AID, or "None"
Sharing	Standards, SIO, Arraw view, Java Card RE entry point, or global array
Static References	Static fields of a CAP file 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.ARRAY_LENGTH (O.JAVAOBJECT, field)	Get length of an array component.
OP.ARRAY_AASTORE(O.JAVAOBJECT, field)	Store into reference array component
OP.ARRAY_T_ALOAD(O.JAVAOBJECT, field)	Read from an array component
OP.ARRAY_T_ASTORE(O.JAVAOBJECT, field)	Write to an array component
OP.CREATE(Sharing, LifeTime) (*)	Creation of an object (new or makeTransient or createArrawView call).
OP.DELETE_APPLET(O.APPLET,...)	Delete an installed applet and its objects, either logically or physically.
OP.DELETE_CAP_FILE(O.CODE_CAP_FILE,...)	Delete a CAP file, either logically or physically.
OP.DELETE_CAP_FILE_APPLET(O.CODE_CAP_FILE,...)	Delete a CAP file and its installed applets, either logically or physically.
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 [JCRE3], §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 (throw, see [JCRE3],§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).

(*) 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.

7.2.1.1 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.

7.2.1.1.1 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.CAP_FILE, 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.
- OP.ARRAY_LENGTH
- OP.ARRAY_T_ALOAD
- OP.ARRAY_T_ASTORE
- OP.ARRAY_AASTORE

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.CAP_FILE	**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 ([JCRE3]§6.2.8) An S.CAP_FILE 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 ([JCRE3]§6.2.8) An S.CAP_FILE 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 ([JCRE3]§6.2.8.10) An S.CAP_FILE may perform OP.TYPE_ACCESS upon an O.JAVAOBJECT with Context attribute different from the currently active context, 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 ([JCRE3], §6.2.8.6,) An S.CAP_FILE may perform OP.INVK_INTERFACE upon an O.JAVAOBJECT with Context attribute different from the currently active context, whose Sharing attribute has the value "SIO", and whose Context attribute has the value "CAP file 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 CAP file whose AID is "Package AID" is "Multiselectable»,

(b) The value of the attribute Selection Status of the CAP file whose AID is "Package AID' is "Non-multiselectable», and either "CAP file AID" is the value of the currently selected applet or otherwise "CAP file AID" does not occur in the attribute ActiveApplets.

- **R.JAVA.5 An S.CAP_FILE may perform an OP.CREATE only if the value of the Sharing parameter(*) is "Standard".**
- **R.JAVA.6 ([JCRE3], §6.2.8): S.CAP_FILE may freely perform OP.ARRAY_ACCESS or OP.ARRAY_LENGTH upon any O.JAVAOBJECT whose Sharing attribute has value "global array".**

  Application Note (R.JAVA.4): The initial setting of security attributes ActiveApplets and Selected Applet Context are initialized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets and Selected Applet Context are never changed in the VM scope.

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: This rule is out of scope of the SPM modelisation because CLEAR_ON_DESELECT objects can be created exclusively in the API, which is also out of scope (Hypothesis 4 of the SPM document [MAV51_SPM])..

3) **S.CAP_FILE performing OP.ARRAY_AASTORE of the reference of an O.JAVAOBJECT whose sharing attribute has value "global array" or "Temporary JCRE entry point".**
4) **S.CAP_FILE performing OP.PUTFIELD or OP.PUTSTATIC of the reference of an O.JAVAOBJECT whose sharing attribute has value "global array" or "Temporary JCRE entry point"**
5) **R.JAVA.7 ([JCRE3], §6.2.8.2): S.CAP_FILE performing OP.ARRAY_T_ASTORE of the reference of an O.JAVAOBJECT, or a primitive value when the O.JAVAOBJECT is an array view without ATTR_WRITABLE_VIEW access attribute.**
6) **R.JAVA.8 ([JCRE3], §6.2.8.2):S.CAP_FILE performing OP.ARRAY_T_ALOAD of the reference of an O.JAVAOBJECT, or a primitive value when the O.JAVAOBJECT is an array view without ATTR_READABLE_VIEW access attribute.**

Application note: FDP_ACF.1.4/FIREWALL:

The initial setting of security attribute Selected Applet Context is initilized by SELECT APDU, which is out of SPM scope. Selected Applet Context is never changed in the VM scope.

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

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 five 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.
- Array Views, having fields/elements access controlled by access control attributes, ATTR_READABLE_VIEW and ATTR_WRITABLE_VIEW 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 ([JCRE3], §6.1.3). An object is owned by an applet instance, by the JCRE or by the CAP file library where it has been defined (these latter objects can only be arrays that initialize static fields of CAP file).

([JCRE3], 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 (CAP file AID) of the selected applet. In this policy, the "Selected Applet Context" is the AID of the selected CAP file.
([JCRE3], §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).

It should be noticed that the invocation of static methods (or access to a static field) is not considered by this policy, as there are no firewall rules. They have no effect on the active context as well and the "acting CAP file" 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 CAP file being selected simultaneously. These applets are referred to as multiselectable applets. Applets that belong to a same CAP file are either all multiselectable or not ([JCVM3], §2.2.5). Therefore, the selection mode can be regarded as an attribute of CAP file. No selection mode is defined for a library CAP file.

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. ([JCRE3], §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.DATA) 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 ([JCRE3], §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** 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 [JCRE3], §4 and [JCVM3], §3.4.

The initial setting of security attribute the Selected Applet Context is initialized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The the Selected Applet Context is never changed in the VM scope.

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 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 [JCRE3], §4 and [JCVM3], §3.4.
The initial setting of security attribute ActiveApplets is initilized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets is never changed in the VM scope.

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**.

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[Editorially Refined] 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 (FMT_MSA.1/JCRE). 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 ([JCRE3], §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[Editorially Refined] The TSF shall not allow **any role** to specify alternative initial values to override the default values when an object or information is created.

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**

Note: the Selected Applet context is out of scope of the VM functionalities. It is a process that occurs prior to VM start

The initial setting of security attributes ActiveApplets and Selected Applet Context are initilized by SELECT APDU and MANAGE_CHANNEL, which are out of SPM scope. The ActiveApplets and Selected Applet Context are never changed in the VM scope.

7.2.1.1.2 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.

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, applet deletion manager or even the card manager 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).
- **The DELETE and INSTALL APDU commands are out of scope of this SPM. The list of registred applets' AIDs is proven to be not modified during the execution inside the VM.**

7.3 Security Requirements Rationale

7.3.1 OBJECTIVES for PP JCS – OPEN Configuration

	O.FIREWALL	O.GLOBAL_ARRAYS_INTEG	O.ARRAY_VIEWS_CONFID	O.ARRAY_VIEWS_INTEG
FDP_IFC.1/JCVM	X	X	X	X
FDP_IFF.1/JCVM	X	X	X	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			
FMT_MTD.1/JCRE	X			
FDP_ACC.2/FIREWALL	X			
FDP_ACF.1/FIREWALL	X			
FMT_MSA.1/JCRE	X			
FMT_MSA.1/JCVM	X			

Table 5: rationale objective vs. SFR

7.3.1.1 SECURITY OBJECTIVES FOR THE TOE

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 (FMT_MSA.1/JCVM, FDP_IFF.1/JCVM, FDP_IFC.1/JCVM). The functional requirements of the class FMT (FMT_MTD.1/JCRE,FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC, FMT_MSA.2/FIREWALL_JCVM, FMT_MSA.3/FIREWALL, FMT_MSA.3/JCVM, FMT_MSA.1/JCRE) also indirectly contribute to meet this objective.

O.GLOBAL_ARRAYS_INTEG This objective is met by the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM), which prevents an application from keeping a pointer to the APDU buffer of the card, to the global byte array of the applet's install method or to the global arrays created by the JCSystem.makeGlobalArray(…) method. Such a pointer could be used to access and modify it when the buffer is being used by another application.

O.ARRAY_VIEWS_CONFID Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_READABLE_VIEW security attribute which ensures that no application can read the contents of the array view.

O.ARRAY_VIEWS_INTEG Array views have security attributes of temporary objects where the JCVM information flow control policy (FDP_IFF.1/JCVM, FDP_IFC.1/JCVM) prevents an application from storing a reference to the array view. Furthermore, array views may not have ATTR_WRITABLE_VIEW security attribute which ensures that no application can alter the contents of the array view.

7.3.2 DEPENDENCIES for PP JCS-OPEN CONFIGURATION

7.3.2.1 SFRS DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
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
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.1/JCRE	(FDP_ACC.1 or FDP_IFC.1), FMT_SMF.1, FMT_SMR.1	FDP_IFC.1/JCVM, FMT_SMR.1/JCRE, FMT_SMF.1/CORE_LC, FDP_ACC.2/FIREWALL
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_SMR.1/JCRE	FIA_UID.1	None(*)
FMT_SMF.1/CORE_LC	none	

Table 6: SFR dependencies

(*) The User identification is the responsibility of the Platform MultiApp V5.1. This identification is done before acceding the Virtual Machine thanks to the SFR: FIA_UID.2/AID User identification before any action (see [ST_MAV5]

7.3.3 Compatibility between SFR of TOE and SFR of [AQU-IC]

The following table lists the SFRs that are declared on the [AQU-IC] Integrated Circuit Security Target [AQU-IC] and separates them in:

IP_SFR: Irrelevant Platform-SFRs not being used by the Composite-ST.
RP_SFR-SERV: Relevant Platform-SFRs being used by the Composite-ST to implement a security service with associated TSFI.
MRP_SFR-MECH: Relevant Platform-SFRs being used by the Composite-ST because of its security properties providing protection against attacks to the TOE as a whole and are addressed in ADV_ARC. These required security properties are a result of the security mechanisms and services that are implemented in the Platform TOE, as specified in [JIL_CPE].

These definitions are according to the [JIL_CPE] on which the Platform TOE on our case is the relaying IC, the [AQU-IC] Integrated Circuit.
The first column lists the [AQU-IC] and the next columns indicate their classification according to the paragraph above. The SFR's on the cells of the classification belong the MultiApp v4.2 TOE described in this document. If there is no SFR on each cell is because not all CC class families have a corresponding match on both sides, but all SFRs from the [AQU-IC] have been classified. Moreover, no contradictions have been found between the Platform-SFRs set and the SFRs related to the composite product.

AQUARIUS_CA_09 SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
Security functional requirements of the TOE defined in [PP0084]			
FRU_FLT.2			X FDP_SDI.2/DATA FPT_PHP.3/SCP
FPT_FLS.1			X FPT_FLS.1/JCS FPT_FLS.1/Installer FPT_FLS.1/ADEL FPT_FLS.1/ODEL FPT_FLS.1/SpecificAPI
FMT_LIM.1		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FAU_SAS.1	X		
FDP_SDC.1			X *see table 27
FDP_SDI.2			X FDP_SDI.2/DATA
FPT_PHP.3			X FPT_PHP.3/SCP
FDP_ITT.1			X *see table 27

AQUARIUS_CA_09 SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FPT_ITT.1		X FPT_ITT.1/SpecificAPI	
FDP_IFC.1			X FDP_IFC.1/JCVM
FCS_RNG.1/PTG.2			x FCS_RNG.1
FMT_LIM.1/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FMT_LIM.2/Loader		X FMT_LIM.1/PERSO FMT_LIM.2/PERSO	
FTP_ITC.1		X FTP_ITC.1/CM FTP_ITC.1/PACE	
FDP_UCT.1		X FDP_ACC.1.1/CMGR	
FDP_UIT.1			X FDP_UIT.1/CM
FDP_ACC.1/Loader		X FDP_ACC.1.1/CMGR	
FDP_ACF.1/Loader		X FDP_ACF.1.1/CMGR	
FDP_ACC.1/Memory		X FDP_ACC.1.1/CMGR	
FDP_ACF.1/ Memory		X FDP_ACF.1.1/CMGR	
FIA_API.1			X FIA_UID.2/AID FIA_UAU.1/CM FIA_UID.1/CM
Additional security functional requirements of the TOE			
FMT_MSA.1		X FMT_MSA.1/ADEL FMT_MSA.1/JCRE FMT_MSA.1/JCVM FMT_MSA.1/CM FMT_MSA.1/CMGR	

AQUARIUS_CA_09 SFR's	IP_SFR	RP_SFR-SERV (*)	RP_SFR-MECH
FMT_MSA.3		X FMT_MSA.3/FIREWALL FMT_MSA.3/JCVM FMT_MSA.3/ADEL FMT_MSA.3/CM FMT_MSA.3/CMGR	
FMT_SMF.1		X FMT_SMF.1/CORE_LC FMT_SMF.1/ADEL FMT_SMF.1/CM	

Table 7 Compatibility between SFR of TOE and SFR of [AQU-IC]

7.3.4 SAR DEPENDENCIES

Requirements	CC dependencies	Satisfied dependencies
ADV_ARC.1	ADV_FSP.1; ADV_TDS.1	ADV_FSP.6; ADV_TDS.6
ADV_FSP.6	ADV_TDS.1; ADV_IMP.1	ADV_TDS.6; ADV_IMP.2
ADV_IMP.2	ADV_TDS.3; ALC_TAT.1; 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
ADV_SPM.1	ADV_FSP.4	ADV_FSP.6
AGD_OPE.1	ADV_FSP.1	ADV_FSP.6
AGD_PRE.1	None	
ALC_CMC.5	ALC_CMS.1; ALC_DVS.2; ALC_LCD.1	ALC_CMS.5; ALC_DVS.2; ALC_LCD.2
ALC_CMS.5	None	
ALC_DEL.1	None	
ALC_DVS.2	None	
ALC_LCD.2	None	
ALC_TAT.3	ADV_IMP.1	ADV_IMP.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.6; ADV_IMP.2; ATE_FUN.2
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

Table 8: SAR dependencies

7.3.5 RATIONALE FOR THE SECURITY ASSURANCE REQUIREMENTS

7.3.5.1 *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.

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 SF provided by MultiApp V5.1 platform: 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
The JCVM information flow control SFP to provide restrictive default values for security attributes that are used to enforce the SFP. And not allow any role to specify alternative initial values to override the default values when an object or information is created.	FMT_MSA.3/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 maintains the roles: the Java Card RE, the Java Card VM. The TSF is able to associate users with roles.	FMT_SMR.1/JCRE
The TSF is 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
([JCRE3]§6.2.8) An S.CAP_FILE 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
([JCRE3]§6.2.8) An S.CAP_FILE 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
([JCRE3]§6.2.8.10) An S.CAP_FILE 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
• ([JCRE3], §6.2.8.6,) An S.CAP_FILE 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, and in either of the cases above the invoked interface method extends the Shareable interface	FDP_ACC.2/FIREWALL FDP_ACF.1/FIREWALL

An S.CAP_FILE 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

8.1.1.1 SF.AID: AID Management

Only **the JCRE** can **modify** the **list of registered applets' AIDs**.	FMT_MTD.1/JCRE

8.1.2 TSFs provided by the AQUARIUS_CA_09

The evaluation is a composite evaluation and uses the results of the CC evaluation provided by [CR-IC]. The IC and its primary embedded software have been evaluated at level EAL 6+. These SF are the same for the IC considered in this ST;

SF	**Description**
SF_PMODE	Manages the different steps of the product life cycle. At each step (boot mode, test mode and user mode), registers, data and memories accesses are limited or not. This allows to restrict product access according to the step (from manufacturing phase to final user phase). In addition, it is not possible to come back to test mode after the deployment of the product.
SF_AUDIT_STORAGE	Allows to store specific data which shall remain permanent in the system such as the unique identification of the product stored in the Flash memory, pre-personalization data and security information.
SF_AUTHENT	Provides mutual authentication between the TOE and the "Terminal" based on cryptographic mechanisms. Authentication is done before the loading operation.
SF_CONF_INT	Provides confidentiality and integrity to data stored in the memories (ROM, RAM, FLASH), in registers and in buses. The SF_CONF_INT prevents the disclosure of internal user data thanks to: ▪ Memories encryption. ▪ Buses encryption. ▪ Register masking and cycling. ▪ Address scrambling. ▪ Integrity mechanisms on memories, buses and registers.
SF_EXEC	Provides protection against an un-correct execution of the code such as: ▪ Mechanisms to detect code re-routing. ▪ Mechanisms to detect illegal opcode execution. ▪ Mechanisms to control the operating conditions. In case of detection of an abnormal execution, an alarm is sent. Ensures also the correct operating conditions of the product during the execution and prevents any malfunction using sensors.
SF_MEM_ACCESS	Provides:

	▪ A Memory Protection Unit (MPU) that defines access permission on different memories areas. ▪ A Flash Protection Unit that defines access permission on NVR areas. Provides also an access control to user data stored in Flash during the deployment of the Loader and after.
SF_PHY_PRO	Provides physical protection to the product against physical manipulation and physical probing. The following features are used: ▪ Active Shield. ▪ Countermeasures added during the layout design.
SF_ALARM	Enables to trig either an interrupt or a hardware reset. This TSF provides preservation of secure state in case of exposure to operation conditions which are not tolerated.
SF_RANDOM	Provides mechanisms to prevent access to sensitive assets during the use by the Secure Embedded Software thanks to: ▪ Generate variation of the clock frequency around a range of frequency. ▪ Randomize the clock stealer. Randomize the execution of the commands.
SF_RNG	Provides a random number generator (PTRNG) that meets PTG.2 class of BSI-AIS31 (German Scheme). It is used for key generation or for security measures.
SF_SEC_LOAD	Allows to load some code in the product in a secure way and, after the loading, to lock the loading mechanism.

Table 9: Security Functions provided by the THALES DIS France SAS AQUARIUS_CA_09 chips

These SF are described in [AQU-IC].

END OF DOCUMENT