Overview of the Java Card™ System Protection Profile Collection

Version 1.0

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054

August 2003

This document has been prepared by:

Trusted Logic SA
5, rue du Bailliage
78000 Versailles, France
http://www.trusted-logic.com/

on behalf of Sun Microsystems, Inc.

For any correspondence on this document please contact:

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 USA
http://www.sun.com
JC_PP_feedback@sun.com

Legal Notice

CONTENTS

Figures and Tables

1 INTRODUCTION

This document provides an overview of the Java Card™ System Protection Profile ("JCSPP") Collection [JCSPP]. It presents the main underlying concepts and the methodology that guided the design and writing of the JCSPP Collection document which describes the four Protection Profiles. The reader is assumed to be familiar with the Java Card technology (`http://java.sun.com/products/javacard`/) and the Common Criteria (CC) [CC1] vocabulary and concepts.

One of the challenges of writing a Protection Profile for a Java Card platform is to provide a single description that addresses:

- The different environments where it can be used (such as closed or open platforms).
- The wide range of choices offered by the Java Card technology (such as logical communication channels with the card, remote invocations of services, and object deletion).
- The different security architectures that have been developed for verifying application code (such as off-card verification of application code, embedded verifiers, and defensive bytecode interpreters) and loading the code onto the card.

The answer to this challenge is a modular structure. First we have defined **groups** of requirements for each functional feature proposed by Java Card technology. For example, groups of security requirements address the isolation of application data during execution, bytecode verification, installation and deletion of applications, and transmission of applications to the card.

These groups represent all the functional options. When a Java Card platform is developed, various parameters must be set. For example:

- The platform can be either closed or open.
- The security architecture for application verification and loading must be chosen.
- A decision must be made regarding the Java Card technology features to support.

A particular combination of these security and functional ingredients determines what is called in the JCSPP Collection a **configuration**. More precisely, a configuration is defined, from both a functional and a security point of view, as a combination of groups of security functional requirements.

The design and development of the JCSPP Collection has greatly benefited from the use of groups of requirements. For example, it:

- Enables a modular construction of a substantial part of a configuration.
- Provides an appropriate setting for the derivation of new configurations from existing ones.
- Allows the definition of several protection profiles in one document without introducing redundancies. The groups of security requirements are defined once but are repeatedly used to define different configurations.

Four typical configurations have been identified, and the corresponding PPs have been defined and presented in the JCSPP Collection. Two of these configurations were chosen because they correspond to standard use-cases. The other two cover the largest range of features proposed in the latest

version of Java Card technology. The difference between the configurations is in the way that verification is performed on application code that is loaded onto the card. A section with a comprehensive presentation of each configuration is included in this document.

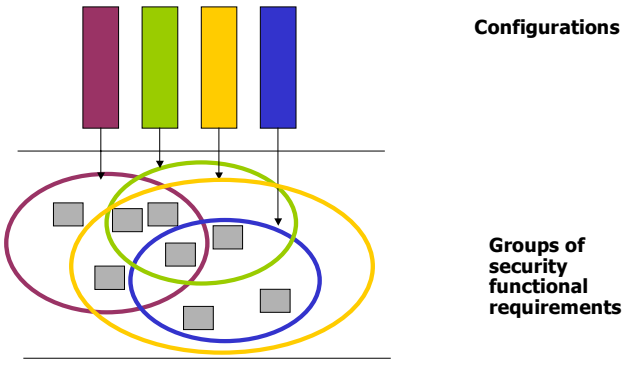

Figure 1. Reutilization of security requirements

The structure of the JCSPP Collection conforms to the general specification of protection profiles defined in the CC specification ([9], appendix B). However, the use of configurations and groups of requirements has naturally led to the definition of a particular sub-structure of the document. When the definition of a CC component depends on the configuration, the structure of the corresponding section introduces a new level of sub-sections, one for each of the configurations. The section *Security Objectives for the TOE*, for instance, is divided into four sub-sections. Each of those sections defines the security objectives proper of one of the configurations. This way of structuring the presentation of the security components is systematically used throughout the JCSPP Collection. The present document includes a detailed description of the structure of the JCSPP Collection.

The rest of this document is organized as follows:

Section 2 introduces the main security-relevant parameters to consider when planning a PP for the Java Card platform.

Section 3 describes the set of configurations of a Java Card platform defined in the JCSPP Collection, and motivates why they were chosen among all the possible configurations. This section also introduces the notion of a group of requirements, and enumerates all the groups necessary to cover several of the possible configurations of a Java Card platform.

Section 4 explains the structure of the JCSPP Collection document.

Section 5 provides guidelines for developing a compliant Security Target (ST), given one of the configurations defined in the JCSPP Collection.

Section 6 lists the references used in this document.

Section 7, the Appendix, presents the results of the work that has been developed to relate the Security Functional Requirements of the JCSPP Collection to the security features of GlobalPlatform and to those of related smart card protection profiles.

2 RISK ANALYSIS OF A JAVA CARD PLATFORM

Risk analysis is the first step towards the definition of the security environment of a Target of Evaluation ("TOE"). A PP can be seen as a document that details the result of such analysis and puts forward sufficient objectives and requirements for the TOE Security Functions ("TSF") to counter the identified threats. Risk analysis is, therefore, the first step in developing a PP.

A risk analysis of a Java Card platform shall identify at least the following parameters:

- The smart card use-cases that currently exist, in particular whether they are closed or open multi-application environments
- The downloading security architectures for open platforms that have been planned
- The collection of optional features or functionalities (API) available in the various specifications of the Java Card platform

Each of these parameters is analyzed in the following sections.

2.1 CLOSED AND OPEN MULTI-APPLICATION CARDS

The Java Card technology is particularly suited for multi-application cards, where several applications co-exist in a single platform. The installed applications may share and exchange data to provide combined services in a more flexible manner.

Most of today's deployed multi-application cards are *closed* cards, where the set of installed applications is fixed once and forever during its production and personalization phase [2]. In these cards, the end-user has no access to card-management features, which are usually disabled prior to card issuance. An example of a closed card is a banking card intended for only two applications: purchasing and electronic wallet.

However, the deployment of *open* cards has already started. An open smart card allows the end-user to load, install, or activate new applications on the card after the card delivery. Downloading new applications may require the agreement of the card issuer.

These two kinds of cards outline two types of runtime environments: those that include application-downloading capabilities, and those that do not. The possibility of loading (typically at post-issuance time) new code on the card introduces several security issues. Thus, the introduction of this functionality in the platform is an important parameter of the risk analysis. The main alternatives are illustrated through the four scenarios.

Note that this choice in the design of the platform depends either on the current application domains (typically from functional and marketing consideration), or on bytecode verification techniques, which enable the downloading of a new piece of code without compromising the security of the other applications on the card.

2.2 BYTECODE VERIFICATION

The Java Card platform provides a multi-application secure runtime environment for smart cards. It allows applications written in a subset of the Java programming language (termed "*applets*") to share data and services within a well-defined security framework. This framework is built upon two main cornerstones: the "Firewall" mechanism and bytecode verification.

The Java Card specification naturally focuses on the runtime of applications, leaving open a number of security issues, mostly concerning the management of *applet*s. Regarding bytecode verification, this can be done either by an off-card IT product before applet downloading, by an embedded verifier immediately after downloading, or at runtime by the interpreter for the Java Card platform. This choice introduces a second security parameter to be fixed while designing a Java Card platform.

Note that the moment of the applet life cycle when the applet is verified plays an important role in the risk analysis of the platform, since it is necessary to ensure that the applet cannot be modified once it has been verified and before it is executed on the card. This means that all the necessary precautions have to be taken to protect the application code and the issuer responsibility when the bytecode verification is performed off-card. Part of this issue is solved when embedded verification can be performed on the card, when the platform has enough resources to support it.

This security parameter results in the refinement of the use-cases identified in the previous section, and to distinguish two kinds of open platforms:

- Standard platforms, which rely on a number of properties of their environment or the installed applications. The platforms that fall into this class are those where bytecode verification is performed before loading, and certified by an electronic signature that is verified later by an embedded card manager. This security architecture represents most of the open cards that are currently deployed.

- Defensive platforms, whose on-card components can be considered autonomous with respect to the smart card environment. This covers those platforms including an on-card bytecode verifier, and those where bytecode verification is performed at runtime. Such platforms are able to implement the whole set of security policies of the specification, including typing rules for the bytecode used on the Java Card platform. They are particularly suited for those applications where the card-user can decide to download new applications independently from the card issuer.

The distinction above is not relevant for closed platforms, because applet installation is usually performed in a secure environment and addressed by organizational policies concerning card management.

2.3 EVOLUTION OF THE JAVA CARD TECHNOLOGY AND OPTIONAL FEATURES

Java Card technology is evolving. Accompanying both the technical evolution of smart card chips and the needs of application developers, new features have been introduced recently, such as remote method invocation (RMI), automated de-allocation of unreachable objects, and the possibility of having several communication channels open between the card and the card reader. In addition, version 2.2 of the Java Card specifications ([JCVM22][JCRE22][JCAPI22]) also introduces some of the previously mentioned card management features, such as applet deletion.

All these new features are introduced in such a way that new versions of a Java Card platform are compatible with applets that have been written for former versions. Each of the new features, in turn, requires the consideration of new security aspects that are specific to it. Therefore, the precise version of Java Card technology implemented by a platform is the third key parameter to be considered in the risk analysis of a Java Card platform. Moreover, some of the new features are sometimes optional, introducing yet a further level of granularity in the security issues of the platform.

Openness	Closed, Open
Bytecode verification	off-card, on-card
Java Card technology functionalities	Version 2.1.1, RMI, Logical channels, Applet installation and deletion, Object deletion

Table 1. Security parameters of the risk analysis

3 CONFIGURATIONS & SECURITY REQUIREMENTS

Each choice relative to one of the three security parameters introduced in the previous section gives rise to a particular TOE environment and a collection of security functional requirements to be imposed on the TOE. According to Table 1, there are several different situations to be addressed. Of course, one possibility could be to create just as many separate PPs. However, this would lead to an unnecessary duplication, since a platform that implements version 2.2 of Java Card technology incorporates all the security requirements of a platform implementing the earlier 2.1.1 version. Moreover, some of the requirements (like those concerning the Firewall security policy) are common to any Java Card platform.

Duplication shall be avoided for several reasons:

- Duplication of documents means a duplication of production and evaluation costs.
- The introduction of each new optional feature in Java Card technology will probably cause the number of PPs and their associated evolution costs to grow exponentially.
- Managing several different versions of a Java Card platform PP sharing common parts may easily lead to undesirable inconsistencies between those versions.

However, the variety of options and use-cases, intrinsic to the commitments of modularity, extensibility, and portability of Java Card technology, does not fit well into the classic structure defined for a PP in the Common Criteria, which was initially tailored for a single, monolithic product.

The approach taken to address this situation was to introduce the notion of a group of security functional requirements and that of a configuration (of the product). These notions allow the definition of a set of security requirements in a modular way, and to provide an adequate setting for the definition of configurable options. Groups and configurations are detailed in the following sections.

3.1 FROM USE CASES TO CONFIGURATIONS

Configurations correspond to a particular instantiation of the security parameters discussed in section 2 of this document.

Each configuration has its own security objectives, which in turn determine the security requirements to be chosen to meet those objectives. Moreover, even if configurations may have the same global collection of objectives, some of them may be objectives for the TOE in one configuration and objectives of the IT environment in another. Therefore, a configuration is described which sets up the precise limits of the TOE, a definite environment for it, and the security requirements to be used.

The following are the configurations, among the several that can be defined, which are addressed in the JCSPP Collection. They have been chosen either because they correspond to existing use-cases, or because they cover the largest range of features of the Java Card platform:

- The **Minimal** configuration corresponds to a multi-application card where no post-issuance downloading of applications is allowed (closed card use-case). The TOE is the simplest

runtime environment for a Java Card system. The IT environment consists of the smart card platform, the card manager and the bytecode verifier.

- The **Java Card System Standard 2.1.1** configuration corresponds to a platform that includes all the functionalities described in version 2.1.1 of the Java Card platform specification ([JCVM21][JCRE21][JCAPI21]). It extends the **Minimal** configuration with the security requirements for loading applets onto the card, post-issuance. These applets have been previously verified off-card by a trusted IT component. No deletion of applets is considered in this configuration. This configuration comes from the use-case of a non-defensive open card.

- The **Java Card System Standard 2.2** configuration extends the previous configuration with all the features of version 2.2 of the Java Card specification. In particular, the deletion of both packages and applets are considered in this configuration. In this case, bytecode verification should also be performed off-card by a trusted IT component.

- The **Defensive** configuration also includes all the features of version 2.2 of the Java Card specification, together with an on-card bytecode verifier. This configuration comes from the defensive open card use-case.

For each configuration, the document contains the corresponding Protection Profile elements: the environment (assets, …, the security objectives), a selection of groups with their security features, and a rationale. These elements are clearly identified and linked so that no ambiguities remain for the reader or evaluator who is looking for the Protection Profile that corresponds to a specific configuration.

3.2 FROM FUNCTIONAL COMPONENTS TO GROUPS

The notion of ***group*** is close to what the CC calls a ***package***: a reusable set of Security Functional Requirements (SFR)s. However, in contrast to the CC package definition, a group is more influenced by the behavior of a given functional component of the Java Card platform than by the specific security objects that they are supposed to meet. Nevertheless, the SFRs included in a group are known to be useful in meeting an identifiable subset of security objectives. One can consider groups as a way to provide a modular definition of the security requirements associated to a set of functionalities of a Java Card platform. This modular presentation improves readability of the PPs and will reduce the cost of ST development by cutting down the workload on ST authors when they specify the IT security requirements.

The JCSPP Collection introduces, among others, groups of requirements concerning isolation of application data during execution, bytecode verification, installation and deletion of applications, and transmission of applications to the card.

The following list describes the groups of security requirements that are defined and used in the JCSPP Collection:

SCP	Contains the security requirements for the smart card platform; that is, the operating system and chip that the Java Card System is implemented upon. It does not define requirements for the TOE, but for its IT environment instead. The group is referred to as *SCPG.*
Core	Contains the basic requirements concerning the runtime environment of the Java Card System, such as the firewall policy and the requirements related to the Java Card API. The group is referred to as *CoreG*.

Bytecode verification	Contains the security requirements concerning the bytecode verification of the application code to be loaded on the card. This group of SFRs may apply to the TOE or to its IT environment depending on the configuration. The group is referred to as *BCVG.*
Installation	Contains the security requirements concerning the installation of post-issuance applications. It does not address card management issues in the broad sense, but only those security aspects of the installation procedure that are related to applet execution. The group is referred to as *InstG.*
Applet deletion	Contains the security requirements for erasing installed applets from the card, a new feature introduced in Java Card System 2.2. It can also be used as a basis for any other application deletion requirements. The group is referred to as *ADELG.*
Remote Method Invocation (RMI)	Contains the security requirements for the remote method invocation features, which provide a new protocol of communication between the terminal and the applets. This was introduced in version 2.2 of the Java Card System. The group is referred to as *RMIG.*
Logical channels	Contains the security requirements for the logical channels, which provide a runtime environment where several applets can be simultaneously selected or a single one can be selected more than once. This was introduced in version 2.2 of the Java Card platform. The group is referred to as *LCG.*
Object deletion	Contains the security requirements for the object deletion capability. This provides a safe memory recovering mechanism. This was introduced in version 2.2 of the Java Card System. The group is referred to as *ODELG.*
Secure carrier	Contains minimal requirements for secure downloading of applications on the card. It also contains the security requirements for preventing, in those configurations that do not support on-card static or dynamic verification of bytecodes, the installation of a *package* that has not been bytecode verified, or that has been modified after bytecode verification. The group is referred to as *CarG.*
Card manager	Contains the minimal requirements that allow defining a policy for controlling access to card content management operations and for expressing card issuer security concerns. The group is referred to as *CMGRG.*

Table 2. Groups of Security Functional Requirements

3.3 CONFIGURATIONS AND GROUPS OF SFRS

This section describes the four configurations and how they are related to the groups of security requirements.

The **Minimal** configuration corresponds to a multi-application card where no loading of applications at post-issuance time is allowed. The TOE is the simplest runtime environment for a Java Card system,

and its IT environment is the smart card platform and the card manager. Only the *SCPG*, *BCVG*, *CoreG* and *CMGRG* groups are included in this configuration.

The **Java Card System Standard 2.1.1** configuration corresponds to a platform that includes all the functionalities described in version 2.1.1 of the Java Card platform. It extends the **Minimal** configuration with the security requirements for loading applications onto the card post-issuance. It is assumed that these applications have been previously verified off-card by a remote, trusted IT component. The loader and the installer form part of the TOE, and, therefore, the groups *CarG* and *InstG* are included. This configuration, however, does not provide functionalities for the deletion of applets. Bytecode verification and card management applies to the TOE IT environment.

The **Java Card System Standard 2.2** configuration extends the **Java Card System Standard 2.1.1** configuration with all the features introduced in version 2.2 of the Java Card platform (RMI, logical channels, applet deletion, and object deletion). Therefore, the groups *RMIG*, *LCG*, *ADELG* and *ODELG are* included. Bytecode verification and card management applies to the TOE IT environment.

The **Defensive** configuration, like the **Java Card System Standard 2.2** configuration, also includes all the features considered in version 2.2 of the Java Card platform. In addition, bytecode verification is performed on-card and the bytecode verifier is then a component of the TOE. The *BCVG* group is therefore also included as part of the TOE. This is not the case for the group *CarG* because installation of malicious applets is prevented independently from the origin of the application and the way it has been downloaded onto the card. Card management applies to the TOE IT environment.

Figure 2 illustrates the relationship between the configurations and the groups of Security Functional Requirements.

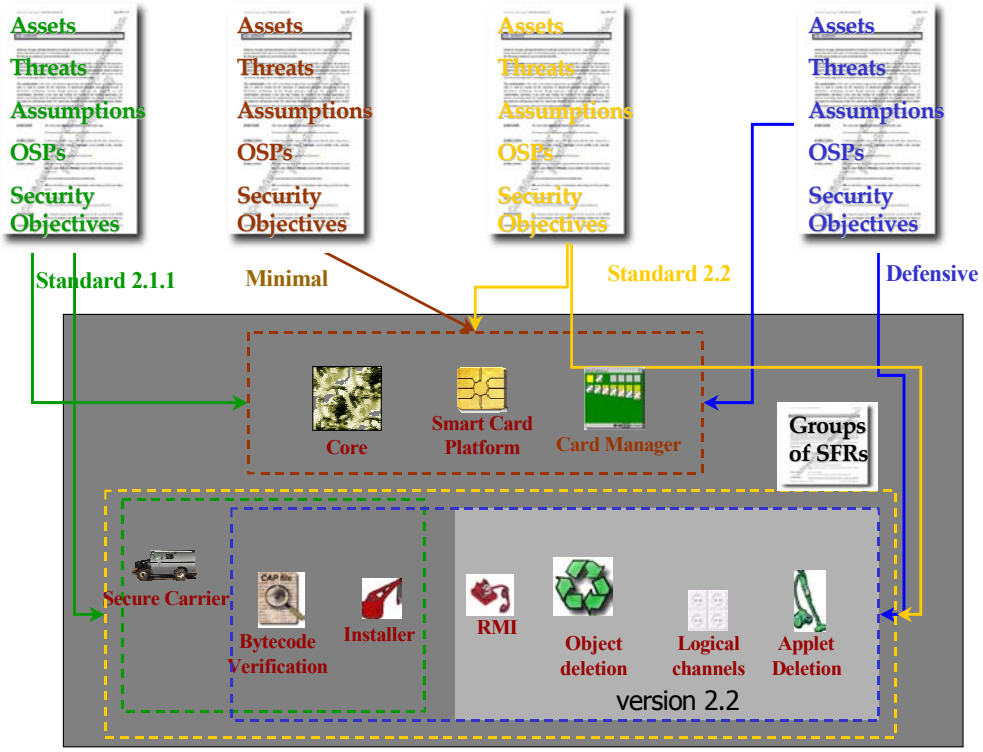

Figure 2. Configurations and groups of SFRs

4 HOW TO READ THE JCSPP COLLECTION

The structure of the JCSPP Collection conforms to the general specification of Protection Profiles defined in the CC specification ([9], appendix B). However, the use of configurations and groups defines a sub-structure of the document that deserves some additional explanation. This section reviews the contents of the document and explains its structure.

The **TOE description** defines the **TOE limits**. As the TOE depends on the configuration, descriptions have been included that clarify which TOE corresponds to each configuration. The groups that are included in each configuration are also described.

4.1 SECURITY ASPECTS AND SECURITY ENVIRONMENT

The TOE security environment is organized by configuration. For instance, in a closed card, post-issuance installation of applets is not possible, either because the TOE has no installation capability or because it has been disabled prior to the issuance of the card. Therefore, the Minimal configuration will not contain any SFR related to the **Installation** group. Conversely, all configurations share a number of common items with respect to the security environment (such as the IC, the development and production of the card, or runtime threats).

To avoid unnecessary duplication of text between the configurations, a notion of *security aspect* was introduced. Security aspects are intended to define the main security issues that are to be addressed in the PPs, in a CC-independent way. In addition, they provide 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 IT environment), and even as organizational security policies.

As an example of security aspect, we take application code verification. This is an important issue in every configuration concerning the runtime security (and safety) of the TOE. A detailed description of bytecode verification, and its relevance and relationship to the runtime of bytecode, is provided in the *#.VERIFICATION* security aspect.

The security environment defined for each of the configurations may contain different instances of this security aspect:

- In the Minimal configuration, the loading and installation of applications occurs in a pre-issuance phase. This use case assumes that these pre-installed applications are securely managed, implemented, and verified. The *#.VERIFICATION* aspect is therefore instantiated as an assumption, termed *A.VERIFICATION*. As the TOE does not contain a bytecode verifier, the verification is located in the environment. The description of the security environment, therefore, contains an environmental objective, *OE.VERIFICATION*, which meets the assumption *A.VERIFICATION*).

- In the Java Card System Standard 2.1.1 and Java Card System Standard 2.2 configurations, *applet*s can be loaded and installed post-issuance. As in the case of the Minimal configuration, the verification occurs in the environment. The description of the security environment therefore contains an objective on the environment, *OE.VERIFICATION* (it will meet the assumption *A.VERIFICATION*). On the other hand, the card and application management also needs organizational measures to be effective. In particular, it must be ensured that the application code that is being installed on the card is the one that has been verified. The

objective *OE.VERIFICATION* also supports an organizational security policy dedicated to this latter security issue. This policy is termed *OSP.VERIFICATION*.

- Finally, in the Defensive configuration, the bytecode verifier is an embedded component included in the TOE. Since bytecode verification is addressed by the SFRs of the TOE, the security aspect is instantiated as a mere objective for the TOE, termed *O.VERIFICATION*.

#.VERIFICATION		
Minimal	Java Card System Standard 2.1.1 & 2.2	Defensive
A.VERIFICATION	*A.VERIFICATION*	*O.VERIFICATION*
OE.VERIFICATION	*OSP.VERIFICATION*	
	OE.VERIFICATION	

Table 3. Security aspect instances

Note: there is no threat termed *T.VERIFICATION* instantiated from the *#.VERIFICATION* aspect. Actually, bytecode verification contributes to the TOE security by helping to address a broad spectrum of threats. There is no verification-specific threat. Other security aspects have "threats instances".

The assets to be protected, the users of the TOE, and their software counterparts are then determined. Assets are common to all the configurations.

The different security components are then determined, organized by configuration. First, assumptions are made on the environment of the TOE for each of the configurations.

The threats to the assets against which specific protection is required within the configurations or their environments are then determined. For example, the post-issuance installation of applets introduces one threat (*T.INSTALL*), and two more (T.INTEG-APPLI-CODE.2 and T.INTEG-APPLI-DATA.2*)* in the case that bytecode verification is performed off-card*.* The latter two threats are absent from the Defensive configuration.

Finally, there are the organizational policies that shall be imposed, when appropriate, on the environment of the TOE. Only the configurations Java Card System Standard 2.1.1 and Standard 2.2 require such a policy.

4.2 SECURITY OBJECTIVES

The **security objectives** to be achieved by each of the TOE configurations and their respective environments are derived from the security environment. The presentation of the objectives is organized by classifying them either as objectives of the TOE or as objectives of the environment of the TOE. Within these two categories, they are presented on a "per configuration" basis.

The **TOE security objectives** for the Minimal configuration are also objectives for all of the other configurations. This is a consequence of the fact that all threats for the Minimal configuration are also threats for the rest of the configurations.

The Java Card System Standard 2.1.1 configuration introduces the objectives that are intended to counter the threats of a use-case where post-issuance installation of applets is provided and bytecode verification is performed off card. This requires that the authenticity and integrity of the (already verified) file loaded into the card can be ensured.

The Java Card System Standard 2.2 configuration introduces these objectives that are intended to counter the threats that can be posed to a platform supporting logical channels, applet deletion, remote method invocation, and object deletion.

Finally the Defensive configuration introduces, as a security objective, the bytecode verification of loaded files. This makes sense, considering that in this configuration the bytecode verifier is a component of the TOE.

There are **security objectives for the IT environment** that are common to all configurations. They are concerned with the security aspects to be enforced by the smart card integrated circuit and operating system.

There are objectives that apply only for the Minimal configuration, concerning the capability of installing and deleting packages and/or applets after the card has been issued.

There are three more environmental objectives. One of them is appropriate for configurations where bytecode verification is performed off-card. This requires that every bytecode must be verified prior its execution on the card. Another objective applies to open configurations and is related to the installation at post-issuance of native code. Finally, there is an objective concerning the minimal requirements to be met by the card manager to ensure the correct behavior of the security functions of the TOE.

4.3 SECURITY REQUIREMENTS

The section ***IT Security Requirements*** of the JCSPP Collection provides a detailed definition of the security requirements that must be satisfied by each configuration of the TOE and its respective environment. All the SFRs that are used have been drawn from [CC2]. Therefore, the four PPs are, in CC terminology, *Part 2 conformant*.

If the security environment section of the JCSPP Collection is sub-divided into configuration parts, the security requirements are, in contrast, presented in a group-by-group basis. As described in Table 2, the SFRs are arranged into several groups that are then composed to form the different configurations of the TOE.

The section ***TOE Security Assurance Requirements*** states the assurance requirements for the evaluation of a compliant ST. The Evaluation Assurance Level ("EAL") required is EAL4 augmented. The four PPs defined in JCSPP Collection share this requirement. Augmentation results from the selection of the Security Assurance Requirements ("SAR")s ***AVA_VLA.3*** and ***ADV_IMP.2.*** All the SARs that are used have been drawn from [CC3]. Therefore, the four PPs are, in CC terminology, *Part 3 conformant*.

4.4 RATIONALE

The rationale is the backbone of a PP or a ST. The JCSPP Collection includes a security objectives rationale and a security requirements rationale. Being a crucial input for evaluation and certification, each of these rationales has been structured so that the four configurations are justified separately.

The rationale for the **security objectives** is organized as follows:

- All the *security objectives fixed for the TOE* contribute to counter some threat on the assets. To provide evidence that all threats are actually prevented by some combination of security objectives, the presentation is oriented by the threats. The security objectives to be achieved by the TOE environment are also related to the threats on the assets of the TOE. Sometimes, the relation may be indirectly stated through the security objectives of the TOE that the environmental objective supports, in the sense that this latter one is intended to counter a collection of threats.

- The security objectives to be achieved by each configuration are related to the assumptions on the TOE and its environment

- The relationship between security objectives and organizational security policies is stated.

The rationale for the **security requirements** consists of justifications that apply to the security **functional** requirements and to the security **assurance** requirements that have been selected.

The rationale for the functional requirements included in the configuration:

- Demonstrates that the set of requirements (both on the TOE and on the environment) is suitable to meet the corresponding security objectives. The presentation is analogous to that of the security objectives. It lists and justifies the requirements that are related to each objective of the configuration.

- Shows that all the **dependencies** between the security functional requirement components are satisfied.

- Justifies the adequacy of the strength of function level (**SOF-medium)** that has been chosen.

The rationale for the chosen **EAL** is described below. As already mentioned, the assurance requirement of each PP is **EAL4 augmented**.

EAL4 has been chosen because it allows a developer to attain a reasonably high assurance level without the need for highly specialized processes and practices. It corresponds to a white box analysis and it can be considered as a reasonable level that can be applied to an existing product line without undue expense and complexity.

Augmentation results from the selection of:

- ***AVA_VLA.3*** Vulnerability Assessment - Vulnerability Analysis - Moderately resistant, and
- ***ADV_IMP.2*** Development – Implementation Representation – Implementation of the TSF.

EAL4 requires vulnerability assessment through imposition of ***AVA_VLA.2***. This dictates a review of identified vulnerabilities only. The component ***AVA_VLA.3*** requires that a systematic search for vulnerabilities be documented and presented. This provides a significant increase in the consideration of vulnerabilities over that provided by ***AVA_VLA.2***.

Through imposition of ***ADV_IMP.1***, EAL4 requires the description of a subset of the implementation representation to capture the detailed internal working of the TSF. The component ***ADV_IMP.2*** requires the developer to provide the implementation representation for the entire TSF.

4.5 A UNIFIED VIEW OF CONFIGURATIONS

A comprehensive presentation of each configuration is included as part of the identification section of the PPs defined in the JCSPP Collection document. This presentation assembles the references to the document's sections where the configuration's security environment, security objectives, security requirements, and rationales are presented.

In addition, there is also a section that provides an all-embracing presentation of the security components of the four PPs. The tables in this section describe the contents of each configuration and the differences between them.

5 HOW TO USE THE JCSPP COLLECTION TO WRITE A SECURITY TARGET

The JCSPP Collection can be used in at least two different ways:

- A security target may claim conformance to one of the PPs that it contains. As already described, each configuration determines a PP in the classical sense, which precisely describes the limits, environment, and security requirements of a product implementing the chosen configuration.

- The JCSPP Collection can also be used as a toolbox for writing an ST for a Java Card platform. An ST developer can do this either by introducing a new configuration, or by including the general security issues and the groups of requirements in the JCSPP Collection that are common to all possible configurations.

5.1 CLAIMING PP CONFORMANCE

It is worth noting that the collection of PPs defined in the JCSPP Collection is subject to a Common Criteria evaluation. Therefore, an ST cannot claim conformance directly to the JCSPP Collection, but rather to one of the PPs.

The CC scheme can only claim full-conformance: one cannot claim to be partially conformant to a PP. As a consequence, all the features included in the configuration chosen by the ST author shall be implemented in the targeted product. To identify a suitable candidate configuration, the ST author shall examine the choices made for each of the three security parameters defined in Section 2. If the TOE is a closed platform, then **Minimal** is the only JCSPP Collection candidate configuration. If the TOE is an open platform, then the version of Java Card technology implemented by the platform shall be considered. If the platform implements version 2.1.1 of Java Card technology, then **Java Card System Standard 2.1.1** is the only JCSPP Collection candidate configuration. Finally, if the TOE is an open Java Card platform, version 2.2, the ST author shall analyze where the bytecode verification step is placed in the platform's security architecture. If verification is performed before downloading, then **Java Card System Standard 2.2** is the candidate configuration. Otherwise, the candidate configuration is **Defensive**.

If the chosen configuration actually corresponds to the TOE, the ST author shall refer to Section 7 of the JCSPP Collection ("*A Unified View of Configurations*"). This section lists all the components of the IT environment and the SFRs included in each of the PPs defined in the JCSPP Collection. The ST author may then proceed as usual, applying the refining and selection operations specified in the CC on each of the SFRs associated to the PP, and possibly extending a group included in the configuration with new, TOE-specific SFRs.

To be compliant with any of the PPs defined in the JCSPP Collection, a Java Card technology-enabled product ST is also required to provide a precise identification of all the components on which it may rely (the underlying smart card platform and the bytecode verifier) or with which it may interact (card manager and native code). The product developer shall also provide evidence that the assumptions and the security requirements defined on those components are consistently enforced. This is necessary to guarantee that the TOE security architecture can rely on them. Native code is slightly different as it is not in the scope of the TOE addressed by the Protection Profiles. In the (very typical) case that the product embodies native applications, they must also be uniquely identified and

evidence must be provided that they do not violate security policies stated for the TOE. Should native software be privileged in this respect, exceptions to the policies must include a rationale for the new security framework they introduce.

5.2 INTRODUCING NEW CONFIGURATIONS

If the TOE does not correspond to any of the configurations defined in the JCSPP Collection, then the ST author may still use the JCSPP Collection as a basis to write an ST. The most likely situation for such a TOE is the result of an instance of the risk analysis parameters presented in Section 2 that is different from the four considered in the JCSPP Collection. An example of such a TOE is an open Java Card platform version 2.1.1 including an embedded bytecode verifier.

If this is the case, the ST author may assemble a new configuration corresponding to the TOE. To assemble the new configuration, the ST author shall refer to the general security issues and the groups of requirements introduced in the JCSPP Collection.

In every configuration, security aspects (§3.1) and their instances provide valuable insights on the security issues concerning both the runtime environment and applet management. These security issues can be used to assemble the TOE security environment and security objectives.

Then, the functional specification of the product must be examined to select those groups of SFRs that are suitable for the configuration. The following guidelines can be applied in the selection of groups:

- The Core and Smart Card Platform groups (*CoreG* and *SCPG*, respectively) must always be included. The Core group must be included as a group of requirements imposed on the TOE. The SCP group may be imposed either on the TOE or on the IT environment, depending on whether the product is restricted to the runtime environment of the Java Card platform or to the entire smart card.

- The Card manager group (*CMGRG*) must also always be included. This group embodies the SFRs necessary for defining a policy for controlling the access to card content management operations. The SFRs then must be instantiated in accordance with the card management component of the product. Analogous to the two groups above, this group of security requirements may be either imposed on the TOE or on the IT environment.

- The installation group (*InstG*) should be included if the TOE is an open platform containing an application installer compliant to the Java Card specification (whatever the version). This group of requirements must be imposed on the TOE.

- The bytecode verification group (*BCVG*) must be included if no assumption or organizational policy answers the issues stated in the security issue named *#.VERIFICATION*. This group may be imposed either on the TOE or on the IT environment. If bytecode verification is performed on the platform, then the group must be imposed on the TOE. If verification is performed before applet downloading, this group should support environmental objectives ensured by other IT products, like an off-card verifier.

- The secure carrier group (*CarG*) provides requirements for secure transmission of previously verified applications. It shall be included when bytecode verification is performed before applet downloading. This group shall be included as part of the TOE.

- The applet deletion group (*ADELG*) shall be included if the TOE contains an applet deletion manager compliant to the Java Card specification, version 2.2. According to the specification,

if the platform includes a post-issuance installer, then an applet deletion manager shall also be included. Therefore, if the platform implements version 2.2 of the Java Card specification, this group shall also be included if the installation group of requirements is included.

- The logical channels and RMI groups (*LCG* and *RMIG*, respectively) shall be included if the platform supports version 2.2 of the Java Card specification. These groups shall be imposed on the TOE.

- The object deletion group (*ODELG*) shall be included if the TOE is a Java Card platform version 2.2, that supports automated deallocation of Java objects. Object deletion is an optional feature only available for version 2.2 of the Java Card platform. This group shall be imposed on the TOE.

Once the new configuration has been defined, the ST author may perform the refinement, selection, and extension operations necessary to transform the JCSPP Collection configuration into a ST.

The previous descriptions imply that one cannot choose any combination of values for the three key parameters of the risk analysis introduced in Section2. Table 4 illustrates the dependencies that relate these parameters.

Core group	Required by all other groups.
Smart card platform group	Required by all other groups.
Card manager group	No dependency.
Installation group	No dependency.
Remote Method Invocation group	Required by the Logical channels group.
Logical channels group	Required by the Remote Method Invocation group.
Object deletion group	No dependency.
Bytecode verification group	No dependency.
Applet deletion group	Required by the Installation group.
Secure carrier group	Required by the Bytecode verification group if verification is performed before downloading.

Table 4. Summary of group dependencies

6 REFERENCES

[CC1] *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model*. Version 2.1. August 1999. CCIMB-99-031.

[CC2] *Common Criteria for Information Technology Security Evaluation, Part 2: Security functional requirements*. Version 2.1. August 1999. CCIMB-99-032.

[CC3] *Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance requirements*. Version 2.1. August 1999. CCIMB-99-033.

[GPCS] *GlobalPlatform Card Specification,* Version 2.1.1, March 2003.

[GPCSRS] *GlobalPlatform Card Security Requirements Specification*, Version 1.0, May 2003.

[JCVM21] *Java Card 2.1.1 Virtual Machine (JCVM) Specification*. Revision 1.0. May 18, 2000. Published by Sun Microsystems, Inc.

[JCAPI21] *Java Card 2.1.1 Application Programming Interface*. Revision 1.0. May 18, 2000. Published by Sun Microsystems, Inc.

[JCRE21] *Java Card 2.1.1 Runtime Environment (JCRE) Specification*. Revision 1.0. May 18, 2000. Published by Sun Microsystems, Inc.

[JCVM22] *Java Card 2.2 Virtual Machine (JCVM) Specification*. June 2002. Published by Sun Microsystems, Inc.

[JCAPI22] *Java Card 2.2 Application Programming Interface*. June 2002. Published by Sun Microsystems, Inc.

[JCRE22] *Java Card 2.2 Runtime Environment (JCRE) Specification*. June 2002. Published by Sun Microsystems, Inc.

[JCBV] *Java Card 2.1.2 Off-Card Verifier*. January 2001. White paper. Published by Sun Microsystems, Inc.

[JCSPP] *Java Card System Protection Profile Collection*. June 2003. Sun Microsystems, Inc.

[SCSUG] *Smart Card Protection Profile*. Smart Card Security User Group. Version 3.0, September 9, 2001. Registered and Certified by Bundesamt für Sicherheit in der Informationstechnik (BSI) under the reference BSI-PP-000 3-2001. Registered and Certified by the French Certification Body under the reference PP/0103. Registered and Certified by the Canadian Certification Body.

[ESPP] *Protection Profile Smart Card IC with Multi-Application Secure Platform*. Version

2.0, Issue November 2000. Registered and Certified by the French Certification Body under the reference PP/0010.

[SSVG] *Smartcard IC Platform Protection Profile*. Version 1.0, July 2001. Registered and Certified by Bundesamt für Sicherheit in der Informationstechnik (BSI) under the reference BSI-PP-0002.

7 APPENDIX: JCSPP COLLECTION, GLOBALPLATFORM AND COMPATIBLE PROTECTION PROFILES

The TOE addressed by each of the four PPs included in the JCSPP Collection is typically only one of the components of a deployed Java Card platform. The overall security of the product also depends on the contribution provided by other constituents, such as the underlying platform and the implemented card manager architecture. The clarification of these dependencies, which are implicit in the Java Card specification, is the key to achieving a **safe and coherent interaction** of the components. That is, the goal is to build security interoperability on top of functional interoperability. The SFRs, SCP, and Card manager included in the JCSPP Collection have been designed with that objective.

On the one hand, these components define security requirements for the operating system and chip upon which the Java Card System is implemented. They also define a policy for controlling access to card content management operations and for expressing card issuer security concerns, respectively. Therefore, they provide a precise characterization of the security measures that the components can enforce to guarantee the correct operation of the security mechanisms of the TOE. Accordingly, in both cases the SFRs apply for the IT environment of the TOE.

On the other hand, these groups of security requirements have been defined to be consistent with those included in already existing (and certified) Protection Profiles for the underlying platform. Examples of such profiles are SCSUG's *Smart Card Protection Profile* [SCSUG] and Eurosmart's *Smart Card IC with Multi-Application Secure Platform*" [ESPP]. They are also consistent with the security policies and features defined in Global Platform's *Card Security Requirements Specification* [GPCSRS].

The GlobalPlatform Card Specification[GPCS] provides a detailed account of the security requirements for GlobalPlatform (GP). No detail, however, is defined concerning the security requirements for the runtime environment (RTE), the underlying operating system (OS), and the integrated circuitry (IC) on top of which the GlobalPlatform is implemented.

In contrast, the document [GPCSRS] presents a unified specification of the security requirements of all the above-mentioned components. In that document, the security requirements are assembled into conceptual unities called **security features**. These security features are, in turn, presented as forming groups that can be interpreted as expressing security requirements of smart card components. The following is the interpretation used throughout that document:

- Group 8 is associated with Applications
- Groups 5, 6 and 7 are associated with GP components
- Groups 3 and 4 are associated with the RTE and GP components
- Group 2 is associated with the RTE, OS and the IC
- Group 1 is associated with the IC.

Section 7.1 presents a table that relates SFRs of the JCSPP Collection with security features or policies included in some of the groups listed above. A preliminary version of this table has been presented in Annex A.3 of [GPCSRS]. The table included in this document is an extension of the version that incorporates the SFRs of the Card manager group. This group of security requirements has been

added to the JCSPP Collection to provide a precise characterization of the security measures that are expected to be enforced in the IT environment of the TOE.

In addition, the security features specified in [GPCSRS] are compatible with two additional (complementary) PPs, namely the SCSUG smart card protection profile (mentioned above) and the Secure Silicon Vendors Group (SSVG) protection profile [SSVG]. A working group within GlobalPlatform is currently defining the guidelines for the development of (composite) Security Targets of smart card products. These guidelines are intended for manufacturers that seek a Common Criteria evaluation of their products. A preliminary outcome of that working group is the table presented in section 7.2 below. This table, originally presented in Annex A.1 of [GPCSRS], depicts the relation between security functional requirements of SCSUG, SSVG and JCSPP Collection, and the GP security features. In addition, the table shows how the security requirements can be used to meet threats and security policies that have been specified in [GPCSRS].

7.1 JCSPP COLLECTION SFRS AND GP SECURITY FEATURES

As already mentioned, Table 5 below relates the SFRs of the JCSPP Collection to the security features specified in [GPCSRS]. The table should be read as follows: the ***SFR*** defined in the ***Group*** of the JCSPP Collection is related to the ***Tables*** of the ***Security Policies or Features*** defined in the ***Group*** of the Open Platform GPCSRS. The last column (***SFR Instantiation***) provides a brief description of how the SFR has been instantiated in the JCSPP Collection.

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FAU_ARP.1/JCS	*Core*	**3**	**Event Actions**	**5-40**	Throw an exception, lock the session or reinitialize the system when a Java Card platform `SecurityException` is raised. A potential security violation is refined to the accumulation or combination of auditable events like, among others, Integrity loss on a key or PIN object, applet life cycle inconsistency, package inconsistency, card tearing (unexpected removal of the card out of the CAD) and power failure, Illegal access to an object or typing error.

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FCS_CKM.1	*Core*	**7**	**RTE API Access**	**5-34**	Requirements on session key generation imposed by the classes `KeyBuilder` and `KeyPair` of the Java Card API.
FCS_CKM.2	*Core*	**7**	**RTE API Access**	**5-34**	Requirements imposed by method `SetKEY` of the Java Card API.
FCS_CKM.3	*Core*	**7**	**RTE API Access**	**5-34**	Access to the keys in accordance with class `Key` of the Java Card API.
FCS_CKM.4	*Core*	**7**	**RTE API Access**	**5-34**	The keys are reset in accordance with the method `clearKey()` in the class `Key` of the Java Card API.
FCS_COP.1	*Core*	**7**	**RTE API Access**	**5-50**	A subset of cryptographic operations in accordance with the Java Card API (`javacardx.crypto.Cipher` and `javacardx.security` packages).
FDP_ACC.2/FIREWALL	*Core*	**3**	**Firewall**	**5-38**	This access control policy controls the access of the Applets to the Java objects, including Shareable objects.
FDP_ACF.1/FIREWALL	*Core*	**3**	**Firewall**	**5-38**	
FMT_MSA.1/JCRE	*Core*	**3**	**Firewall**	**5-38**	
FMT_MSA.2/JCRE	*Core*	**3**	**Firewall**	**5-38**	
FMT_MSA.3/FIREWALL	*Core*	**3**	**Firewall**	**5-38**	
FMT_MTD.1/JCRE	*Core*	**3**	**Firewall**	**5-38**	
FPT_SEP.1	*Core*	**3**	**Firewall**	**5-38**	Separations between different execution contexts are required.
FDP_IFC.1/JCVM	*Core*	**3**	**Firewall**	**5-38**	Controls that the currently selected Applet does not keep a pointer to the APDU buffer beyond the current session.
FDP_IFF.1/JCVM	*Core*	**3**	**Firewall**	**5-38**	
FDP_RIP.1	*Core*	**3**	**Object Reuse**	**5-39**	APDU buffer, bArray of install, Transients, Cryptographic buffer.
FDP_SDI.2	*Core*	**2**	**Memory Content**	**5-47**	Monitoring of crypto keys, PINs and code.

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FIA_ATD.1/AID	*Core*	**3**	**Subject/Object Identification**	**5-41**	Maintenance of security attributes belonging to individual users: a package has a package AID and a list of would-be installed applets AIDs.
FIA_UID.2/AID	*Core*	**3**	**Subject/Object Identification**	**5-41**	Every action is always performed by an identified user interpreted as the *installed* applet or the package that owns it.
FIA_USB.1	*Core*	**3**	**Subject/Object Identification**	**5-41**	The security functions shall associate the appropriate user security attributes with subjects acting on behalf of that user.
FMT_SMR.1/ROLES	*Core*	**3**	**Supervisor**	**5-35**	The supervisor role mentioned in the security feature shall be associated to the JCRE introduced by the component.
FPR_UNO.1	*Core*	**7**	**OP API access, RTE API access**	**5-33, 5-34**	The unobservability can be applied, depending on implementations, to key and PIN values, and to the result of cryptographic operations and comparisons performed on those values.
FPT_FLS.1/JCS	*Core*	**4**	**Failure Management**	**5-45**	A secure state is preserved when failures associated to auditable events listed in FAU_SAA.1.2/JCS component occur.
FPT_RVM.1	*Core*	**3**	**Supervisor**	**5-35**	Non-bypassability of the card security functions.
FPT_TST.1	*Core*	**4**	**Self Test**	**5-44**	A suite of self-tests during initial start-up is run.
FDP_ITC.2/Installer	*Inst*	**5**	**Card Content Management**	**5-4, 5-5**	Rules are enforced when importing application packages and installing new applets.

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FMT_SMR.1/ROLES	*Inst*	**5**	**Section 3.1, P-Roles**	**n/a**	The Installer, which acts on behalf of the card issuer, is involved in the loading of packages and installation of applets.
FPT_FLS.1/CM	*Inst*	**4**	**Failure Management**	**5-45**	Preservation of secure state when the Installer fails to load/install a package/applet.
FPT_RCV.3/CM	*Inst*	**4**	**Failure Management**	**5-45**	When automated recovery from a failure or service discontinuity is not possible, the TSF shall enter a maintenance mode where the ability to return the TOE to a secure state is provided.
FRU_RSA.1/Installer	*Inst*	**5**	**Card Content Management**	**5-4, 5-5**	The controlled resources are the number of packages, classes, fields and methods that can be loaded on the card.
FDP_IFC.2/BCV	*BCV*	**6**	**P.ACV, Mandated DAP verification**	**5-30**	Information flow control policy describing the constraints imposed on the operations (the bytecodes) that make information (integers, references, etc) to flow between the storage positions of the JCVM.
FDP_IFF.2/BCV	*BCV*				
FMT_MSA.1/BCV.1	*BCV*				
FMT_MSA.1/BCV.2	*BCV*				
FMT_MSA.2/JCRE	*BCV*				
FMT_MSA.3/BCV	*BCV*				
FMT_SMR.1/ROLES	*BCV*	**6**	**Section 3.1, P.Roles**	**n/a**	The role introduced in the SFR corresponds to the Verification Authority in the GPCSRS.
FRU_RSA.1/BCV	*BCV*	**6**	**P.ACV, Mandated DAP verification**	**5-30**	The resources under control are the size of the operand stack and the number of local variables of each method.
FDP_ACC.2/ADEL	*ADEL*		**Card Content**	**5-8,**	

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FDP_ACF.1/ADEL	ADEL	**5,6**	**Management, Token Verification, Security Domain Access**	**5-9, 5-21**	The access policy requires the applet deletion manager associated to the deleted applet to be selected, and prevents deletion when the applet or package is referred from other applets or packages. In GP, the applet deletion manager may be an application with delegated management.
FPT_FLS.1/ADEL	ADEL	**4**	**Failure Management**	**5-45**	The operation that shall preserve a secure state is applet and package deletion.
FDP_RIP.2/OBJ-DEL	ODEL	**3**	**Object Reuse**	**5-39**	Freed data resources resulting from the invocation of the method `javacard.framework.JCSystem.requestObjectDeletion()` may be reused. Requirements on *de-allocation* after the invocation of the method are described in version 2.2 of the Java Card specification.
FPT_FLS.1/OBJ-DEL	ODEL	**4**	**Failure Management**	**5-45**	The preservation of a secure state is required when the object deletion functions fail to delete all the unreferenced objects owned by the applet that requested the execution of the method.

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FCO_NRO.2/CM	*Car*	**6**	**P.LFV , Mandated DAP Verification**	**5-30**	The type of information where origin is controlled are the application packages loaded on the card. The *Car* group contains the security requirements for preventing, in those configurations, which do not support on-card static or dynamic verification of bytecodes, the installation of a package that has not been bytecode verified, or that has been modified after bytecode verification.
FTP_ITC.1/CM	*Car*	**6**	**Secure Channel**	**5-22, 5-23, 5-24, 5-25, 5-26, 5-27**	A secure channel is required for the communication between the CAD and the card.
FDP_IFC.2/CM	*Car*	**6**	**Secure Channel Initiation**	**5-22, 5-23, 5-24**	The information flow control policy describes the exchanges of messages that are authorized by the communication protocol implementing the secure channel.
FDP_IFF.1/CM	*Car*				
FMT_MSA.1/CM	*Car*				
FMT_MSA.3/CM	*Car*				
FDP_UIT.1/CM	*Car*	**6**	**Message Integrity and Authentication**	**5-24**	The modification, deletion, insertion and replay of messages shall be prevented during package loading.
FIA_UID.1/CM	*Car*	**5,6**	**Issuer Security Domain Access, Other functions**	**5-1, 5-32**	Allowed actions on behalf of the user to be performed before the user is identified (like the GETDATA command) are left open in the JCSPP Collection.
FPT_PHP.3/SCP	*SCP*	**1**	**Tamper Resistance**	**5-51**	Not instantiated
FPT_AMT.1/SCP	*SCP*	**4**	**Self Test**	**5-44**	Tests shall be run at each power on.
FPT_FLS.1/SCP	*SCP*	**4**	**Failure Management**	**5-45**	Not instantiated.
FPT_RCV.3/SCP	*SCP*	**4**	**Failure Management**	**5-45**	Not instantiated

JCSPP Collection		GPCSRS			SFR Instantiation
SFR	**Group**	**Group**	**Security Policy or Feature**	**Table**	
FPT_RCV.4/SCP	*SCP*	**4**	**Failure Management**	**5-45**	The operations that shall be protected are non-atomic memory accesses, like writing a Java field.
FPT_RVM.1/SCP	*SCP*	**4**	**Failure Management**	**5-45**	Not instantiated.
FPT_SEP.1/SCP	*SCP*	**2**	**Memory Content**	**5-47**	Not instantiated.
FRU_FLT.1/SCP	*SCP*	**4**	**Failure Management**	**5-45**	Not instantiated.
FDP_ACC.1/CMGR	*CMGR*	**5**	**Card Content Management**	**5-4, 5-5, 5-6, 5-7, 5-8**	Not instantiated.
FDP_ACF.1/CMGR	*CMGR*	**5**	**Card Content Management**		
FMT_MSA.1/CMGR	*CMGR*	**5**	**Card Content Management**		
FMT_MSA.3/CMGR	*CMGR*	**5**	**Card Content Management**		
FIA_UID.1/CMGR	*CMGR*	**5**	**Card Content Management**		

Table 5. JCSPP Collection SFRs and GP security groups

7.2 GP SECURITY AND COMPATIBLE PROTECTION PROFILES

Table 6 shows how the Security Features relate to the Security Function Requirements (SFRs) defined in the applicable Protection Profiles. This table was originally presented in Annex A.1 of [GPCSRS]. It depicts the relation between security functional requirements of SCSUG, SSVG and JCSPP Collection, and the GP security features. In addition, the table shows how the security requirements can be used to meet threats and security policies that have been specified in [GPCSRS].

Unmarked SFRs are from SCSUG, those marked with one star * are from SSVG, and those marked with a double star (**) are from the JCSPP Collection. Threats that are underlined are threats that the Security Features protect against but are not identified in the SCSUG and SSVG Protection Profiles. Similarly, threats marked with a double star (**) are threats that the extra SFRs from JCSPP Collection protect against, but which are not identified in the SCSUG and SSVG.

Security Feature and SFRs	Applicable Threats and Policies
Group 7	
OP API Access Security Feature	
FDP_ACC.1, FDP_ACF.1, FMT_MOF.1, FMT_MSA.2, FMT_MTD.1, FMT_MTD.3 FPR_UNO.1**	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.BRUTE-FORCE, T.FIRST_USE, T.FLT_INS, T.INV_INP, T.LINK, T.LNK_ATT, T.UA_LOAD, T.I_LINK**
RTE API Access Security Feature	
FMT_MOF.1, FMT_MSA.2, FMT_MTD.3 (FPR_UNO.1, FCS_CKM.1,FCS_CKM.2, FCS_CKM.3,FCS_CKM.4, FCS_COP.1)**	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.BRUTE-FORCE, T.INV_INP, T.LNK_ATT, T.UA_LOAD, T.I_LINK**, T.CRYPT_ATK**
Group 6	
Security Domain Access Security Feature	
FDP_ACC.1, FDP_ACF.1, FMT_MOF.1, FMT_MSA.2, FMT_MTD.1, FMT_MTD.3 FDP_ACC.2**	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.BRUTE-FORCE, T.FIRST_USE, T.FLT_INS, T.INV_INP, T.LINK, T.LNK_ATT, T.UA_LOAD
Secure Channel Security Feature	
Secure Channel Initiation	

Security Feature and SFRs	Applicable Threats and Policies
FTP_ITC.1, FCS_COP.1, FIA_UAU.1, FIA_UAU.7, FIA_UID.1 (FDP_IFC.2, FDP_IFF.1, FMT_MSA.1, FMT_MSA.3)**	P.CRYPT_STD, P.DATA_ACC, P.FILE_STR, P.SEC_COM, T.ACCESS, T.BRUTE-FORCE, **T.CLON**, T.CRYPT_ATK, T.FIRST_USE, T.INV_INP, T.LINK, T.LNK_ATT, T.REUSE, T.UA_LOAD
Message Data Confidentiality	
FCS_COP.1	
Message Integrity and Authentication	
FCS_COP.1, FDP_UIT.1, FPT_ITI.1, FPT_RPL.1	
Key and other Secret Data Receiving Services	
FCS_COP.1	
Non-Secure Channel Security Feature	
FDP_ETC.1, FDP_ITC.1	P.DATA_ACC, P.SEC_COM, T.ACCESS, T.FLT_INS, T.LINK, T.LNK_ATT, **T.UA_LOAD**
Mandated DAP Verification Security Feature	
FCS_COP.1 (FDP_IFC.2, FDP_IFF.2, FDP_MSA.1, FDP_MSA.2, FMT_MSA.3, FRU_RSA.1, FCO_NRO.2)**	P.CRYPT_STD, T.CRYPT_ATK, **T.UA_LOAD**
DAP Verification Security Feature	
FCS_COP.1	P.CRYPT_STD, T.CRYPT_ATK, **T.UA_LOAD**

Security Feature and SFRs		Applicable Threats and Policies
CVM Handling Security Feature		
	FIA_UAU.7, FIA_AFL.1, FMT_MTD.2	T.BRUTE-FORCE, T.INV_INP, T.LNK_ATT, T.REUSE
Other functions Security Feature		
	FIA_UID**	T.ACCESS**
Group 5		
Issuer Security Domain Access Security Feature		
	FDP_ACC.1, FDP_ACF.1, FMT_MOF.1, FMT_MSA.2, FMT_MTD.1, FMT_MTD.3 FIA_UID.1**	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.BRUTE-FORCE, T.FIRST_USE, T.FLT_INS, T.INV_INP, T.LINK, T.LNK_ATT, T.UA_LOAD
Token Verification Security Feature		
	FCS_COP.1 (FDP_ACC.2, FDP_ACF.1)**	P.CRYPT_STD, T.CRYPT_ATK, **T.LINK, T.UA_LOAD,** T.ACCESS
Card Content Management Security Feature		
	(FDP_ACC.2, FDP_ACF.1, FDP_ITC.2)**	T.UA_LOAD**, T.ACCESS**
Receipt Generation Security Feature		
	FCS_COP.1	P.CRYPT_STD, T.CRYPT_ATK, **T.UA_LOAD**

Security Feature and SFRs	Applicable Threats and Policies
Life Cycle State Management Security Feature	
FDP_ACC.1, FDP_ACF.1, FMT_MOF.1, FMT_MSA.1, FMT_MSA.2, FMT_MSA.3, FMT_MTD.3, FMT_REV.1	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.BRUTE-FORCE, **T.DIS_VUL**, T.FIRST_USE, T.FLT_INS, T.INV_INP, T.LINK, T.LNK_ATT, T.UA_LOAD
Key Management Security Feature	
Key Generation Services	
FCS_CKM.1	A.KEY_SUPP, P.CRYPT_STD, T.CRYPT_ATK, **T.LINK**
Key Distribution Services	
FCS_CKM.2	
Key Access Services	
FCS_CKM.3	
Key Destruction Services	
FCS_CKM.4	
Group 4	
Self Test Security Feature	
FPT_TST.1 FPT_AMT.1**	T.FORCD_RST, T.LNK_ATT, **T.P_ALTER**
Failure Management Security Feature	
FPT_FLS.1, FPT_RCV.3, FPT_RCV.4 FPT_RVM.1**	T.BRUTE-FORCE, T.ENV_STRS, T.FORCD_RST, T.I_LEAK, T.INV_INP, T.LNK_ATT

Security Feature and SFRs		Applicable Threats and Policies
Group 3		
Supervisor Security Feature		
	FIA_ATD.1, FMT_SMR.1, FPT_RVM.1 FMT_SMR.1**	A.ROLE_MAN, P.DATA_ACC, P.FILE_STR, T.ACCESS, **T.APP_FTN**, T.BRUTE-FORCE, T.INV_INP, T.LNK_ATT, T.UA_LOAD
Firewall Security Feature		
	FMT_MSA.3, FDP_ACF.1, FDP_IFC.1, FDP_IFF.1, FMT_MSA.1, FPT_SEP.1 (FDP_ACC.2, FMT_MSA.2,FMT_MSA.3, FMT_MTD.1)**	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.APP_FTN, T.BRUTE-FORCE, T.FIRST_USE, T.FLT_INS, T.INV_INP, T.LC_FTN, T.LINK, T.LNK_ATT, T.UA_LOAD
Object Reuse Security Feature		
	FDP_RIP.1 FDP_RIP.2**	**T.ACCESS**, T.BRUTE-FORCE, T.FORCD_RST, T.INV_INP, T.LNK_ATT,
Event Actions Security Feature		
	FAU_ARP.1, FAU_SAA.1	P.AUDIT, T.BRUTE-FORCE, T.INV_INP, T.LNK_ATT
Subject/Object Identification Security Feature		

Security Feature and SFRs		Applicable Threats and Policies
	FIA_ATD.1, FMT_MTD.1, FMT_SMR.1 (FIA_UID.2, FIA_USB.1)**	A.ROLE_MAN, P.DATA_ACC, P.FILE_STR, T.ACCESS, T.UA_LOAD
Card Audit Security Feature		
	FAU_LST.1, FAU_SAS.1*, FAU_SEL.1, FAU_STG.1, FAU_STG.3	P.AUDIT, P.IDENT, T.LNK_ATT
Group 2		
Sensors & Alarms Security Feature		
	FAU_ARP.1, FAU_SAA.1, FPT_FLS.1, FRU_FLT.2*	P.AUDIT, T.BRUTE-FORCE, T.CRYPT_ATK, T.ENV_STRS, T.I_LEAK, T.INV_INP, T.LC_FTN, T.LNK_ATT
Internal Communications Security Feature		
	FDP_IFC.1, FDP_ITT.1, FPT_ITT.1	P.DATA_ACC, P.FILE_STR, T.ACCESS, T.APP_FTN, T.BRUTE-FORCE, **T.CLON** , T.FLT_INS, **T.I_LEAK**, T.LC_FTN, T.LINK, T.LNK_ATT, T.P_PROBE, T.UA_LOAD
Memory Content Security Feature		
	FMT_LIM.1*, FMT_LIM.2*, FPT_SEP.1 FDP_SDI.2**	T.APP_FTN, T.BRUTE-FORCE, **T.CLON**, T.INV_INP, T.LC_FTN, **T.LINK**, T.LNK_ATT, T.P_ALTER**
Random Number Generation Security Feature		

Security Feature and SFRs		Applicable Threats and Policies
	FCS_RND.1*	T.CRYPT_ATK
Encryption/Decryption Security Feature		
	FCS_COP.1	P.CRYPT_STD, T.CRYPT_ATK
Group 1		
Tamper Resistance Security Feature		
	FPT_PHP.3	T.CLON, T.ENV_STRS, T.I_LEAK, T.P_ALTER, T.P_PROBE

Table 6. GP security and compatible PPs

End of Document