U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness

Version 1.03

Information Assurance Directorate

29 June 2007

This page intentionally left blank.

Foreword

1 This publication, "*U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*", is issued by the Information Assurance Directorate as part of its program to promulgate security standards for information systems. This protection profile is based on the "Common Criteria for Information Technology Security Evaluations, Version 2.3." [1]

2 Comments on this document should be directed to: ppcomments@iatf.net. The comments should include the title of the document, the page, the section number, and paragraph number, detailed comment and recommendations.

Table of Contents

List of Figures

List of Tables

1. Introduction

3 This section contains overview information necessary to allow a Protection Profile (PP) to be registered through a Protection Profile Registry. The PP identification provides the labeling and descriptive information necessary to identify, catalogue, register, and cross-reference a PP. The PP overview summarizes the profile in narrative form and provides sufficient information for a potential user to determine whether the PP is of interest. The overview can also be used as a stand-alone abstract for PP catalogues and registers. The "Conventions" section provides the notation, formatting, and conventions used in this protection profile. The "Glossary of Terms" section gives a basic definition of terms, which are specific to this PP. The "Document Organization" section briefly explains how this document is organized.

1.1 Identification

4 Title: U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness Version 1.03.

5 Registration: Information Assurance Directorate

6 Keywords: separation kernel, high robustness, data isolation, information flow control, partition, cryptography, commercial-off-the-shelf (COTS).

1.2 Overview

7 This "*U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness*" (SKPP) specifies the security functional and assurance requirements for a class of separation kernels [10]. Unlike those traditional security kernels which perform all trusted functions for a secure operating system, a separation kernel's primary security function is to partition (viz. separate) the subjects and resources of a system into security policy-equivalence classes, and to enforce the rules for authorized information flows between and within partitions.

8 Products that conform to this protection profile support *information flow control, resource isolation, trusted initialization, trusted delivery, trusted recovery* and *audit* capabilities. [6] The isolation and information flow policies are defined by the separation kernel's configuration data. A conformant product also includes the support tools and procedures used to accurately generate and securely distribute that configuration data. Specific assurance requirements are allocated to those support tools and procedures.

9 A separation kernel evaluated against this PP provides a highly robust foundation for system services and applications in mission-critical embedded systems, and a high degree of assurance for the enforcement of related security policies. Such policies include those for the management of classified and other high-valued information, whose confidentiality, integrity or releasability must be protected. For example, SKPP separation mechanisms, when integrated within a high assurance security architecture, are appropriate to support critical security policies for the Department of Defense (DoD), Intelligence Community, the Department of Homeland Security, Federal Aviation Administration, and industrial sectors such as finance and manufacturing.

10 The claim that products conforming to this protection profile are candidates for use in National Security Systems[1] derives from its basis in DoD and National Information Assurance (IA) guidance and policies. However, conformance to this protection profile, by itself, does not offer sufficient confidence that national security information is appropriately protected in the context of a larger system in which the conformant product is integrated. Designers of such systems must apply appropriate systems security engineering principles and techniques to afford acceptable protection for national security information. In particular, it is the responsibility of the system designer and authorized administrator to define support for a coherent application-level security policy in the separation kernel's configuration data, as well as to ensure that the configuration data itself is coherent and self-consistent. It is only with well-formed configuration data that the separation kernel can be expected to enforce mission-critical security policies. Requirements for coherent configuration data are indicated in the environmental objectives (OE.TRUSTED_FLOWS). The judgment as to whether a given instantiation of configuration data is well formed with respect to a particular application-level security policy is beyond the scope of this protection profile, but must be determined before secure deployment of an SKPP-based product.

1.3 Mutual Recognition of Common Criteria Certificates

11 The assurance requirements contained in this PP reflect techniques, activities, and evidence, appropriate for the establishment of trustworthiness in a compliant TOE for application in U.S. Government high robustness environments. The assurance requirements are comprised of both CC-defined assurance components from EAL6 and EAL7 and explicitly stated assurance components which are either new (i.e., not contained in the CC) or modifications of existing CC assurance components. Hence, this PP makes no EAL claim.

12 COTS separation kernels meeting the requirements of this profile provide a high level of robustness. Under the "Arrangement on the Mutual Recognition of Common Criteria Certificates in the field of Information Technology Security" document, only CC requirements at or below EAL4 are mutually recognized. Because this profile contains assurance components that exceed EAL 4, the US will recognize only certificates issued by the US evaluation scheme to meet this profile. Other national schemes are likewise under no obligation to recognize US certificates with assurance components exceeding EAL4.

1.4 Conventions

13 The notation, formatting, and conventions used in this protection profile are consistent with version 2.3 of the Common Criteria for Information Technology Security Evaluation. Font style and clarifying information conventions were developed to aid the reader. *Italicized* words are

[1] National Security Systems are systems that contain classified information or involve intelligence activities, involve cryptologic activities related to national security, involve command and control of military forces, involve equipment that is an integral part of a weapon or weapon system, or involve equipment that is critical to the direct fulfillment of military or intelligence missions.

defined in the glossary.

14 The CC permits four functional component operations: assignment, iteration, refinement, and selection to be performed on functional requirements. These operations are defined in Common Criteria, Part 1, Section 6.4.1.3.2 as:

- assignment: allows the specification of an identified parameter;
- refinement: allows the addition of details or the narrowing of requirements;
- selection: allows the specification of one or more elements from a list; and
- iteration: allows a component to be instantiated with varying context and/or operations.

15 *Assignments or selections* occurring in CC components left to be specified by the developer in subsequent security target documentation are italicized and identified between brackets ("[ ]"). In addition, when an assignment or selection has been left to the discretion of the developer, the text "assignment:" or "selection:" is indicated within the brackets. Assignments or selection created by the PP author (for the developer to complete) are bold, italicized, and between brackets ("[ ]"). CC selections completed by the PP author are underlined and CC assignments completed by the PP author are bold.

16 *Refinements* are identified with "**Refinement:**" right after the short name. They permit the addition of extra detail when the component is used. The underlying notion of a refinement is that of narrowing. There are two types of narrowing possible: narrowing of implementation and narrowing of scope [1]. Additions to the CC text are specified in bold. Deletions of the CC text are identified in the "End Notes" with a bold number after the element ("**8**").

17 *Iterations* are identified with a number inside parentheses ("(#)"). These follow the short family name and allow components to be used more than once with varying operations.

18 *Explicit Requirements* are used when the Common Criteria does not offer suitable requirements to meet the PP needs. The convention for explicit requirements is the same as that used in the CC. To ensure these requirements are explicitly identified, the ending "_EXP" is appended to the newly created short name.

19 *Application Notes* are used to provide the reader with additional requirement understanding or to clarify the author's intent. These are italicized and usually appear following the element needing clarification.

20 Table 1.1 provides examples of the conventions (explained in the above paragraphs) for the permitted operations. The examples in Table 1.1 are not from this PP.

Table 1.1. Functional Requirements Operation Conventions

Convention	Purpose	Operation
Bold	The purpose of bolded text is used to alert the reader that additional text has been added to the CC. This could be an assignment that was completed by the PP author or a refinement to the CC statement. Examples: FDP_IFC.2.1 The TSF shall enforce **Information Flow Control policy** on **subjects and all resources** and on all operations that cause information to flow to and from subjects covered by the SFP. FAU_GEN.1.1 **Refinement:** The TSF shall be able to generate **audit data for** the following auditable events:	(Completed) Assignment or Refinement
Italics	The purpose of italicized text is to inform the reader of an assignment or selection operation to be completed by the developer or ST author. It has been left as it appears in the CC requirement statement. Examples: FAU_ARP.1.1 The TSF shall take [assignment: *list of least disruptive actions*] upon detection of a potential security violation. FDP_RIP.2.1 The TSF shall ensure that any previous information content of a resource is made unavailable upon the *[selection: allocation of the resource to, deallocation of the resource from]* all exported resources.	Assignment (to be completed by developer or ST author) or Selection (to be completed by developer or ST author)
<u>Underline</u>	The purpose of underlined text is to inform the reader that a choice was made from a list provided by the CC selection operation statement. Example: FAU_SEL.1.1 The TSF shall be able to include or exclude auditable events from the set of audited events based on the following attributes: **a)** <u>subject identity,</u> **b)** <u>event type,</u> **c) success of auditable security events, and** **d) failure of auditable security events.**	Selection (completed by PP author)

Convention	Purpose	Operation
Parentheses (Iteration #)	The purpose of using parentheses and an iteration number is to inform the reader that the author has selected a new field of assignments or selections with the same requirement and that the requirement will be used multiple times. Iterations are performed at the component level. The component behavior name includes information specific to the iteration between parentheses. Example: **5.4.1.1 Explicit: Management of TSF Data (for Configuration Data) (FMT_MTD_EXP.1(1))** FMT_MTD_EXP.1.1(1) The TSF shall restrict the ability to select and activate the TSF policy configuration data to authorized subjects. **5.4.1.1 Explicit: Management of TSF Data (for General TSF Data) (FMT_MTD_EXP.1(2))** FMT_MTD_EXP.1.1(2) The TSF shall prevent modification of TSF policy configuration data.	
Explicit: (**_EXP**)	The purpose of using **Explicit:** before the family or component behavior name is to alert the reader and to explicitly identify a newly created component. To ensure these requirements are explicitly identified, the "**_EXP**" is appended to the newly created short name and the component and element names are bolded. Example: **5.4.1.1 Explicit: Management of Security Functions Behavior (FMT_MOF_EXP.1)** **FMT_MOF_EXP.1.1** The TSF shall restrict the ability to enable and disable audit generation to the configuration data.	Explicit Requirement

Convention	Purpose	Operation
Endnotes	The purpose of endnotes is to alert the reader that the author has deleted Common Criteria text. An endnote number is inserted at the end of the requirement, and the endnote is recorded on the last page of the section. The endnote statement first states that a deletion was performed and then provides the rationale. Following is the family behavior or requirement in its original and modified form. A strikethrough is used to identify deleted text and bold for added text. A text deletion rationale is provided. Examples: Text as shown: FAU_SAA.1.2 **Refinement:** The TSF shall monitor the accumulation or combination of the following events known to indicate a potential security violation: [assignment: *subset of defined auditable events*].**1** Endnote statement: A deletion of CC text was performed in FAU_SAA.1.2. Rationale: The words "enforce the following rules for monitoring audited events: a)" were deleted for clarity and flow of the requirement. Additionally the assignment was moved from the middle of the requirement to the end for clarity and flow of the requirement. FAU_SAA.1.2 **Refinement:** The TSF shall ~~enforce the following rules for monitoring audited events: a)~~ **monitor the** accumulation or combination of [~~assignment: *subset of defined auditable events*~~] **the following events** known to indicate a potential security violation: [assignment*: subset of defined auditable events*].	Refinement

1.5 Glossary of Terms

21 This profile includes terms from the Common Criteria [1] by reference. Other terms are described in this section to aid in the understanding and application of the requirements.

Administrator, authorized	Any person authorized to configure, install, integrate, or maintain the Target of Evaluation (TOE) or its data. An authorized administrator is one of several possible trusted individuals: see *trusted individual.*
Assurance baseline	The collection of all the evaluation evidence and evaluation materials at the time of the TOE evaluation [8].

Assurance maintenance plan	The plan that defines the assurance maintenance process for the *Target of Maintenance* (TOM) and outlines the technical approach to be taken to maintain the assurance gained during the evaluation process [8].
Configuration change – Activation – Constrained – Dynamic – Offline/Static – Selective – Total – Unconstrained Next configuration	A *configuration change* modifies the TOE configuration data, thereby changing the operational configuration of the TOE. For example, the *Partitioned Information Flow Policy* could be modified. A configuration change involves two abstract actions: the first defines what the *next configuration* is going to be, and the second activates the change. The actual *activation* can occur offline, i.e., *a static configuration change,* or during an execution session, i.e., a *dynamic configuration change*. Replacing all configuration data with data from a configuration vector is called a *total configuration change*. Modification of individual elements of configuration data is called a *selective configuration change*. Selective changes can be *constrained* by the TOE to a subset of the configuration data, or can be *unconstrained*. *Static configuration changes* are manifested by the initialization function. Dynamic configuration changes can occur via the initialization function (i.e., through a *restart*), or through a *reconfiguration function*.

Configuration data Configuration vector TSF internal – Configuration vector set TSF internal –	*Configuration data* is a set of values in the TOE Security Functions (TSF) that defines the *initial secure state* and the *operational configuration* of the TOE. The trusted initialization function transforms a configuration vector into the configuration data. The TSF relies on the configuration data to maintain *secure state* during runtime. The configuration data is a subset of *TSF data*, and includes, but is not limited to, the following descriptions: • assignment (binding) of subjects and exported resources (including portions of addressable memory) to partitions • *Partitioned Information Flow Policy* • processor time and memory allocation quotas • audit configuration parameters • clock parameters (e.g., time zone, granularity) • execution periods for self-test A *Configuration vector* is a set of values that is suitable for use by the initialization function to create TSF configuration data. The *Configuration vector set* (CVS) is a collection of multiple configuration vectors. The configuration vector set is imported to the TSF by the initialization function. An imported vector is called a *TSF internal vector*. An imported vector set is called a *TSF internal vector set*. There must be at least one TSF internal vector defined at all times. The configuration vector set is used by the TSF to support configuration changes. The configuration vector set must be protected from modification to ensure the integrity of the TOE configuration.
Configuration function	The procedures and automated mechanisms employed to generate the TSF configuration vectors and corresponding integrity seals. The output of the configuration function can take different forms, for example: placement of the implementation or configuration information onto suitable media (e.g., CD, ROM or flash memory). The configuration function may be combined with the *load function* if the configuration data is compiled as part of the TSF implementation.
Controlled operation	An operation available only to *authorized subjects*. See *subject*.
Covert channel	An unintended and/or unauthorized communications path that can be used to transfer information in a manner that violates a security policy [6]. Covert channels allow transfer of information through indirect access by subjects to internal resources; whereas, a transfer of information in violation of the security policy through exported resource(s) would be a TSF flaw.

Delivery, trusted	Procedures and automated mechanisms employed to deliver a copy of the TOE from the TOE developer to the customer, such that the customer copy is assured to be the same as the developer's master copy.
Execution session	An *execution session* is the set of states from initialization to either (1) completion of the shutdown of the TOE or (2) commencement of a *restart* of the TOE. Both *operational mode* and *maintenance mode* can occur in a single execution session.
Flow Flow, mode of Mode, direction Mode, attribute Flow, partition	Information movement in the TOE may be initiated by the TSF (e.g., appending TSF status or audit data to an exported resource) or by a subject. Subjects invoke information movement via "controlled operations," [2] such as write_resource. A controlled operation may cause one or more *flows*, each of which characterizes information moving between a subject and a unique exported resource, which when projected to partition space (i.e., to the related partitions), comprises a flow between the subject's partition and the exported resource's partition. A flow is defined as a [partition/subject, partition/exported resource, mode] triplet. The mode of the flow consists of a direction and an attribute. The direction indicates data movement from the subject and its partition to the exported resource and its partition or vice versa. The attribute has one of three values: ALLOW, DENY, and NULL. This results in six possible values for the mode of a flow: e.g., READ-ALLOW, READ-DENY, READ-NULL, WRITE-ALLOW, WRITE-DENY, WRITE-NULL. NULL is interpreted as an implicit DENY for Partition rules and as "don't care" for Subject-Exported Resource rules. Abstractly, there is a flow definition for both directions of data movement for every partition/partition pair and for every subject/exported-resource pair in the system. Thus, the "combined" mode for each of these pairs has nine possible values: [READ-ALLOW, with WRITE-ALLOW or WRITE-DENY or WRITE-NULL], [READ-DENY, with WRITE-ALLOW or WRITE-DENY or WRITE-NULL], [READ-NULL, with WRITE-ALLOW or WRITE-DENY or WRITE-NULL]. In the runtime configuration data, or in a given configuration vector, a 'blank' or unassigned value for a particular direction and pair is interpreted as NULL. The terminology used for mode "direction" (such as send and receive, write and read or modify and observe) and "attribute" can vary from TOE to TOE, as long as the semantics described here are expressible. ST authors may further restrict the mode primitives to establish policies regarding execution, device control, etc.

Initialization function	The procedures and automated mechanisms that establish the TSF in an *initial secure state*. The initialization function includes the TOE boot mechanism that establishes the TSF *security domain* and brings the software portion of the TSF implementation and TSF data into the TSF security domain (e.g., read it from disk, from ROM, or from flash memory into a memory space allocated for TSF functions and data). Initialization can occur as a result of system power-on or from a *restart*.
Load function	The procedures and automated mechanisms to convert the software portion of the TSF implementation and/or configuration vectors into a TOE-useable form. The load function can take different forms, including: compilation of configuration data as part of the TSF implementation; or the insertion or installation of the media into the TOE hardware at either the TOE developer or customer site.
Maintenance mode	A contiguous period during an execution session when operational mode functions are restricted, or recovery functions are available that are not available during operational mode, or both. The intended use of maintenance mode is to enable the TOE to return to a secure state, or prevent the TOE from entering an insecure state. The functions restricted or made available during maintenance mode are specific to the developer's strategy for failure recovery.
Operational configuration	The *operational configuration* of the TOE determines its behavior during an execution session, as specified in the configuration data.
Operational mode	A runtime mode in which all security functions of the TOE are available.

Partition	Given a set *A* of elements *a*, an abstract *PARTITION* (upper case) of A is a collection of subsets of those elements such that each $a \in A$ belongs to exactly one of the subsets. In this document, *A* is the set of all exported resources, and each subset is called a *partition* (lower case). The members of each partition are designated by the configuration data to be treated equivalently with respect to the *Partition Rule* of the *Partitioned Information Flow Policy (PIFP)* (see *PIFP*, FDP_IFF), thus forming an *equivalence class*. Each individual subject and each individual exported resource is assigned (*bound*) to exactly a single partition. Zero or more subjects and zero or more exported resources may be bound to a particular partition, thus a partition may be empty or null (i.e., an empty set) – for example it may help to simplify administration of the TOE to have a placeholder for emergency activity (see dynamic configuration change). Also, a partition may be configured such that its subjects and exported resources may be only written to or only read from. Note that a partition is not an active entity: see *subject*.
Partitioned information flow policy (PIFP)	The policy enforced by the TSF that provides for the separation of partitions via controlled information flows between partitions, and between the subjects and exported resources allocated to those partitions when required. Note that the PIFP defines only the authorizations for flows between partitions and between the subjects and exported resources allocated to those partitions. Flow authorizations may be expressed as a combination of explicit positive authorizations and implicit or explicit negative authorizations. The PIFP does not define the size and locations of partitions or the allocation of subjects and exported resources to partitions. (See *configuration data*).
Platform Platform Component	The physical components of the TOE; the TOE hardware and accompanying firmware. The TOE *platform* is defined by the ST author to consist of one or more *platform components*, each of which is independently procurable. Examples of platform components are: the complete platform, a CPU, and a disk drive.
Platform interface – External – Internal	An *internal platform interface* is accessible only to TOE components, whereas an *external platform interface* is one that is directly accessible to entities outside the TOE, as well as to TOE components.
Principle of Least Privilege	This principle requires that each subject and TSF internal module be granted the most restrictive set of privileges needed for the performance of authorized tasks [12]. The application of this principle limits the damage that can result from accident, error, or unauthorized use [5].

Reconfiguration function	A *reconfiguration function* causes a configuration change during an execution session without passing control to the initialization function.
Residual information protection (RIP)	Protection of information that has been logically deleted or released, which is not available to subjects, but may still be present within the system and may be recoverable. It also applies to protection of resources that are serially reused by different subjects within the system. [2] Note that residual information protection applies to all resources in the TOE: those exported by the TSF as well as those internal to the TSF: see *resource, exported resource, internal resource.*
Resource Resource, Exported Resource, Internal	*Resources* are the totality of all hardware, firmware and software and data that are executed, utilized, created, protected or exported by the TSF. *Exported resources* are those resources to which an explicit reference is possible via a TSF interface (TSFI), e.g., the programming or configuration interface. See also, *Subject.* *Internal resources* are those resources used exclusively by the TSF, and which have no explicit reference via a TSF interface.
Restart	A *restart* occurs when the initialization function is invoked during an execution session without cycling the power off and on.
Secure state Secure state, initial	The meaning of "secure state" is dependent on the TSP model. "Secure state" in this protection profile means that (1) the TSF data is consistent and uncorrupted, and the TSF can correctly enforce the policy represented by the TSP model, and (2) all TSF actions and transitions subsequent to the *initial secure state* (a) are allowed by the TSP model, or, (b) are successfully rolled back to some previous secure state, or, (c) result in a transition of the TSF to the initial secure state. The *initial secure state* is the secure state arrived at after a successful initialization. There is only one initial secure state associated with an execution session.

Security domain	A *security domain* is a bounded and protected set of resources. The TSF maintains separate security domains for its own protection, and provides a security domain for each subject. The security domains of the TSF are distinct from (i.e., non-overlapping) the security domains of subjects and other entities external to the TSF. The security domain of a subject includes the exported resources that it is allowed to use, including the memory and I/O addresses (viz., the "address space") that it may access. A subject's security domain may include resources in partitions other than its own. The security domains of different subjects may overlap.
Separation kernel	Hardware and/or firmware and/or software mechanisms whose primary function is to establish, isolate and separate multiple partitions and control information flow between the subjects and exported resources allocated to those partitions.
Subject Subject, authorized	An active entity within the TSF scope of control (TSC) that causes operations to be performed. A *subject* is an abstraction created by the TSF and exported at the TSFI. A subject is a type of exported resource. In this protection profile, there are runtime TOE security administrative functions that can only be performed by *authorized subjects*, which are designated as such in the configuration data. In contrast, the PIFP rules define the authorized flows for <u>all</u> subjects, including authorized subjects.
Target of Maintenance (TOM)	The subject of the assurance maintenance process, comprising an evaluated TOE together with any changes to the associated *assurance baseline* [8].
TSF data	Data created by and for the TOE that, when in a TSF security domain, affects the operation of the TSF. TSF data includes but is not limited to internal data structures, configuration data, and TSF-generated data.
TSF internal vector TSF internal vector set	See *configuration vector* and *configuration vector set* above.
Trusted individual	A person who performs procedures upon which the security of the TOE and the processes used to develop the TOE may depend. The roles and responsibilities of these persons may include those that develop, configure, install, manage, operate and maintain the TOE, as required for a specific TOE type developed to execute in specific operational environments and contexts. See *authorized administrator*. The requirements for establishing the trustworthiness of trusted individuals are allocated to the environment. See the security objective OE.TRUSTED_INDIVIDUAL.

User	The term “user” has different meanings, depending on context [1]. During runtime, subjects interact with the TOE, and are referred to as users, in terms of what they are allowed to do or observe. Also, TOE-application developers (including integrators in some cases), who write the code that defines the behavior of subjects, are referred to as users, in terms of how they understand, or how they may interact with the TOE via subjects. Similarly, trusted individuals are users, in terms of how they understand, or how they may interact with the TOE, e.g., via administrative functions. In this sense, trusted individuals are the only “human users” [1] of the TOE. The customer is also a user, in the sense that they are the recipients of trusted delivery, and as data owners, are the ultimate beneficiaries of the security properties provided by the TOE.

1.6 Document Organization

22 *Section 1* provides the introductory material for the protection profile.

23 *Section 2* describes the Target of Evaluation in terms of its envisioned usage and connectivity.

24 *Section 3* defines the expected TOE security environment in terms of the threats to its security, the security assumptions made about its use, and the security policies that must be followed.

25 *Section 4* identifies the security objectives derived from the threats and policies.

26 *Section 5* identifies and defines the security functional requirements from the CC that must be met by the TOE in order for the functionality-based objectives to be met.

27 *Section 6* identifies the TOE security assurance requirements.

28 *Section 7* provides a rationale to explicitly demonstrate that the information technology security objectives satisfy the policies and threats. Arguments are provided for the coverage of each policy and threat. The section then explains how the set of requirements are complete relative to the objectives, and that each security objective is addressed by one or more component requirements. Arguments are provided for the coverage of each objective.

29 *Section 8* identifies background material used as reference to create this profile.

30 *Appendix A* defines frequently used acronyms.

31 *Appendix B* identifies cryptographic standards, policies and other publication referenced in this PP.

32 *Appendix C* provides a rationale for the IFC/IFF requirements.

33 *Appendix D* provides a rationale for the secure state definition.

34 *Appendix E* provides a description of the various types of TSF data.

35 *Appendix F* provides an example scenario for TOE functions.

36 *Appendix G* provides a rationale for platform assurance requirements.

2. Target of Evaluation (TOE) Description

2.1 Product Type

37 This protection profile ("SKPP") specifies requirements for a separation kernel TOE, inclusive of its underlying platform[2], assured to High Robustness criteria (refer to Section 6 for a mapping of High Robustness to Common Criteria Security Assurance Requirements). To be highly robust, the SKPP requires that the functionality, architecture and design of the separation kernel be minimized in size and complexity. The resulting TOE is suitable for protecting highly sensitive information (see Use of High Robustness, Section 2.8). Its core functional requirements include:

- Protection of all *resources* (including CPU, memory and devices) from unauthorized access
- Separation of *internal resources* used by the TSF from exported resources made available to *subjects*
- Partitioning and isolation of *exported resources*
- Mediation of information flows between *partitions* and between exported resources
- Audit services

38 The separation kernel allocates all exported resources under its control into partitions. The partitions are isolated except for explicitly allowed information flows. The actions of a subject in one partition are isolated from (viz., cannot be detected by or communicated to) subjects in another partition, unless that *flow* has been allowed. The partitions and flows are defined in *configuration data*. Note that "partition" and "subject" are orthogonal abstractions. "Partition," as indicated by its mathematical genesis, provides for a set-theoretic grouping of system entities, whereas "subject" allows us to reason about the individual active entities of a system. Thus, a partition (a collection, containing zero or more elements) is not a subject (an active element), but may contain zero or more subjects.

39 The TOE provides to its hosted software programs high-assurance partitioning and information flow control properties that are both tamperproof and non-bypassable. These capabilities provide a configurable trusted foundation for a variety of system architectures. For example, in one class of system security architecture, software programs enforce application-level (vs. kernel-level) security policies, within the constraints of the separation kernel's policy. Examples of hosted software programs include multilevel secure reference monitors, guards, device drivers, file managers, and message-passing services, as well as those that implement traditional operating

[2] The existing functional and assurance requirements contained in this protection profile are broad enough to accommodate a uni-processor as well as a multi-processor TOE. However, in a multi-processor implementation there is an expected increase in TOE complexity and, therefore, the level of difficulty in generating the required evidence for some areas (e.g., security policy definition/modeling, architecture requirements, covert channel analysis, testing, processor state consistency).

system, middleware and virtual machine monitor abstractions.

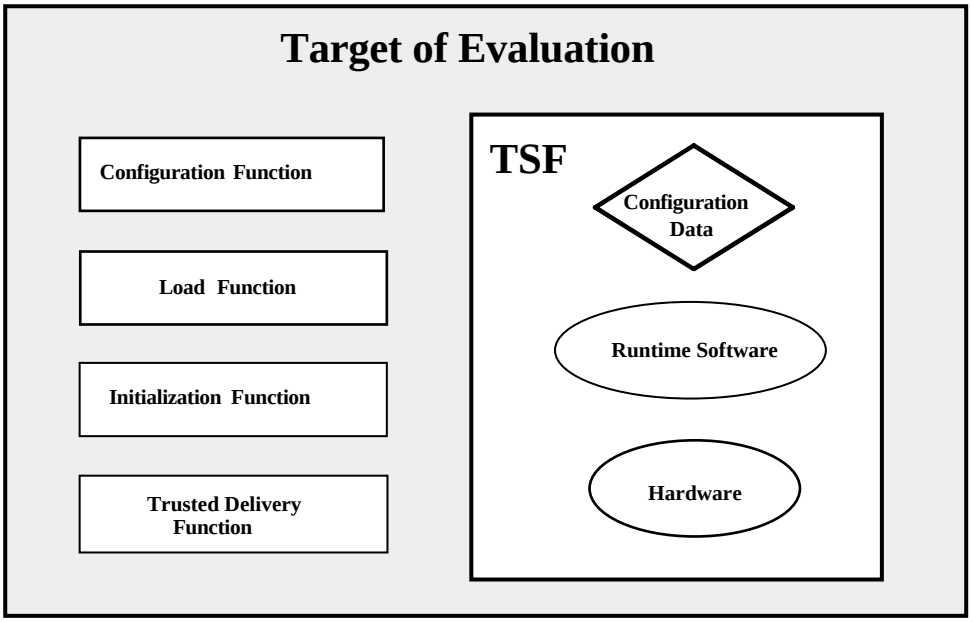


Figure 2-1. Allocation of TOE Components

40 Figure 2-1 depicts the components that comprise the TOE and TSF. Each TOE component outside of the TSF serves a role in establishing the TSF's initial secure state. After initialization, the TSF will enforce the defined policy. The security functional requirements (SFRs) for the TSF are found in Section 5. Functional requirements for non-TSF TOE components, as well as assurance requirements for all TOE components, are found in Section 6. The role of non-TSF functions in establishing the initial secure state of the TSF are as follows:

- **Trusted Delivery:** The TOE developer employs cryptographically-based trusted delivery functions and procedures to deliver the TOE to the customer. The customer may be a system integrator, application developer or end user. Trusted delivery is used for the initial product distribution as well as for updates. See Section 2.5.

- **Configuration**: The TOE's *configuration function* translates human-readable (e.g., ASCII) representations of *configuration vectors* into machine-readable (e.g., binary) format. An *authorized administrator* uses the configuration function and related procedures to generate, validate, and protect the integrity of each configuration vector.

- **Load**: A *trusted individual* employs the *load function* and related procedures to transfer ("load") the software implementation and configuration vectors into a form that is accessible by the TOE. An example is the placement of configuration vectors into flash memory. TOE software and configuration vectors (i.e., a *configuration vector set*) may be loaded together or separately. Loading may occur in the customer IT environment as well as in the TOE developer IT environment (see example in Appendix F). The load function may also be used as part of offline trusted recovery.

- **Initialization:** A *trusted individual* or IT mechanism in the TOE environment starts

the TOE *initialization function* (e.g. via a power-on switch or electronic signal). At this time, the initialization function: verifies the integrity of the TSF code and data; transfers that code and data into the TSF security domain; establishes the TSF in *operational mode* in its *initial secure state* or establishes the TSF in *maintenance mode* in support of recovery actions to reestablish *secure state*. In this process, the initialization function uses the information in one of the configuration vectors to establish the *configuration data*.

41 Figure 2-2 is a conceptual illustration of how configuration information is created and transformed into configuration data and TSF *internal vectors*. Other TOE functionality, including that of the TSF, is discussed in subsequent sections.

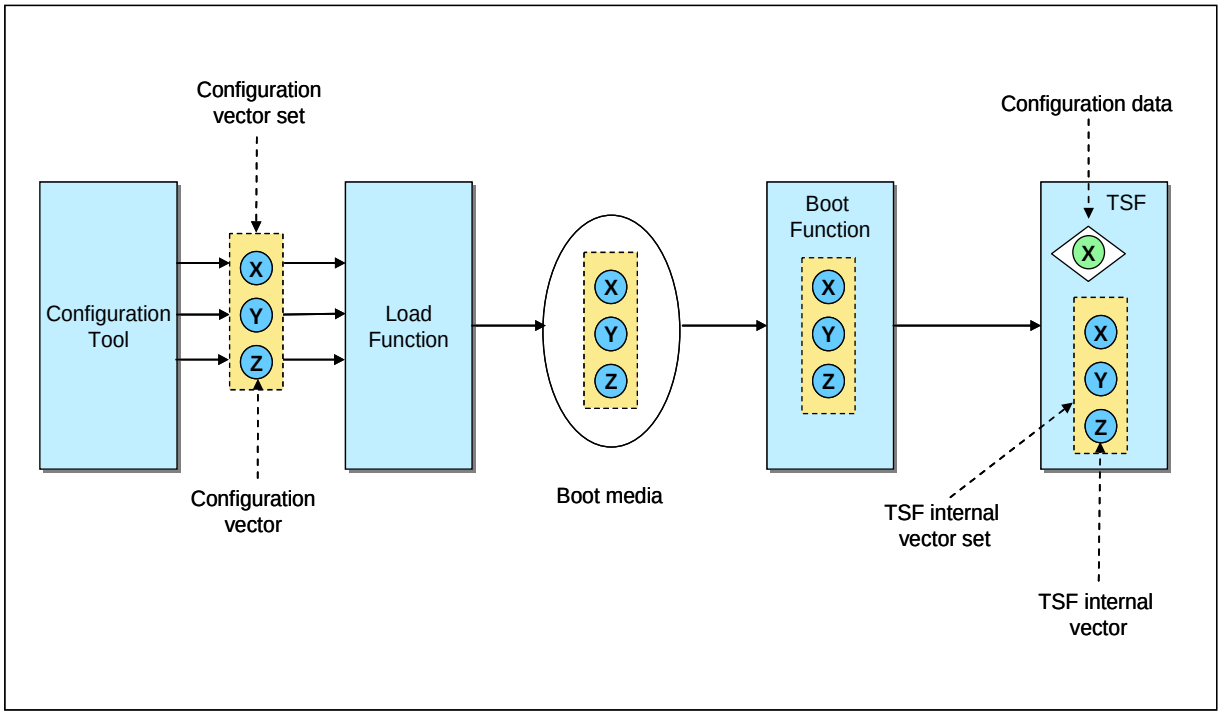


Figure 2-2. Example Configuration Data Transformation

2.2 General TOE Functionality

42 A TOE includes the following security features:

- Information flow control that enforces strict partition isolation, with the exception of explicit interactions specified by the configuration data
- Cryptographic mechanisms that provide functions to verify the integrity of TSF code and data during trusted delivery
- Trusted initialization and recovery functions
- Detection and response to security function failures
- Generation of audit data

43 Among the features not required are:

- User interfaces during an *execution session* or initialization
- Identification and Authentication which mandates authorized users to be uniquely identified and authenticated by the TSF
- Discretionary Access Control (DAC) which restricts access to objects based on the identity of subjects and/or groups to which they belong, and allows authorized users to specify protection for objects that they control
- Cryptographic services for applications to encrypt, decrypt, hash, and digitally sign data as it resides within the system and as it is transmitted to other systems
- Complete physical protection mechanisms

44 These features, if required in a system utilizing the TOE, must be provided by that system. Alternatively, the developer can extend the TOE functionality as defined in this protection profile, for example, through incorporation of additional requirements in the ST.

2.3 TOE Concepts

45 The goal of the separation kernel is to virtualize and allocate shared *resources* such that each partition encompasses a resource set that appears to be entirely its own. To achieve this ideal for resources that can only be accessed by one subject at a time, such as the CPU, the TSF must ensure that the temporal usage patterns from different partitions are not apparent to each other (e.g., through static "periods processing"). For resources such as memory, which do not require mutual exclusion to the whole, the TSF might achieve isolation by allocating physically distinct portions of the resource to different partitions. Furthermore, TSF utilization of its own *internal resources* must also preserve the desired isolation properties. Subjects, and resources made available to subjects by the TSF, are called *exported resources*.

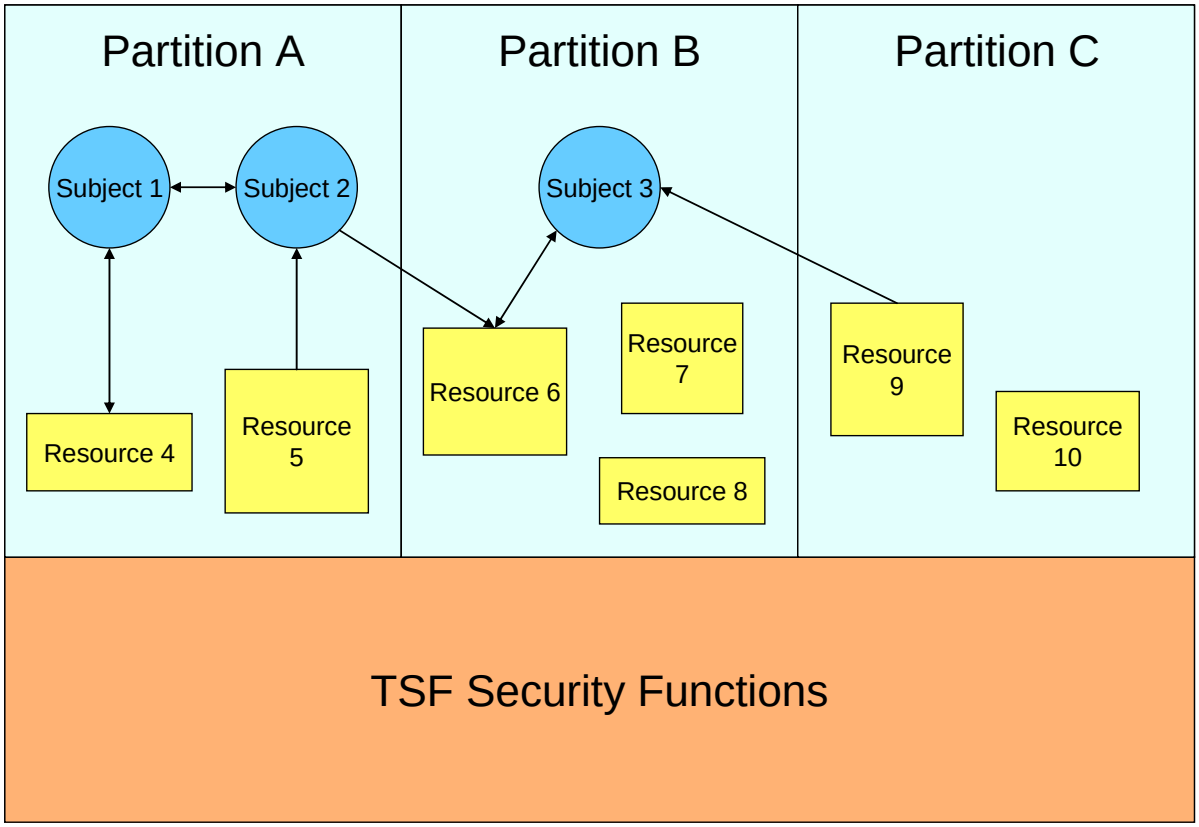


Figure 2-3. Allocation of TOE Resources

46 The TOE rules for isolation are referred to as the *Partitioned Information Flow Policy* (PIFP). The PIFP defines the authorizations for information flow between partitions and between subjects and exported resources. The *mode* or direction of the *flow* – such as send, receive, read (including execute-only, which could be further restricted by the ST author), write or read-write – indicates whether information flows from the subject to the exported resource (e.g., write) or from the resource to the subject (e.g., read), or both. Thus, an information flow is defined as a <partition/subject, partition/exported resource, mode> triplet. Note that the exported resource may be another subject. By default, the TOE allows no information flow between partitions or between subjects and exported resources.

47 Figure 2-3 shows a hypothetical example of the allocation of subjects and other exported resources to partitions. The resources inside of each rectangle are bound to that partition. Allowed information flow is indicated by the directed arrows. Inter-partition flows are also shown, for example, Subject 2 is allowed to write Resource 6, and Subject 3 is allowed to read Resource 9. This example is intended to illustrate the application the *Least Privilege Abstraction* of the PIFP (see Section 2.3.2). With this policy abstraction, subject(s) in a partition can have different access rights to resources in the same or different partitions. Resources 7, 8 and 10 are included to illustrate this finer grained control of information flow.

48 The *configuration data* may be comprised of:

- Definition of partitions, both in terms of the allocation of exported resources to partitions

(including the allocation of subjects and physical memory to the partition), partition identity, and partition quotas for time and space resource consumption

- Definition of the authorizations for information flow between partition-pairs and between subject-exported resource pairs
- Designation of authorizations for subjects
- Definition of test parameters to be used by the abstract machine test and TSF self-test mechanisms
- Definition of audit function behavior

2.3.1 Principle of Least Privilege

49 The Principle of Least Privilege (PoLP) is a foundational element in the design of high assurance systems and the rationale for its use is thoroughly documented elsewhere [5]. Several aspects of this principle are especially germane to understanding the significant requirements for least privilege in the SKPP. These requirements involve both the TSF internal structure and the ability of the TSF to mediate the actions of subjects.

50 Assume that a given TSF is made up of several components. If one of the TSF components became corrupted, the adverse effects on other components would be less pervasive if PoLP has been followed in the design. Second, because the privileges afforded each component will be minimal with respect to the overall policy, security analysis of the TSF is less complex. And finally, if a TSF limits each subject to utilize only those flows that it requires to complete its function, the TOE audit functions will be able to more accurately record information associated with the causes of various actions (see P.ACCOUNTABILITY in Section 3).

51 Thus, the ability to achieve the goals of confinement of damage, evaluatability and accountability is governed by both the degree to which the TSF is structured with least privilege and the granularity with which PoLP is applied to resources exported at the TSF interface.

2.3.2 Partitions and the Partitioned Information Flow Policy (PIFP)

52 A *partition* is an abstraction implemented by the TSF from resources under its control. Each partition is allocated zero or more *exported resources* (e.g., programs, tasks, processes, threads, files, buffers, devices, etc), as defined in the configuration data.

53 The TSF may initiate information flow (e.g., appending TSF status or audit data to an exported resource) or a subject may do so. Subjects invoke information flow in the TOE via "controlled operations."[2]

54 The PIFP is based on the following fundamental principles:

A. The scope of the PIFP includes all exported resources; there are no exemptions.

B. A controlled operation may result in multiple information flows, in which case, the PIFP must explicitly authorize each flow. Therefore, none of the flows associated with the controlled operation may occur if any one (1) of the multiple information flows is unauthorized. The purpose of this restriction is to reduce the complexity of the TOE.

C. The SKPP defines two (2) partition policy abstractions, each of which represents a different granularity of policy enforcement with respect to information flow.

55 The policy abstractions are the partition abstraction and the least privilege abstraction. The policy abstraction most appropriate for a given partition is determined by the heterogeneity of flows *required* by the subjects in that partition (per PoLP, the flows **allowed** by the configuration data may never exceed the flows that are **required** by the functionality of the subjects). Recall, each flow is a triplet: <partition/subject, partition/exported resource, mode>).

56 The two policy abstractions are defined below, along with examples to illustrate how to interpret the PIFP for each type of partition.

57 **PARTITION ABSTRACTION: The subjects in a partition have homogeneous requirements for access, on a per-partition basis, to exported resources.**

58 In the partition abstraction, the TSF enforces the same restrictions on all subjects bound to a given partition, that is, the flow authorizations assigned to that partition apply equally to all subjects in that partition. Administrative procedures (as specified in AGD_ADM) are relied upon to ensure that the partition can only be configured with subjects whose functionality requires the exact same set of access rights to all of the exported resources of a given partition (i.e., the same or a different partition). The TSF enforces restrictions on the subjects in the partition at the granularity of access to partitions, using partition identities, per the Partition Rule defined in FDP_IFF.1.2-NIAP-0407. For example, if any one subject of a partition requires access to resources in another partition, then all subjects in that partition must have the same access to all of those resources. Since (1) all of the subjects would have identical subject-exported resource authorization to cause a flow (e.g., each subject in partition *p* is allowed to read each resource in partition *q*), and (2) the Partition Rules are set minimally as required by PoLP, then the Partition Rules for *p* would be identical to the rules for the subjects in *p* (viz., *p* read *q*) and the Partition Rule can be used to indicate the allowed flows for all of the subjects in partition *p* without weakening PoLP.

Table 2-1. Access Matrix Representation for Partition Abstraction

		Partition D Resources			**Partition E Resources**			**Partition F Resources**			
		R1	**R2**	**R3**	**R4**	**R5**	**R6**	**R7**	**R8**	**R9**	**R10**
Partition D Subjects	**S1**	RW	RW	RW	R	R	R	W	W	W	W
	S2	RW	RW	RW	R	R	R	W	W	W	W
	S3	RW	RW	RW	R	R	R	W	W	W	W
Partition E Subjects	**-**	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	-	-	-	-
Partition F Subjects	**S4**	-	-	-	RW	RW	RW	R	R	R	R
	S5	-	-	-	RW	RW	RW	R	R	R	R
	S6	-	-	-	RW	RW	RW	R	R	R	R

59 An example of a configuration that meets this restriction is shown in Table 2-1 (where 'R' = read, 'W' = write; the ST author may further refine these modes). Here, all of the subjects of

partition D require read-only access to the all of the exported resources in partition E, etc., and there are no subjects in partition E.

60 Assurance requirements stipulate that procedures or functions must be provided to ensure that these configuration restrictions are upheld. Also, the partition identifier may equate to an identifier for the set of subjects assigned to that partition.

61 **LEAST PRIVILEGE ABSTRACTION: The subjects in a partition have heterogeneous requirements for access to exported resources.**

62 For this case, the partition may have subjects that require different access rights to support their functionality. Information flow is enforced using separate identities for each subject, exported resource, and the partition itself. The TOE supports PoLP by providing the ability to restrict the subjects in the partition in terms of both partition-partition flows and subject-exported resource flows. The least privilege abstraction requires that both partition-pair and subject-exported resource pair authorizations are used to determine if a flow mode is allowed. Additionally, the least privilege abstraction requires that the subject-exported resource pair authorization takes precedence over the partition-pair authorization, i.e., the subject-exported resource pair authorization overrides the partition-pair authorization. This precedence relationship does not require that all subjects have fine-grained subject-exported resource authorizations. For the case where fine-grained subject-exported resource authorizations do not exist, the partition-pair authorizations apply. This allows for per-partition grouping of subjects irrespective of their access requirements. The least privilege abstraction subject-exported authorization rules also must be expressive enough to support the Principle of Least Privilege such that the TSF can differentiate between positive and negative authorizations for each requested flow.

63 In the least privilege abstraction, the TSF enforces restrictions on the subjects in the partition at the granularity of access to exported resources as defined by the Subject-Exported Resource Rule defined in FDP_IFF.1.2-NIAP-0407.

64 Table 2-2 provides a reference example for implementation of the Least Privilege Abstraction showing the inter-relationship of the partition-partition and subject-resource authorization that comprise the abstraction. Tables 2-2a-2e provide examples of the versatility of the Least Privilege abstraction, again, using an access matrix representation (where ‘r’ = read, ‘w’ = write and all partitions use the least privilege abstraction.

65 The examples use the following terminology and conventions:

- A subject-resource flow is expressed in the form :
  - `Flow: [S, R, M]`

    where the argument S is a subject, R is a resource and M is a mode.
- The partition of a resource, r2, is indicated as:
  - `r2.P`
- The partition of a subject, s3, is indicated as:
  - `s3.P`
- `PA` is a matrix of rules expressing the partition-to-partition policy
- `SA` is a matrix of rules expressing the subject-exported resource policy

- Authorizations are expressed using a three-value logic for each possible mode (e.g., `r`, `w`), where for the example of a '`w`' mode:
  - `w = ALLOW Write` (explicit allow)
  - `!w = DENY Write` (explicit deny)
  - *`blank`* `= NULL` (implicit DENY for `PA`; don't care for `SA`)
- Given a flow, `Flow1`, the corresponding rules can be expressed in the forms:
  - `SA (Flow1.S, Flow1.R).M = [ALLOW|DENY|NULL]`
  - `PA (Flow1.S.P, Flow1.R.P).M = [ALLOW|DENY|NULL]`

  As with flows, the subject and the subject's partition are the first arguments of SA and PA (respectively), and they are also the row (as opposed to column) indexes in the corresponding matrices in Table 2-2.
- The following are equivalent expressions for explicit denial of `flow1` in SA:
  - `SA (Flow1.S, Flow1.R).M = DENY`
  - `!Flow1 ∈ SA`

66 Using this notation, the security policy stated in FDP_IFF.1.2-NIAP-0407 is:

- An operation shall be allowed in the Partition Abstraction *only if* for all of its `m` flows:
  - $PA\ (Flow_n.S.P,\ Flow_n.R.P).M = ALLOW$

  where `n = 1..m`
- An operation shall be allowed in the Least Privilege Abstraction *only if* for all of its `m` flows either:
  1. $SA(Flow_n.S,\ Flow_n.R).M = ALLOW$

     $(equivalently,\ Flow_n \in SA)$

     - or -

  2. both:

     a. $PA(Flow_n.S.P,\ Flow_n.R.P).M = ALLOW)$

     - and -

     b. $SA(Flow_n.S,\ Flow_n.R).M = NULL$

  where `n = 1..m`

67 The partition-pair and subject-exported resource authorizations do not have to be consistent. However, the TOE must be configured minimally with respect to both the authorized partition flows and the authorized subject-exported resource flows, per PoLP.

Table 2-2. Reference Access Matrix Representation for Least Privilege Abstraction

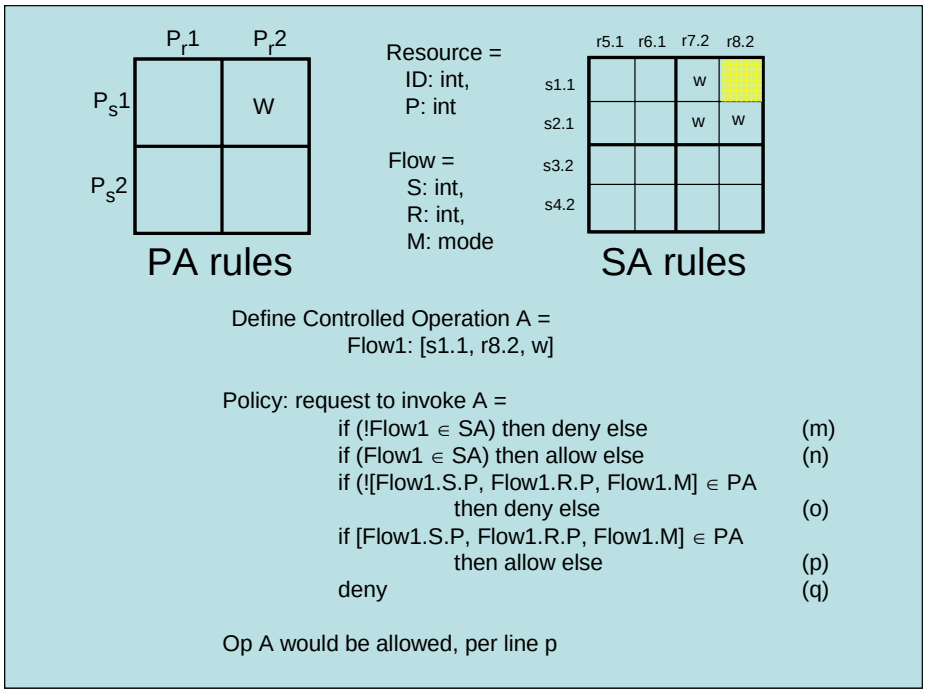


Table 2-2a. Example of TOE Implementing Explicit DENY SA Rule for Least Privilege Abstraction

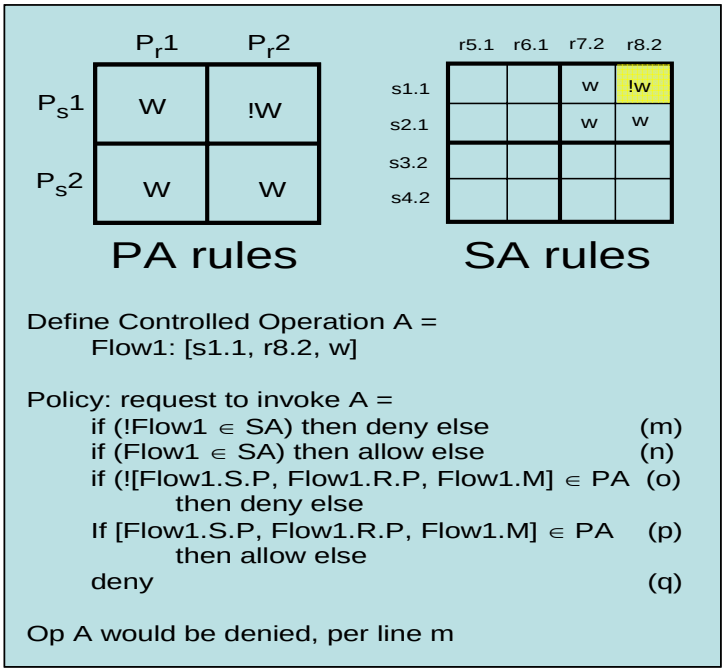


Table 2-2b. Example of TOE Implementing Explicit ALLOW PA Rule with Don't-Care SA Rule for Least Privilege Abstraction

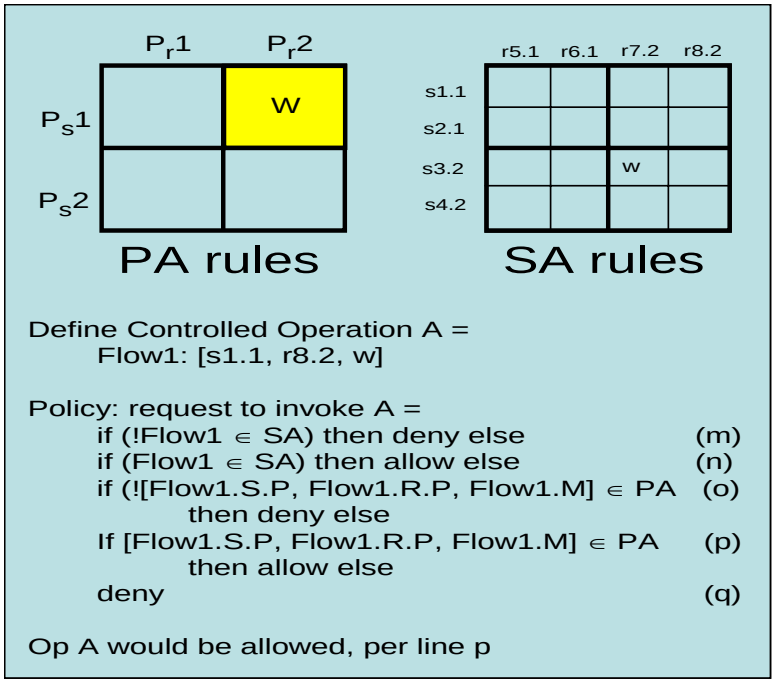


Table 2-2c. Example of TOE Implementing Explicit ALLOW SA Rule with Implicit DENY PA Rule for Least Privilege Abstraction

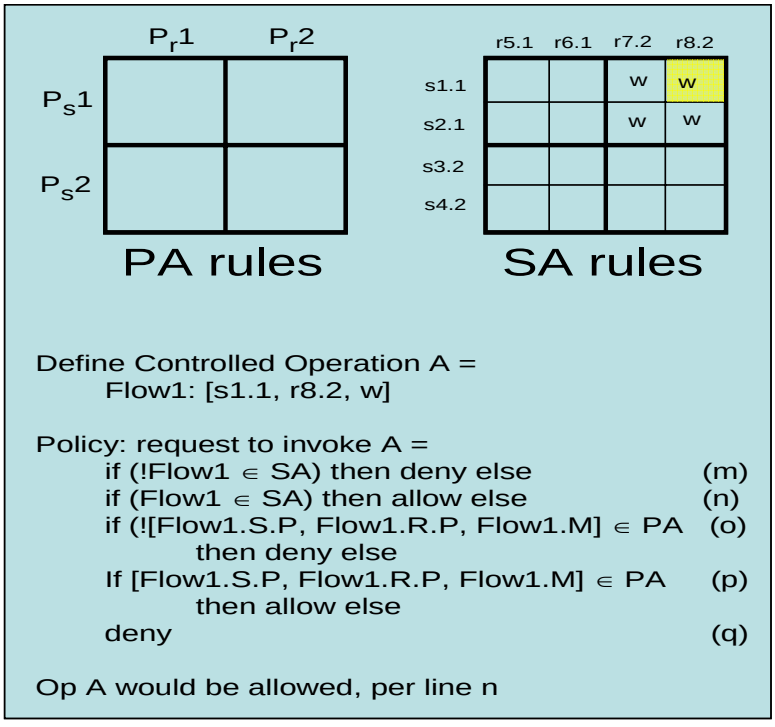


Table 2-2d. Example of TOE Implementing Default DENY Rule with Don't Care SA Rule for Least Privilege Abstraction

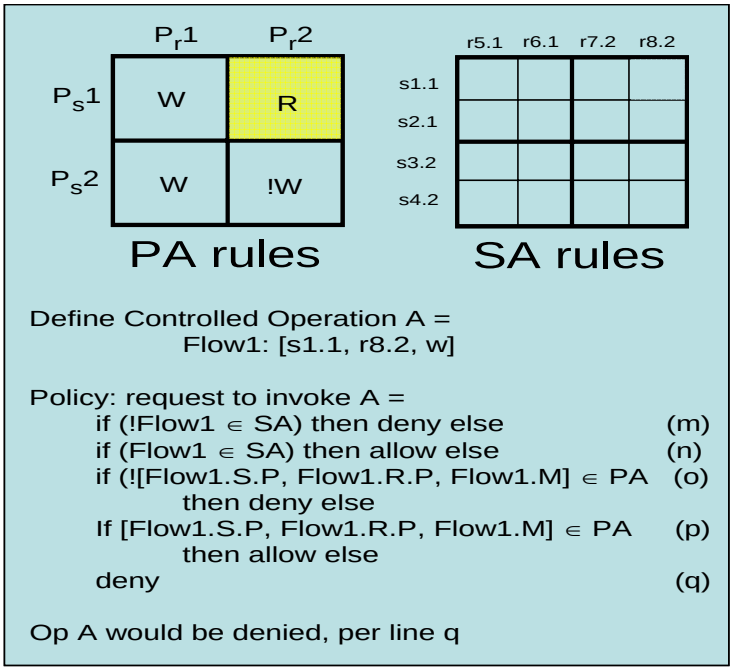
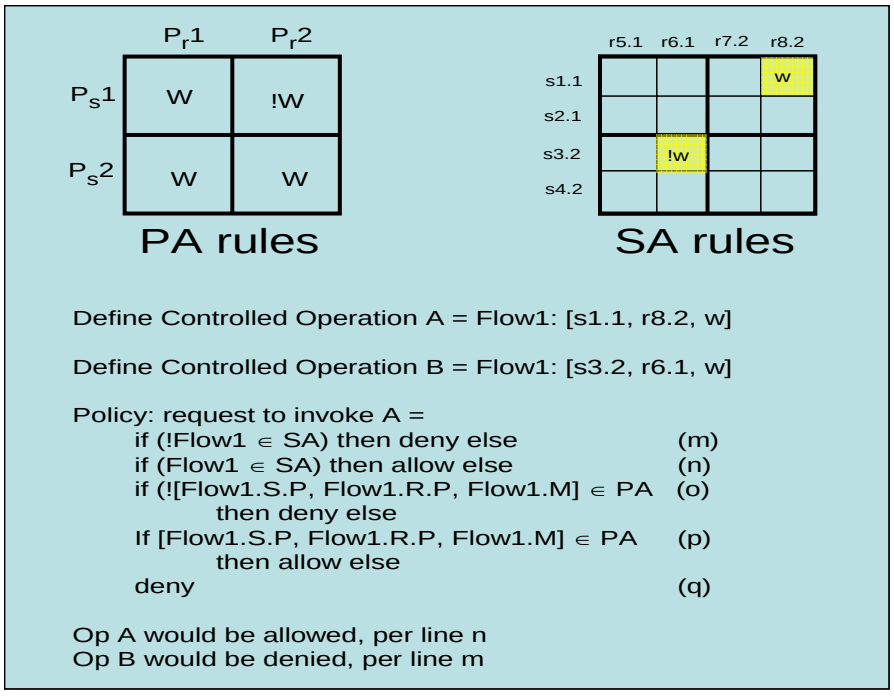


Table 2-2e. Example of TOE Implementing SA Over-rides PA Rules for Least Privilege Abstraction



2.3.3 Partitions and Subject Address Spaces

68 The TOE may manage memory and make portions of it accessible (viz., "addressable") at the TSF interface in a variety of ways, including (controlled) access to physical memory, and the use of various virtual memory techniques such as paging and segmentation. The TOE must assign distinct portions of memory to each partition, forming non-overlapping partition address spaces. The TOE must also assign address spaces to subjects, which may overlap (e.g., memory shared between subjects). However, while a subject is an abstract resource that has been bound to exactly one partition, a subject's address space – those resources to which the subject can refer – may include memory from different partition address spaces. The configuration data defines the partition and subject address spaces.

69 Table 2-3 shows an example of the allocation of physical memory in a TOE instantiation with 4GB of memory that is not virtualized. The memory regions of each partition are exported (i.e., made available to subjects by the kernel) in smaller units, called *blocks*. The blocks are delineated as offset:size relative to the start of their region, as measured in kilobytes. Partition 1 has one subject, *m*, and Partition 2 has one subject, *n*. Table 2-4 shows that the subjects in Partition 1 are allowed to read resources from Partition 2. Table 2-5 shows that Subject *m* has an address space that includes blocks from both Partition 1 and Partition 2, and both subjects have heterogeneous flow assignments, indicative of the Least Privilege abstraction. Another example of subjects with address spaces that cross partition boundaries is shown in Figure 2-3 (Subject 2 and Subject 3).

70 Note that asynchronous devices, such as a DMA device, can be modeled as subjects to account for their spontaneous actions, although their actions must still be bound by the PIFP.

Table 2-3. Partition Address Spaces and Subject Bindings

	Subjects	**Memory Regions**	**Blocks**
Kernel		0 to .5 GB	
Partition 1	*m*	1.5 to 3 GB	A = 0:64 B = 512:64
Partition 2	*n*	3 to 4 GB	C = 1:100 D = 512:128

Table 2-4. Partition Flow Table

Subject \ **Resource**	**Partition 1**	**Partition 2**
Partition 1	RW	R
Partition 2		RW

Table 2-5. Subject-resource Flow Table

	Resources (Memory Blocks)			
	A	**B**	**C**	**D**
Subject *m*	RW	R	R	R
Subject *n*			R	RW

2.3.4 TOE Configuration Changes

71 The entire PIFP can be changed offline – i.e., statically – by making available to the TOE a different configuration vector or vector set (see Figure 2-2). In support of system-level requirements (e.g., in response to operation needs or for reliability and availability), it may be necessary to change the PIFP during an *execution session*. This protection profile does not mandate that all TOE instances provide an online – i.e., dynamic – configuration change capability, but if such functionality is available, it may be a *total* or a *selective* change, as described below. In these descriptions, the most salient differences are underlined.

2.3.4.1 Static Total Configuration Change

72 The TOE is initialized with one configuration vector, which is used to define the TSF *configuration data*.

73 This change requires that the TOE be halted (e.g., via external power loss or an authorized subject). Then, before re-initialization of the TOE, an authorized administrator or mechanism outside of the TOE loads or specifies a different configuration vector, which will become the configuration data as a result of the next TSF initialization.

74 The assurance issue associated with this configuration change capability is to ensure that each configuration vector used reflects the organization's intent for TOE behaviour and policy enforcement.

2.3.4.2 Dynamic Total Configuration Change

75 This change includes the following capabilities and assurance issues, over and above those of the Static Total Configuration Change, as illustrated in Figure 2-4:

76 The TOE is initialized to contain a pre-specified configuration vector set (CVS). The TSF provides the capabilities for an *authorized subject* to (1) designate a "next" configuration vector (see *j'* in Figure 2-4 indicating where the next configuration indicator *j* is updated), and (2) change the configuration data from its current values to those defined by the "next" vector. This change can occur through either a TSF initialization (e.g., *restart*) or an online reconfiguration operation, as shown in Figure 2-4.

77 The SKPP does not require that the selection of the next configuration vector and the *activation* of that vector occur together as an atomic action. Nor is it required that activating the next configuration data includes halting or restarting the TOE – so long as secure state is maintained

before, during and after the change.

78 The additional assurance issue associated with this configuration change capability is to ensure that only an authorized subject may request the configuration change, that the TOE properly executes the change request, and that secure state is maintained before, during and after the change, especially for the case of the online activation operation.

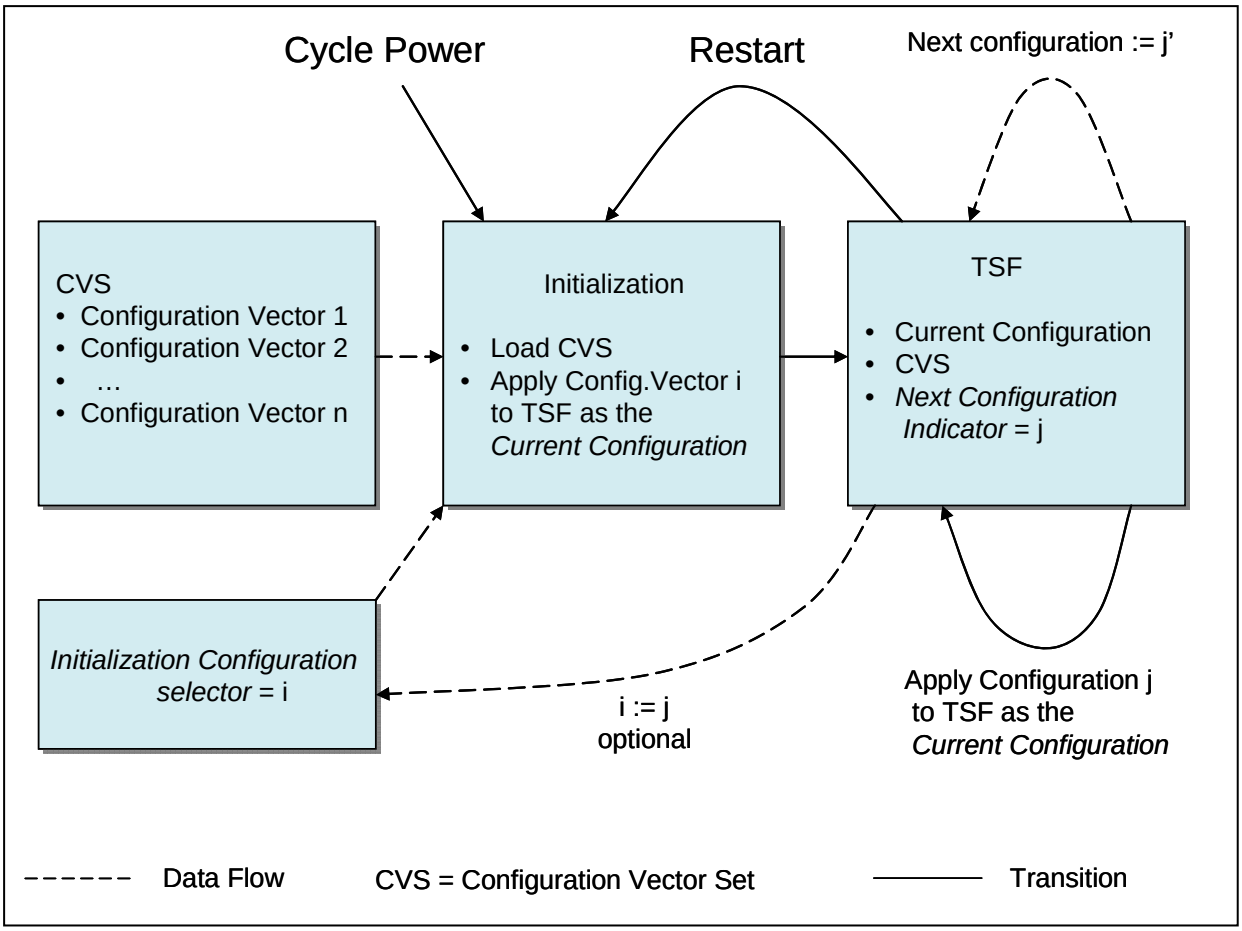


Figure 2-4. TOE Configuration Change

2.3.4.3 Dynamic Selective Configuration Change

79 There are two forms of dynamic selective change allowed. Whereas the previous change capabilities require the selection of a configuration vector that predefined all aspects of the TOE configuration, the selective change capabilities enable changes to individual configuration data items. As with the Dynamic Total Configuration Change, these capabilities also require an authorized subject to interact with the TSF to specify and initiate changes.

80 The two forms of selective change are *constrained* and *unconstrained*. In selecting either of these options, the TOE developer incurs the responsibility to document any additional security requirements in the Security Target such that the Common Criteria Part III, Class ASE Security Target evaluation criteria are met. Furthermore, the TOE developer must provide the rationale to

demonstrate that the added functional requirements continue to meet the security objectives of this profile, and are complete and consistent with the remaining functional requirements of this profile.

81 The functional properties and assurance issues associated with selective configuration change are as follows:

- **Constrained Selective Change**

  The TSF provides the capability for an authorized subject to perform runtime changes to individual configuration data items, as selected by the ST author, within explicit constraints that are enforced by the TSF. Constraints may be mandatory across all authorized changes, as well as specific to a single item.

  Such changes may include creating and destroying partitions and the creation or destruction of information flows between partitions, and between subjects and exported resources.

  The change constraints limit the type of policy "transitions" (e.g., prohibiting ad hoc changes to a resource's assigned partition) and new PIFP definitions (e.g., prohibiting configurations that would allow information flows between certain "types" of partitions, subjects and exported resources) that are achievable through selective configuration changes. The change constraints can be defined in the configuration data or in the TSF itself.

  The additional assurance issue associated with this capability is to ensure that ad hoc policy change requests and change constraints are consistent with the organization's policy intents, including those for partitioning and information flow. Additionally, the authorized subject that performs these changes must be trusted with respect to the policy allocated to the TSF, since it shares in the determination of how policy is to be enforced.

- **Unconstrained Selective Change**

  The Unconstrained Selective Change is identical to the capabilities and assurance issues of the Constrained Selective Change, except that:

  There are no constraints on changes to the selected configuration data items.

  The additional assurance issue associated with this capability is that, without any change constraints it may be difficult for the TOE vendor to provide a convincing definition of "secure transition" in the PIFP model[3].

2.4 Modes, States, and Trusted Recovery

82 While in an execution session the TOE is in either operational mode or maintenance mode, and simultaneously, is in either a secure state or an insecure state. A normal successful initialization brings the TOE to a secure state, in operational mode (see "O\S" in Table 2-6). For the failures

[3] The difficulty of analyzing the safety of arbitrary policy changes was proven to be mathematically "undecidable" by Harrison, Russo and Ullman. [7]

and conditions specified by FPT_FLS.1,the TOE must remain in a secure state (i.e., "O\S, "M\S" or "H\S"). Other failures and conditions could cause the TOE to temporarily enter an operational insecure state ("O\I"). That is, between the time that a security failure first occurs and the time that the TSF can detect it and respond, the conservative assumption is that the failure introduces insecurity. This insecure state is ephemeral because the TOE must return to a secure state by: (1) remaining in operational mode, e.g., if the failure is directly recoverable, (2) transitioning to maintenance mode if the failure can be repaired there ("M\I"), or (3) halting ("H\S").

Table 2-6. Possible Mode/State Combination

	STATE / MODE	Secure (S)	Insecure (I)
Execution Session	Operational (O)	O\S	O\I
	Maintenance (M)	M\S	M\I
Halted (H)		H\S	n/a

83 The purpose of maintenance mode is to provide a security environment in which to re-establish a secure state or to ensure the ability of the TOE to continue to maintain a secure state. In maintenance mode, the TOE must continue to ensure that no actions occur that would violate the TSP. This may involve the disabling of subjects or TOE functions, and may include the repair of damaged internal data structures.

84 When halted, the TOE is considered to be in a secure state, since no subject actions are possible. However, if the TOE halted as a result of being in an insecure state, then any inconsistency between the insecure and secure state must be resolved prior to the resumption of the TOE in the operational mode. This could be accomplished in several ways, such as: halting, offline maintenance, initialization into a new TOE configuration, initialization into maintenance mode, or combinations of these. Figure 2-5 illustrates the possible transitions between "halted" and the two modes of execution.

85 This protection profile allows the transition to maintenance mode to occur by way of halting the system and restarting with a suitable configuration vector. Also, it does not require interaction with an authorized administrator or other trusted individual to return from maintenance mode to secure operational mode, as some other protection profiles might (e.g., see [2], paragraph 1245). The implementation of maintenance mode is ST-specific, and its properties must be captured in the Security Policy Model.

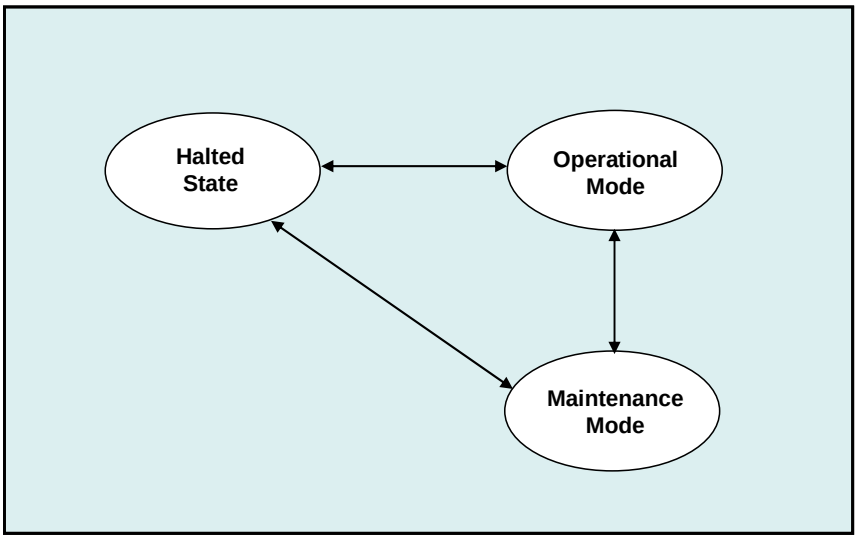


Figure 2-5. TOE Transition Diagram

2.5 Trusted Delivery

86 The components of the TOE may be delivered to the customer environment in various ways, both for the initial delivery and for subsequent updates. This protection profile requires the TOE to include trusted delivery procedures and/or functions to verify that the on-site version of the TOE matches the master version (see "Trusted Delivery" in Figure 2-6). Such a verification function may be configured to execute on the TSF hardware or on other hardware, but in either case the function and the hardware that it runs on are evaluated as part of the TOE, just as with the initialization and configuration functions. Data integrity validation of the TOE and related configuration vectors occurs again as part of initialization. Figure 2-6 shows how applications and TOE configuration vectors may be installed by the TOE developer and/or by various entities within the customer environment. If TOE components were modified after trusted delivery, then the TOE would not be in an evaluated configuration.

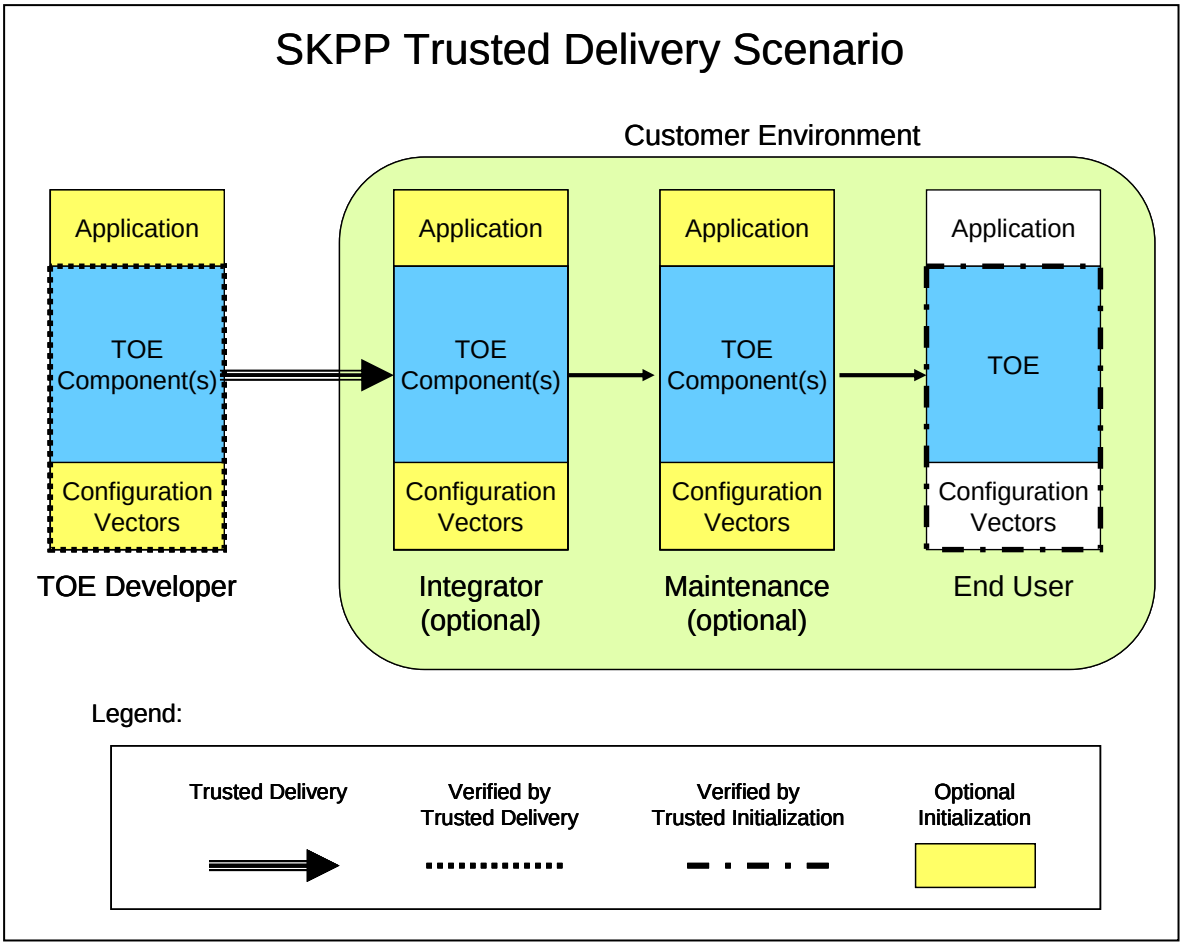


Figure 2-6. Trusted Delivery Scenario

2.6 Platform Considerations

87 For a high assurance system, the platform, which may consist of both hardware and firmware, is considered to be an integral part of the TOE and is subject to certain evaluation requirements. Appendix G provides a rationale for platform assurance requirements.

2.6.1 Platform Components

88 The hardware platform for a TOE is assembled from one or more instances of one or more types of platform components. It is up to the ST author to define the components that make up the platform. The intent is that a platform component should be an entity that can be procured independently—either from the TOE developer or from other commercial sources—and assembled in combination with other platform components to make up the entire platform.

89 For example, a very restrictive platform definition could say that there is just one type of component (the complete platform), and that only one explicitly identified instance of that component is acceptable in the platform definition. This restrictive approach could simplify evaluation and documentation, but at the expense of limiting the scope of the TOE's evaluation

to only one particular piece of hardware. Such a limitation might be acceptable in the case of a TOE that is intimately tied to a particular piece of purpose-built hardware, but is less acceptable when the TOE is intended to run on commodity, general-purpose hardware platforms.

90 A less restrictive platform definition could still define a single component type, but allow variability in that component's characteristics. For example, a hardware platform could be defined as a particular model of server computer, but allow variability in the processor speed, amount of memory, number of network interfaces, etc. This is still a simple approach for evaluation purposes, as the platform definition need only accommodate variations in characteristics, not interfaces between components.

91 A more open platform definition could define several component types that can be assembled (in accordance with defined rules) to provide the TOE's hardware platform. For example, a product intended for general-purpose platforms could define component types for “computer”, “disk storage”, “network interface”, and so forth, and then give rules for how those components may be assembled.

92 A platform component can be defined either by explicit identification or by specification. Explicit identification (i.e., by a manufacturer's model designation) is simpler for evaluation and documentation purposes, but may undesirable because that particular model may cease to be available at some point in the future.

93 In contrast, definition of a platform component by specification accommodates hardware evolution, but at the cost of a more complex evaluation process and assessment effort by the end-user customer (who must assess a potential platform component against its specification). The evaluation sponsor may choose to facilitate this activity by performing such assessments and making the results available to end-users.

2.6.2 Platform Interfaces

94 A platform consists of one or more components: C1, C2, … Cn. Figure 2-7 shows platform components and their interfaces. Each platform component presents zero or more internal inter-component interfaces that are restricted to the platform itself, and zero or more programming or I/O interfaces which are accessible by the TOE and applications running on the TOE.

95 Platform components, in particular the CPU, may support a notion of privilege by presenting minimally a privileged (kernel) mode and an unprivileged (user) mode. Some platform components, i.e. CPUs, support more granularity of privilege in the unprivileged modes (e.g. rings). All privileged platform interfaces are internal to the TOE with access to those interfaces, in general, restricted to the TSF. The TSF may choose to virtualize and export certain privileged platform interfaces, while reserving the remainder for its exclusive use.[4] Unprivileged platform interfaces are accessible to both the TOE and its applications.

[4] Normally, the system designer cannot control which platform interfaces are privileged; however, for certain processors, the microcode can be modified so that instructions that are ordinary unprivileged can be turned into privileged ones.

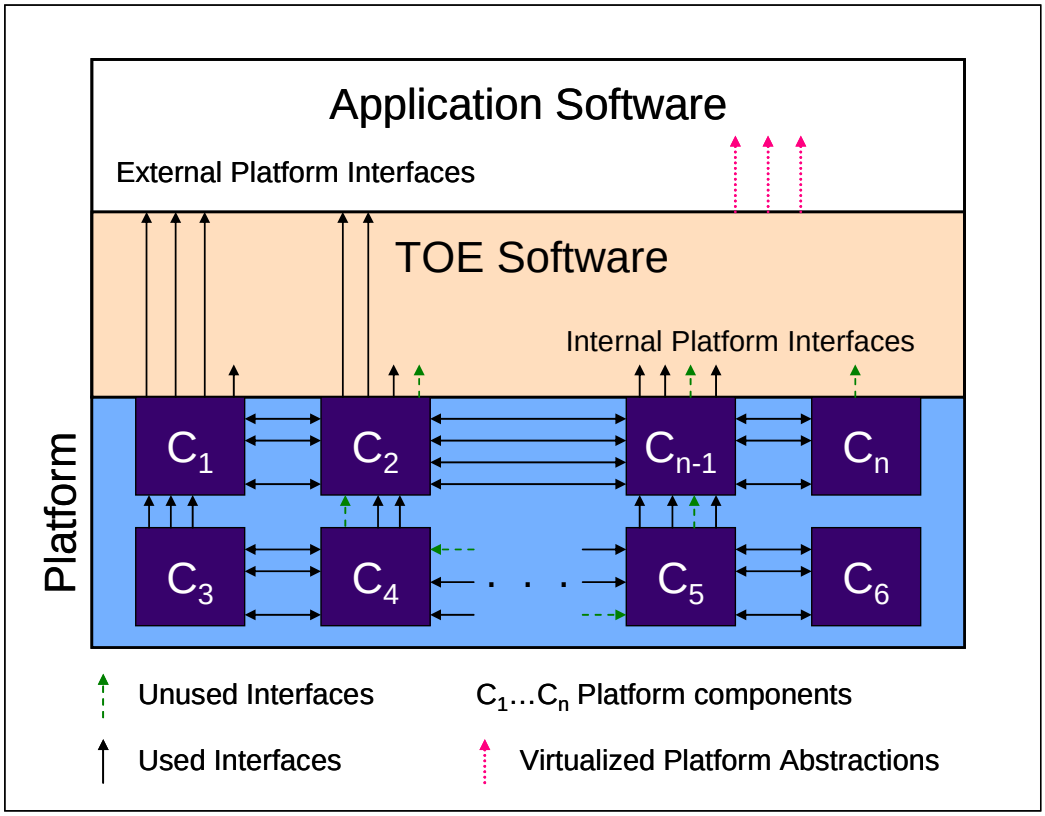


Figure 2-7. Platform Components and Their Relationships

96 Because external platform interfaces are accessible to untrusted entities, emphasis is placed on ensuring their correct behavior. An untrusted entity may, for example, attempt to use such interfaces in ways that violate the normal assumptions and rules for correct use. Internal platform interfaces are accessible only within the TSF, which is designed to use them in correct and well-defined ways.

97 The TSF might not utilize all possible platform component interfaces presented to it, leaving some platform component interfaces unused. Similarly, some inter-component interfaces might not be used. The TSF may also virtualize selected internal platform interfaces and present those virtualized abstractions at its interface. These abstractions, as well as the external platform interfaces, are examples of the TOE's exported resources.

2.7 Evaluation Considerations

2.7.1 Security Management

98 This protection profile does not include administrative roles, identification & authentication, or security management functions associated with individuals who acquire administrative roles. It is expected the TOE security management decisions and actions are performed offline by

authorized administrators or during an execution session by subjects granted the authorization to do so. If the TOE requires a user interface for authorized administrators to perform security management, then the appropriate requirements must be added to the Security Target such that they are consistent with the objectives of this protection profile.

2.7.2 TOE Component Development Diversity

99 The TOE may include multiple hardware or software components that have been created by different developers. While this diversity is not a conceptual problem for evaluation, it may present constraints on the execution of the evaluation process, such as the order of the evaluation of components. Regardless, the various components for a given TOE must be evaluated to the requirements of this protection profile, based upon the role they serve while enforcing or supporting the enforcement of the PIFP, supporting the generation of configuration vectors, trusted distribution, or secure initialization of the TOE.

100 Similarly, the modular and component structure of each evaluated TOE may differ significantly. The TOE may consist of separate initialization and runtime components, as well as separate hardware independent and hardware dependent components[5]. This structure will be a critical factor in the TOE evaluation, especially with regard to whether a module or component is determined to be in the TSF, the TOE, or in the IT environment, since different requirements will apply. For example, initialization components are not part of the TSF but, as part of the TOE, are subject to evaluation scrutiny (see Figure 2-1). The SKPP takes a standard approach to address different or unique requirements for different TOE configurations, as follows:

- When possible, differences in the criteria are first addressed through use of the CC-defined operations of assignment, selection, iteration and refinement.
- In the cases where the CC-defined operations do not suffice, the CC-defined explicitly stated requirements model is used.
- Where there are differences in the implications of the criteria, those differences are addressed by the Application Notes that follow the criteria.

101 When the TOE is used in composition with other components or products to make up a larger system, it is the responsibility of the larger system's designers to articulate support for a coherent application-level security policy in the TOE configuration data, as well as to ensure that the configuration data itself is self-consistent. It is only with well-formed configuration data that the TOE can be expected to enforce mission-critical policies. The judgment as to whether a given instantiation of configuration data is self-consistent, or well formed with respect to the intended application-level security policy is beyond the scope of this protection profile and beyond the scope of the evaluation of the TOE.

[5] Examples of hardware dependent components are an "architecture support package" (ASP) for interaction with a specific processor and a "board support package" (BSP) for interaction with a specific processor environment (devices, buses, I/O, etc.).

2.8 Use of High Robustness

102 A high robustness TOE is necessary protection for environments where the presence of both sophisticated threat agents and high value resources makes the likelihood of an attempted compromise high. An alternative perspective is to consider the damage to the organization that would result if a TOE compromise were to occur. "Likelihood of compromise" and "damage resulting from compromise" are parallel notions. They both are intrinsically linked to the value of the data being processed – the more valuable the data, the greater the likelihood that an adversary will attempt to compromise the TOE, similarly the greater the damage to the organization that would result from such compromise.

3. TOE Security Environment

103 This section defines the expected TOE security environment in terms of the threats, security assumptions, and the security policies that must be followed for the high robustness TOE.

3.1 Threats

104 The following threats are addressed by PP compliant TOEs:

T.ADMIN_ERROR	An administrator may incorrectly install or configure the TOE (including the misapplication of the protections afforded by the PIFP), or install a corrupted TOE resulting in ineffective security mechanisms.
T.ALTERED_DELIVERY	The TOE may be corrupted or otherwise modified during delivery such that the on-site version does not match the master distribution version.
T.CONFIGURATION_CHANGE	The lack of TSF-enforced constraints on the ability of an authorized subject to invoke or dictate how the TOE is reconfigured may result in the TOE transitioning to an insecure (unknown, inconsistent, etc) state.
T.CONFIGURATION_INTEGRITY	The TOE may be placed in a configuration that is not consistent with that of the configuration vector due to the improper loading of the configuration vector or incorrect use of the configuration vector during TOE initialization.
T.COVERT_CHANNEL_EXPLOIT	An unauthorized information flow may occur between partitions as a result of covert channel exploitation.
T.DENIAL_OF_SERVICE	A malicious subject may block others from system resources (e.g., system memory, persistent storage, and processing time) via a resource exhaustion attack.
T.INCORRECT_CONFIG	The configuration vectors are not an accurate and complete description of the operational configuration of the TOE as used by an organization.
T.INCORRECT_LOAD	The software portion of the TSF implementation and/or configuration vectors are not correctly converted into a TOE-useable form.

T.INSECURE_STATE	The TOE may be placed in an insecure state as a result of an erroneous initialization, halt, reconfiguration or restart, transition to maintenance mode, or as a result of an unsuccessful recovery from a system failure or discontinuity.
T.LEAST_PRIVILEGE	The design and implementation of the TSF internals may not suffice to limit the damage resulting from accident, error or unauthorized use.
T.POOR_DESIGN	Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious subject.
T.POOR_IMPLEMENTATION	Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious subject.
T.POOR_TEST	Lack of or insufficient evaluation and runtime tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered.
T.TSF_COMPROMISE	A malicious subject may cause TSF data or executable code to be inappropriately accessed (viewed, modified, executed, or deleted).
T.UNAUTHORIZED_ACCESS	A subject may gain access to resources or TOE security management functions for which it is not authorized according to the TOE security policy.

3.2 Security Policy

122 The following organizational security policies are addressed by PP compliant TOEs:

P.ACCOUNTABILITY	The TOE shall provide the capability to make available information regarding the occurrence of security relevant events.
P.CONFIGURATION_CHANGE	The TOE shall support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.
P.CRYPTOGRAPHY	The TOE shall use NSA approved cryptographic mechanisms.

P.INDEPENDENT_TESTING	The TOE shall undergo independent testing.
P.RATINGS_MAINTENANCE	A plan for procedures and processes to maintain the TOE's rating shall be in place to maintain the TOE's rating once it is evaluated.
P.SYSTEM_INTEGRITY	The TOE shall provide the ability to periodically validate its correct operation.
P.USER_GUIDANCE	The TOE shall provide documentation regarding the correct use of the TOE security features.
P.VULNERABILITY_ANALYSIS_AND_TEST	The TOE shall undergo independent vulnerability analysis and penetration testing by NSA to demonstrate that the TOE is resistant to an attacker possessing a high attack potential.

3.3 Security Usage Assumptions

123 The specific conditions below are assumed to exist in a PP-compliant TOE environment:

A.PHYSICAL	It is assumed that the non-IT environment provides the TOE with appropriate physical security commensurate with the value of the IT assets protected by the TOE.
A.SUBJECT_ALLOCATION	It is assumed that a properly trained trusted individual will create configuration vectors such that, for those partitions to which subjects are allocated, each partition is allocated one or more subjects (i.e., subjects with homogeneous access requirements, or subjects with heterogeneous access requirements) that are appropriate for the policy abstraction supported by the TOE.
A.COVERT_CHANNELS	If the TOE has covert storage and/or timing channels, then for all subjects executing on that TOE, it is assumed that relative to the IT assets to which they have access, those subjects will have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.
A.TRUSTED_FLOWS	For any subject configured to have unrestricted access in multiple policy equivalence classes, it is assumed that the subject is trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.[6]

[6] The TOE is allowed to be configured with multiple partitions representing a single policy equivalence class, and

A.TRUSTED_INDIVIDUAL	It is assumed that any individual allowed to perform procedures upon which the security of the TOE may depend is trusted with assurance commensurate with the value of the IT assets.

the resources in such a group of partitions would be treated equivalently with respect to the Partition Flow Rule of the PIFP. For example, it might be desirable in a larger system that is built on an SKPP TOE for multiple TOE partitions to be interpreted as "SECRET" in the application domain. To support this, the TOE configuration data could be created to allow both read and write between each of those partitions. Refer to Section 7 for further discussion of rationale for this assumption.

4. Security Objectives

124 This section defines the security objectives for the TOE and its environment. These objectives are suitable to counter all identified threats and cover all identified organizational security policies and assumptions. The security objectives allocated to the TOE are identified with "O." preceding the name of the objective. The security objectives allocated to the environment are identified with "OE." preceding the name of the objective.

4.1 TOE Security Objectives

O.ACCESS	The TOE will ensure that subjects gain only authorized access to exported resources.
O.ADMIN_GUIDANCE	The TOE will provide administrators with the necessary information for secure management of the TOE.
O.AUDIT_GENERATION	The TOE will provide the capability to detect, generate and export audit records for security relevant auditable events.
O.AUTHORIZED_SUBJECT	The TOE will ensure that only authorized subjects are allowed to access restricted resources.
O.BOUNDED_EXECUTION	The TOE will exhibit predictable and worst-case bounded execution behavior.
O.CHANGE_MANAGEMENT	The configuration of, and all changes to, the configuration items that comprise the TOE and its development evidence will be analyzed, tracked, and controlled by trusted individuals throughout the TOE's development.
O.CONFIGURATION_CHANGE	The TOE will support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.
O.CORRECT_CONFIG	The TOE will provide procedures and mechanisms to generate the configuration vectors such that they accurately describe the operational configuration of the TOE as used by an organization.

O.CORRECT_INIT	The TOE will provide mechanisms to correctly transfer the software portion of the TSF implementation and TSF data into the TSF's security domain and to correctly establish the TOE in an operational configuration consistent with the configuration vector that defines the configuration data.
O.CORRECT_LOAD	The TOE will provide procedures and mechanisms to correctly convert the software portion of the TSF implementation and/or configuration vectors into a TOE-useable form.
O.CORRECT_TSF_OPERATION	The TOE will provide a runtime self-test capability. The TOE will provide the means for an authorized subject to invoke and obtain the results of the self-test. The TOE will take action in response to any failure of a runtime self-test capability.
O.COVERT_CHANNEL_ANALYSIS	The TOE will undergo appropriate covert channel analysis by NSA to demonstrate that the TOE satisfies covert channel mitigation metrics.
O.CRYPTOGRAPHY	The TOE will use NIST FIPS-validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, handling, and storage of keys) and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).
O.FUNCTIONAL_TESTING	The TOE will undergo independent security functional testing that demonstrates the TSF satisfies the security functional requirements.
O.INIT_SECURE_STATE	The TOE will provide mechanisms to transition the TSF to an initial secure state without protection compromise.
O.INSTALL_GUIDANCE	The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.
O.INTERNAL_LEAST_PRIVILEGE	The entire TSF will be structured to achieve the principle of least privilege among TSF modules.

O.MANAGE	The TOE will provide all the functions necessary to support the administrative users and authorized subjects in their management of the TOE security functions and configuration data, and restrict these functions from use by unauthorized subjects.
O.RATINGS_MAINTENANCE	Procedures and processes to maintain the TOE's rating will be documented.
O.RECOVERY_SECURE_STATE	The TOE will provide procedures and/or mechanisms, which can be used in the event of failure, faults, or discontinuity, to preserve secure state and to transition the TSF back to a secure state without protection compromise.
O.REFERENCE_MONITOR	The TOE will provide a reference validation mechanism responsible for the enforcement of the TSP. The reference validation mechanism will execute in its own security domain. The reference validation mechanism must be tamper proof, its enforcement functions must be always invoked, and its design and implementation must be of size and complexity small enough to be subject to analysis and tests, the completeness of which can be assured.
O.RESIDUAL_INFORMATION	The TOE will ensure that any information contained in a protected resource is not released to subjects when the resource is reallocated.
O.RESOURCE_ALLOCATION	The TOE will provide mechanisms that enforce constraints on the allocation of exported TOE resources.
O.SECURE_STATE	The TOE will preserve secure state during an execution session.
O.SOUND_DESIGN	The TOE will be designed using sound design principles and techniques which will be accurately documented. The TOE design will be completely and accurately documented.
O.SOUND_IMPLEMENTATION	The implementation of the TOE will be an accurate instantiation of its design.

O.SUBJECT_ISOLATION	The TOE will provide mechanisms to protect each subject from unauthorized interference by other subjects.
O.TRANSITION	The TOE will provide the capabilities for an authorized subject to restart the TOE, halt the TOE and transition the TOE into maintenance mode.
O.TRUSTED_DELIVERY	The integrity of the TOE must be protected during the initial delivery and subsequent updates, and verified to ensure that the on-site version matches the master distribution version.
O.TSF_INTEGRITY	The TOE will verify the integrity of the TSF code and data.
O.USER_GUIDANCE	The TOE will provide users with the necessary information for secure use of the TOE.
O.VULNERABILITY_ANALYSIS_TEST	The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.

4.2 Environment Security Objectives

OE.PHYSICAL	Physical security will be provided for the TOE by the non-IT environment commensurate with the value of the IT assets protected by the TOE.
OE.SUBJECT_ALLOCATION	A properly trained trusted individual will create configuration vectors such that, for those partitions to which subjects are allocated, each partition is allocated one or more subjects (i.e., subjects with homogeneous access requirements, or subjects with heterogeneous access requirements) that are appropriate for the policy abstraction supported by the TOE.

OE.COVERT_CHANNELS	If the TOE has covert storage and/or timing channels, then all subjects executing on that TOE will, relative to the IT assets to which they have access, have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.
OE.TRUSTED_FLOWS	For each configuration of the TOE, a partial order of the flows that are allowed between policy equivalence classes will be identified[7]. Any subject allowed by the configuration data to cause information flow that is contrary to the partial order will be trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.
OE.TRUSTED_INDIVIDUAL	Any individual allowed to perform procedures upon which the security of the TOE may depend must be trusted with assurance commensurate with the value of the IT assets.

[7] The partial ordering and equivalence class properties of a lattice flow policy are described by Denning [9].

5. TOE Security Functional Requirements

125 This section contains the requirements for the TOE security functions (TSF). The requirements are applied against the TOE in conjunction with the underlying hardware that supports it. The requirements contained in this section are either selected from Part 2 of the CC or are explicitly stated in accordance with the CC rules for explicitly stated requirements (refer to CC Part III APE_SRE). Table 5.1 lists the explicitly stated functional requirements in this section.

Table 5.1. Explicitly Stated Functional Requirements

Explicit Component	Component Behavior Name
FAU_SAR_EXP.1	Audit Review
FAU_SEL_EXP.1	Selective Audit
FIA_ATD_EXP.1	Partition, Subject and Exported Resource Attribute Definition
FIA_USB_EXP.1	Partition, Subject and Exported Resource Attribute Binding
FMT_MCD_EXP.1	Management of Configuration Data
FMT_MSA_EXP.1	Management of Security Attributes
FMT_MSA_EXP.3	Static Policy Attribute Initialization
FPT_CFG_EXP.1	Configuration Change
FPT_ESS_EXP.1	Establishment of Secure State
FPT_HLT_EXP.1	TOE Halt
FPT_MTN_EXP.1	TOE Maintenance
FPT_MTN_EXP.2	TOE Maintenance Secure
FPT_PLP_EXP.1	TSF Least Privilege
FPT_RCV_EXP.2	Automated Recovery
FPT_RST_EXP.1	TOE Restart
FPT_TST_EXP.1	TSF Testing
FRU_PRU_EXP.1	TSF Predictable Resource Utilization

5.1 Security Audit (FAU)

5.1.1 Security Audit Automatic Response (FAU_ARP)

5.1.1.1 Security Alarms (FAU_ARP.1)

FAU_ARP.1.1 **Refinement:** The TSF shall take [assignment: *list of the actions **to take***] upon detection of **any failure of the tests defined in FPT_AMT.1 and FPT_TST.1**.1

Application Note: The TSF must take some action. The ST author is to fill in the open assignment with the list of actions that are applicable for the TOE's intended use, with particular attention to providing the ability for the TOE to support system-level requirements for fault/failure detection and response. Acceptable actions include a means to notify the IT environment or explicit action taken by the TSF (e.g., shutdown, reconfiguration).

5.1.2 Security Audit Data Generation (FAU_GEN)

5.1.2.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1-NIAP-0407 The TSF shall be able to generate an audit record of the following auditable events:

a) Start-up and shutdown of the audit functions;

b) All auditable events for the basic level of audit;

c) **All auditable events listed in Table 5.2; and**

d) [selection: [assignment: ***all auditable events at a basic level of audit introduced by the inclusion of additional SFRs determined by the ST author***], [assignment: ***all auditable events commensurate with a basic level of audit introduced by the inclusion of explicit requirements determined by the ST author***], ***"no additional events"***]

Application Note: For the selection, the ST author should choose one or both of the assignments (as detailed in the following paragraphs), or select "no additional events".

For the first assignment in the selection, the ST author augments the table (or lists explicitly) the audit events associated with the basic level of audit for any SFRs that the ST author includes in the ST that are not included in this PP.

Likewise, for the second assignment the ST author includes audit events that may arise due to the inclusion of any explicit requirements in the ST that are not already in the PP. Because "basic" audit is not defined for such requirements, the ST author will need to determine a set of events that are commensurate with the type of information that is captured at the basic level for similar requirements.

If no additional (CC or explicit) SFRs are included, or if additional SFRs are included that do not have "basic" audit associated with them, then it is acceptable to assign "no additional events" in this item.

In determining whether or not added functionality should have auditable events, the ST author is to assess the added functionality in terms of its conceptual relationship with the core functionality expressed in this PP and their corresponding requirements for auditable events. As an example: FAU_SEL_EXP.1 requires that the set of auditable events be statically determined prior to execution of the TSF and that the set of auditable events are not modifiable during runtime. Since there is no capability to modify the audit behavior at runtime, there is no requirement to audit changes to the runtime behavior. However, should the ST author provide the capability for authorized subjects to modify the behavior of the audit mechanism during runtime, then any such runtime modification constitutes an auditable event.

Application Note: The audit record structure is to be documented in the administrative guidance as required by AGD_ADM_EXP.1.14C. The TSF is expected to identify each auditable event and to capture data that characterizes each auditable event as defined in Table 5-2 Auditable Events. The TSF is not required to notify the IT environment of the existence of the audit data and the TSF is not required to "push" the information to the IT environment.

Application Note: It is common that for purposes of engineering analysis, embedded system components record operational and health and status data to support post-operation analysis and debugging. This data is referred to as instrumentation. This data is not necessarily security relevant, that is, not associated with enforcement of the security policy by the TSF. The audit data generation requirements in this PP should not be confused with instrumentation requirements levied by applications. This protection profile does not forbid integrating audit data with instrumentation data. However, if a single mechanism is used to manage both, then all collected data must be protected as security-relevant audit data per the requirements in this profile.

Table 5.2. Auditable Events

Security Functional Requirement	Audit events prompted by requirement
Security Alarms (FAU_ARP.1)	• Actions taken due to failure of TSF self tests and tests defined in FPT_AMT.1.1 *Application Note: TSF self tests include the suite of tests to determine the correct operation of the software portion of the TSF implementation and the TSF integrity tests for verification of TSF data and TSF executable code integrity.*
Audit Data Generation (FAU_GEN.1)	(None)
Explicit: Audit Review (FAU_SAR_EXP.1)	(None)
Explicit: Selective Audit (FAU_SEL_EXP.1)	(None)
Complete Information Flow Control (for Information Flow Control Policy) (FDP_IFC.2)	(None)
Simple Security Attributes (FDP_IFF.1)	• Denial of requested operation
Limited Illicit Information Flows (FDP_IFF.3)	• The use of identified illicit information flow channels
Explicit: Full Residual Information Protection (FDP_RIP.2)	(None)
Explicit: Partition, Subject and Exported Resource Attribute Definition (FDP_ATD_EXP.1)	(None)
Explicit: User-Subject Binding (FIA_USB_EXP.1 (1), (2), (3))	• Unsuccessful binding of security attributes to individual partitions, subjects, non-subject exported resources
Explicit: Management of Configuration Data (FMT_MCD_EXP.1)	(None)
Management of Security Functions (FMT_MOF.1)	(None)

Explicit: Management of Security Attributes (FMT_MSA_EXP.1)	(None)
Static Policy Attribute Initialization (FMT_MSA.3)	• Any TSF assignment of a restrictive default value
Management of TSF Data (FMT_MTD.1)	(None)
Secure TSF Data (FMT_MTD.3)	• Rejection of specified values for TSF data
Specification of Management Functions (FMT_SMF.1)	(None)
Underlying Abstract Machine Test (FPT_AMT.1)	• Failures detected by tests of the underlying abstract machine and the results of the tests
Explicit: Configuration Change (FPT_CHG_EXP.1)	• All requests for a configuration change
Explicit: Establishment of Secure State (FPT_ESS_EXP.1)	• Startup of the TOE, i.e., successful and unsuccessful establishment of secure state
Failure with Preservation of Secure State (FPT_FLS.1)	• Failures detected by the FPT_AMT.1 and FMT_TST.1 tests • Other TSF failures specified in the assignment statement of FPT_FLS.1.1b
Explicit: TOE Halt (FPT_HLT_EXP.1)	(None)
Explicit: TOE Maintenance (FPT_MTN_EXP.1)	• Halt of the TOE when the TSF is unable to preserve secure state after transitioning to maintenance mode from a secure state
Explicit: TOE Maintenance Secure (FPT_MTN_EXP.2)	(None)
Explicit: TSF Least Privilege (FPT_PLP_EXP.1)	(None)
Explicit: Automated Recovery (FPT_RCV_EXP.2)	• TOE condition that causes the TSF to be in an insecure state • Action taken to attempt to recover the TOE to a secure state
Function Recovery (FPT_RCV.4)	• The inability of the TOE to return to a secure state after failure of a security function • The detection of a failure of a security function
Explicit: TOE Restart (FPT_RST_EXP.1)	(None)
Non-Bypassability of the TSF (FPT_RVM.1)	(None)
Complete Reference Monitor (FPT_SEP.3)	(None)
Reliable Time Stamp (FPT_STM.1)	• Changes to the TSF-internal time source

Explicit: TSF Testing (FPT_TST_EXP.1)	• Failures of TSF self tests and the results of the tests
Explicit: Minimum and Maximum Quotas (FRU_RSA.2)	• Attempt to exceed memory quota • Attempt to exceed processing time quota
Explicit: TSF Predictable Resource Utilization (FRU_PRU_EXP.1)	(None)

Application Note: The use of the word "None" in Table 5-2 means that there are no requirements for auditing events associated with the functions/mechanisms that implement the stated Security Functional Requirement.

FAU_GEN.1.2-NIAP-0407 The TSF shall record within each audit record at least the following information:

a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST,

- **the identity of the resource;**
- **for changes that affect the PIFP attributes, the new and old values of the PIFP attributes specified at the TSFI.**

Application Note: It is acceptable for the TSF to provide a timestamp that reflects the date and time of the event as a relative time within the TOE. This is acceptable so long as the IT environment is able to correlate that timestamp to date and time of day and the IT environment is able to establish event sequences based upon timestamp values (that is, the granularity to which a precise ordering of events can be determined is a function of the precision of the underlying clock which in turn establishes the time-stamp format). The administrative documentation is required to discuss the structure, precision, and interpretation of the timestamp (ref AGD_ADM_EXP.1.14C).

Application Note: Audit information associated with security functions that are included in the ST but that are not included in this PP should be contained within the audit record.

5.1.3 Security Audit Review (FAU_SAR)

5.1.3.1 **Explicit: Audit Review (FAU_SAR_EXP.1)**

FAU_SAR_EXP.1.1 The TSF shall export audit records for use by authorized subjects.

FAU_SAR_EXP.1.2 The TSF shall provide the audit records in a manner suitable for an authorized subject to interpret the information.

Application Note: FAU_SAR_EXP.1.1 may be satisfied by placing the audit records in an internal "circular buffer" for which the authorized subject has responsibility for collecting the information prior to it becoming overwritten. This element requires the TSF to provide an external interface that the authorized subject can use to request the audit records.

Application Note: AGD_ADM_EXP.1.14C requires the administrator guidance to include information on how to interprete audit records.

5.1.4 Security Audit Event Selection (FAU_SEL)

5.1.4.1 **Selective Audit (FAU_SEL_EXP.1)**

FAU_SEL_EXP.1.1 The TSF shall be able to include auditable events in or exclude auditable events from the set of runtime audited events based on the following attributes as specified by the configuration data:

a) Resource identity,

b) Subject identity,

c) Event type,

d) Success of auditable security events,

e) Failure of auditable security events,

f) [selection: [assignment: *list of additional attributes specific to the audit capabilities of the implementation*], *no additional attributes*].

Application Note: The following clarification is provided with regard to the use of the words "audited" and "auditable" above. As used above "auditable events" refers to the totality of events that the TSF is capable of auditing at runtime. The set of "audited events" should be read as the "set of events to be audited during an execution session" and refers to the subset of auditable events for which the TSF will generate an audit event record should the indicated event occur during runtime.

The TSF is not required to provide a run-time capability for management of the audit function behavior. It is acceptable for the TSF to provide the means for the audit function behavior to be specified by the configuration data, and for that behavior to remain in effect and unchanged until such time that the TOE is initialized with a different set of audit configuration data.

5.2 User Data Protection (FDP)

5.2.1 Information Flow Control Policy (FDP_IFC)

5.2.1.1 Complete Information Flow Control (FDP_IFC.2)

FDP_IFC.2.1 **Refinement:** The TSF shall enforce the **Partitioned Information Flow SFP** on

- **All partitions**
- **All subjects**
- **All exported resources**

for all possible operations that cause information to flow between subjects and exported resources. **2**

FDP_IFC.2.2 **Refinement:** The TSF shall ensure that all operations that cause any information to flow **between** any subject **and any exported resource** are covered by an information flow control SFP. **3**

Application Note: Regardless of the abstraction at which information flows are enforced (refer to FDP_IFF.1.1-NIAP-0407), the policy applies to all partitions, all subjects, and all exported resources.

5.2.2 Information Flow Control Functions (FDP_IFF)

5.2.2.1 Simple Security Attributes (FDP_IFF.1)

FDP_IFF.1.1-NIAP-0407: **Refinement:** The TSF shall enforce the **Partitioned Information Flow SFP as a [selection: *Partition Abstraction, Least Privilege Abstraction*]** based on **the flow(s) caused by an operation, and the following types of partition, subject, and exported resource security attributes associated with the operation**: **4**

- **The identity of the subject involved in the flow of information;**
- **The identity of the partition to which the subject is assigned;**
- **The identity of the exported resource involved in the flow of information;**
- **The identity of the partition to which the exported resource is assigned.**

FDP_IFF.1.2-NIAP-0407 **Refinement:** The TSF shall permit **an** operation if**, for each flow associated with the operation, the following** rules hold: **5**

- For a TOE that is configured to enforce the PIFP as the Partition Abstraction:
  a) The identity of the subject is in the set of defined subjects for the identified partition;
  b) The identity of the exported resource is in the set of defined exported resources for the identified partition.
  c) For the identified partition-pair, the partition-pair rule explicitly authorizes the mode of the flow;
- For a TOE that is configured to enforce the PIFP as the Least Privilege Abstraction:
  a) The identity of the subject is in the set of defined subjects for the identified partition;
  b) The identity of the exported resource is in the set of defined exported resources for the identified partition.
  c) For the identified subject-exported resource pair
     1. a subject-exported resource rule explicitly authorizes the mode of the flow;

     -OR-

     2. the partition-pair rule corresponding to the subject-exported resource pair explicitly authorizes the mode of the flow, and
     3. the subject-exported resource pair rule is NULL for the mode of the flow.

Application Note: Regardless of which abstraction is chosen for an execution session, the authorized administrator has the responsibility to allocate subjects and exported resources to partitions and to specify their authorizations such that requirements of the chosen abstraction are met and the Principle of Least Privilege is achieved. Refer to Section 2 and AGD_ADM for further information about selecting the correct abstraction.

Application Note: The PIFP abstraction to be enforced during an execution session is determined by the configuration data (see FPT_ESS_EXP.1.2), and applies to all partitions: the TOE is either configured to enforce the Partition Abstraction or it is configured to enforce the Least Privilege Abstraction.

Application Note: An individual flow is characterized by the triplet consisting of the [partition/subject, partition/exported resource, mode] associated with the operation that invokes the flow.

Application Note: The identity of the subject and the partition to which it is assigned is required for both policy abstractions. For the Partition Abstraction, it is acceptable for a subject to share the identity of the partition to which it is assigned for the purpose of flow mediation.

Application Note: The identity of the resource and the partition to which it is assigned is required only for the Least Privilege Abstraction.

FDP_IFF.1.3-NIAP-0407 The TSF shall enforce the following information flow control rules: no additional information flow control SFP rules**.**

FDP_IFF.1.4-NIAP-0407: The TSF shall provide the following: no additional SFP capabilities.

FDP_IFF.1.5-NIAP-0407: The TSF shall explicitly authorize an information flow based on the following rules: no explicit authorization rules.

FDP_IFF.1.6-NIAP-0407: The TSF shall explicitly deny an information flow based on the following rules: no explicit denial rules.

5.2.2.2 Limited Illicit Information Flows (FDP_IFF.3)

FDP_IFF.3.1 The TSF shall enforce the **Partitioned Information Flow SFP** to limit the capacity of **covert timing channels and covert storage channels between partitions** to [assignment: ***metric establishing maximum covert channel capacity***].

5.2.3 Residual Information Protection (FDP_RIP)

5.2.3.1 Full Residual Information Protection (FDP_RIP.2)

FDP_RIP.2.1 **Refinement:** The TSF shall ensure that any previous information content of a resource is made unavailable upon the [selection: *allocation of the resource, deallocation of the resource*]. **6**

Application Note: The general intent of this requirement is to ensure that when a resource is reallocated, unauthorized access to the contents of the resource is prevented. This requirement applies to those cases where the resource is explicitly allocated to and later deallocated from a subject. In the case of an explicitly shared resource, such as two subjects with access to a common address space, the TSF is not required to sanitize the address space upon context switch to/from either of the subjects.

Application Note: This requirement applies to all TOE resources – those internal to the TSF as well as those exported by the TSF.

5.3 Identification and Authentication (FIA)

5.3.1 User Attribute Definition (FIA_ATD)

5.3.1.1 Explicit: User Attribute Definition (for partition attributes) (FIA_ATD_EXP.1(1))

FIA_ATD_EXP.1.1(1) The TSF shall maintain the following list of configuration data security attributes for each partition:

- Identity of the partition
- Minimum and maximum quotas for memory
- Minimum and maximum quotas for processing time
- Information flow authorizations
- [selection: [assignment: *list of additional partition security attributes*], *"no other partition security attributes"*].

5.3.1.2 Explicit: User Attribute Definition (for subject attributes) (FIA_ATD_EXP.1(2))

FIA_ATD_EXP.1.1(2) The TSF shall maintain the following list of configuration data security attributes for each subject:

- Identity of the subject
- Identity of the partition to which the subject is bound
- Subject authorizations
- Information flow authorizations
- [selection: [assignment: *list of additional subject security attributes*], *"no other subject security attributes"*].

5.3.1.3 Explicit: User Attribute Definition (for non-subject exported resource attributes) (FIA_ATD_EXP.1(3))

FIA_ATD_EXP.1.1(3) The TSF shall maintain the following list of configuration data security attributes for each non-subject exported resource:

- Identity of the non-subject exported resource
- Identity of the partition to which the non-subject exported resource is bound
- Information flow authorizations
- [selection: [assignment: *list of additional non-subject exported resource security attributes*], *"no other non-subject exported resource security attributes"*].

Application Note: The configuration data fulfills the function that is typically performed by an individual authorized to define users and to grant authorization to users for interaction with exported resources.

5.3.2 User-Subject Binding (FIA_USB)

5.3.2.1 Explicit: User-Subject Binding (for partition attribute binding) (FIA_USB_EXP.1(1))

FIA_USB_EXP.1.1(1) The TSF shall associate the following configuration data security attributes with partitions:

- Partition ID
- Partition minimum/maximum memory quotas
- Partition minimum/maximum processing time quotas
- Information flow authorizations to other partitions
- [selection: [assignment: *list of other partition security attributes*], *"no other partition security attributes"*].

FIA_USB_EXP.1.2(1) The TSF shall enforce the following rules on the initial association of configuration data security attributes with partitions:

a) The identity of the partition is in the set of defined partitions;

b) [assignment: *other rules for the initial association of attributes*].

FIA_USB_EXP.1.3(1) The TSF shall enforce the following rules governing changes to the configuration data security attributes associated with partitions: [assignment: *rules for the changing of attributes*].

5.3.2.2 Explicit: User-Subject Binding (for subject attribute binding) (FIA_USB_EXP.1(2))

FIA_USB_EXP.1.1(2) The TSF shall associate the following configuration data security attributes with subjects:

- Subject ID
- Partition ID to which the subject is to be bound
- Authorizations for invoking TSFI
- Information flow authorizations relevant to the abstraction selected in FDP_IFF.1.1-NIAP-0407
- [selection: [assignment: *list of other subject security attributes*], *"no other subject security attributes"*].

Application Note: In developing the ST, the ST developer must ensure that the information flow authorizations specified in the ST are consistent with the subject attributes required by the flow policy enforcement mechanism when determining if a specific flow is authorized.

FIA_USB_EXP.1.2(2) The TSF shall enforce the following rules on the initial association of configuration data security attributes with subjects:

a) The identity of the partition to which the subject is assigned is in the set of defined partitions;

b) [assignment: *other rules for the initial association of attributes*].

FIA_USB_EXP.1.3(2) The TSF shall enforce the following rules governing changes to the configuration data security attributes associated with subjects: [assignment: *rules for the changing of attributes*].

5.3.2.3 Explicit: User-Subject Binding (for non-subject exported resource attribute binding) (FIA_USB_EXP.1(3))

FIA_USB_EXP.1.1(3) The TSF shall associate the following configuration data security attributes with non-subject exported resources:

- Exported resource ID
- Partition ID to which the non-subject exported resource is to be bound
- Information flow authorizations relevant to the abstraction selected in FDP_IFF.1.1-NIAP-0407
- [selection: [assignment: *list of other non-subject exported resource security attributes*], *"no other non-subject exported resource security attributes"*].

Application Note: In developing the ST, the ST developer must ensure that the information flow authorizations specified in the ST are consistent with the non-subject exported resource attributes required by the flow policy enforcement mechanism when determining if a specific flow is authorized

FIA_USB_EXP.1.2(3) The TSF shall enforce the following rules on the initial association of configuration data security attributes with non-subject exported resources:

a) The identity of the partition to which the non-subject exported resource is assigned is in the set of defined partitions;

b) [assignment: *other rules for the initial association of attributes*].

FIA_USB_EXP.1.3(3) The TSF shall enforce the following rules governing changes to the configuration data security attributes associated with non-subject exported resources: [assignment: *rules for the changing of attributes*].

Application Note: The concept of user-subject binding applies to the TOE in the sense that the TSF is required to perform the binding of partition and exported resource attributes defined in the configuration data to the internal representation of those attributes for each partition, subject and non-subject exported resource when they are created during TOE initialization.

5.4 Security Management (FMT)

Application Note: This PP addresses security management of the TOE with the assumption that the TOE provides no capability for direct interaction between authorized administrators and the TSF during runtime. As a result, this profile does not address security management roles and the association of authenticated users to security management roles.

However, this profile requires that the TOE must, by design, provide inherent support for the various type of security management functions that are typically performed by authorized administrators (e.g., trusted initialization, definition of initialization parameters, policy definition and enforcement attributes governing subject/resource interaction, TSF function behavior, fault detection and response, trusted recovery, and TOE reconfiguration). This profile expects that the combination of off-line TOE tools and procedures and TSF functionality will serve to implement the totality of the required security management functions.

Application Note: The security management components contained in this profile define the minimal set of required capability, consistent with terms defined in the Glossary and discussion of the TOE found in Section 2.

In this regard, the profile requires that only authorized subjects are able to invoke a change of the operational configuration to a new configuration, and requires that the Partitioned Information Flow SFP enforcement attributes be defined completely by each configuration vector. The management functions associated with these capabilities are found in this section.

Application Note: This profile allows the TOE developer to provide dynamic configuration change capability. Should the TOE developer wish to implement greater dynamicity in the reconfiguration capability, then it is the responsibility of the TOE developer to express the detailed requirements of that capability in the Security Target. The Security Target must address both the functional requirements for the dynamic configuration change capability as well any derived requirements to ensure that the reconfiguration capability is consistent with the objectives of this profile and does in fact satisfy them.

5.4.1 Explicit: Management of Configuration Data (FMT_MCD_EXP)

5.4.1.1 **Explicit: Management of Configuration Data (FMT_MCD_EXP.1)**

FMT_MCD_EXP.1.1 The TSF shall prevent unauthorized modification of the configuration data.

Application Note: The TSF is required to maintain configuration data and TSF internal vector set consistent with its capability to reconfigure. The configuration data defines the TSF configuration and the initial secure state of the TSF. It must be protected from modification to preserve the integrity of the TSF secure state at all times including the following: shutdown, start-up, restart, and configuration change of the TOE into the same configuration or some other configuration.

5.4.2 Management of Functions in TSF (FMT_MOF)

Application Note: The requirement for the TSF to "restrict the ability …" means that the TSF must have the means to prevent unauthorized subjects from invoking the indicated capability/service provided by the TSF. A TOE that prevents all subjects within the TSC from invoking the capability (i.e., no subject can have the authorization) meets requirements of this form. However, for cases where the TOE Environment invokes a TSF-provided service (e.g., via hardware watchdog timer, etc), those entities are considered "authorized", and an argument is to be made to demonstrate that other entities are not able to invoke the capability/service.

Application Note: Some SFRs state requirements for restrictions associated with capabilities that are optional and which, if selected in the Security Target, have security-relevant implications. For those cases, the requirement to restrict the ability to invoke a non-implemented capability is met by the fact that the capability does not exist and the TOE would no concept of an authorized subject for that capability.

5.4.2.1 Management of Security Functions Behavior (to change the TOE configuration) (FMT_MOF.1(1))

FMT_MOF.1.1(1) **Refinement:** The TSF shall restrict the ability to **invoke a configuration change of the TOE** to **authorized subjects.**[7]

5.4.2.2 Management of Security Functions Behavior (to restart the TOE) (FMT_MOF.1(2))

FMT_MOF.1.1(2) **Refinement:** The TSF shall restrict the ability to **invoke a restart of the TOE** to **authorized subjects.** [8]

Application Note: The restart function will result in the execution of the initialization function. See Glossary of Terms section for a description of the initialization function.

5.4.2.3 Management of Security Functions Behavior (to halt the TOE) (FMT_MOF.1(3))

FMT_MOF.1.1(3) **Refinement:** The TSF shall restrict the ability to **invoke a halt of the TOE** to **authorized subjects.** [9]

5.4.2.4 Management of Security Functions Behavior (to initiate TOE self-tests) (FMT_MOF.1(4))

FMT_MOF.1.1(4) **Refinement:** The TSF shall restrict the ability to **initiate TSF self-tests** to **authorized subjects.** [10]

5.4.2.5 Management of Security Functions Behavior (to transition the TOE to maintenance mode) (FMT_MOF.1(5))

FMT_MOF.1.1(5) **Refinement:** The TSF shall restrict the ability to **invoke a transition of the TOE to maintenance mode** to **authorized subjects.** [11]

1Application Note: The configuration data alone provides the designation of those subjects with authorization to perform the following: invoke a configuration change of the TOE, invoke a restart of the TOE, invoke a halt of the TOE, initiate TSF self-tests and invoke a transition of the TOE to maintenance mode..

Application Note: FMT_MSA_EXP allows additional authorizations to be assigned to subject. Should the TOE developer choose to implement such authorizations, additional iterations of this component in the form of the above refinements must be included in the ST.

5.4.3 Management of Security Attributes (FMT_MSA)

5.4.3.1 Explicit: Management of Security Attributes (FMT_MSA_EXP.1)

FMT_MSA_EXP.1.1 The TSF shall assign the following authorizations to subjects as specified by the configuration data:

- Ability to invoke a TOE configuration change,
- Ability to invoke a TOE restart,
- Ability to invoke a TOE halt,
- Ability to invoke TSF self-tests,
- Ability to obtain results of TSF self-tests,
- Ability to enter a maintenance mode,
- Ability to obtain audit information,
- [assignment: *list of additional authorizations that may be assigned to subjects*].

FMT_MSA_EXP.1.2 The TSF shall only assign authorizations to subjects as specified by the configuration data.

Application Note: FPT_TST allows authorized subjects to invoke and obtain the results of the TSF self-tests. Should that capability be implemented, this requirement supports designation of those subjects authorized to perform those functions.

5.4.3.2 Explicit: Static Policy Attribute Initialization (FMT_MSA_EXP.3)

FMT_MSA_EXP.3.1 The TSF shall provide restrictive default values for each attribute that has not been assigned a value by the configuration data.

Application Note: This requirement applies to all attributes associated with the establishment of secure state that occurs during the initialization of the TOE (to include initialization as a result of system power-on, as part of a reconfiguration, or as part of a trusted recovery).

5.4.4 Management of TSF Data (FMT_MTD)

5.4.4.1 Management of TSF Data (for obtaining TSF self-test results) (FMT_MTD.1(1))

FMT_MTD.1.1(1) The TSF shall restrict the ability to **obtain** the **results of TSF self-tests** to **authorized subjects**.

5.4.4.2 Management of TSF Data (for obtaining audit information) (FMT_MTD.1(2))

FMT_MTD.1.1(2) The TSF shall restrict the ability to **obtain audit information** to **authorized subjects.**

5.4.4.3 Secure TSF Data (FMT_MTD.3)

FMT_MTD.3.1 **Refinement**: The TSF shall ensure that only **valid** values are accepted for TSF data. **12**

Application Note: Valid implies that the values fall within the defined range for the TSF data (e.g., an audit enable/disable indicator must be within range of a Boolean type).

5.4.5 Specification of Management Functions (FMT_SMF)

5.4.5.1 Specification of Management Functions (FMT_SMF.1)

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

- **Restart the TOE,**
- **Halt the TOE,**
- **Conduct TSF self-tests,**
- **Transition the TOE to maintenance mode,**
- **[selection: *change the TOE configuration*, [assignment: *additional management functions*], *"no additional management functions"*]**

Application Note: The selection must be consistent with FPT_CFG_EXP.1.1. Therefore, the ST author must select "change the TOE configuration" above if the TOE provides a configuration change capability as indicated by the selection made in FPT_CFG_EXP.1.1.

5.5 Protection of the TSF (FPT)

5.5.1 Underlying Abstract Machine Test (FPT_AMT)

5.5.1.1 Abstract Machine Testing (FPT_AMT.1)

FPT_AMT.1.1 **Refinement:** The TSF shall run a suite of tests during start-up, periodically during normal operation, **during recovery**, **and [assignment: *other conditions under which abstract machine testing should occur*]** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the **software portions of the** TSF.[13]

Application Note: The assignment statement is to be completed to express all forms of periodic abstract machine testing capability implemented by the TSF. Note that should the TSF implement a configuration change capability (ref: FPT_CFG_EXP.1), then "other conditions …" should include configuration change.

Application Note: The test suite need only cover aspects of the underlying abstract machine on which the TSF relies for policy support and enforcement, to include domain separation. The test suite for periodic testing may be a subset of the start-up test. The periodic test suite should constitute the maximum set of tests that can be run without interfering with the normal system operation. The periodic test suite may be further divided into different test groups. Each test group may be scheduled to run at different times during run-time.

Application Note: Annex J of the CC, Part 2, explains that with respect to the FPT class, the TSF consists of three parts: a) the TSF's abstract machine, b) the TSF's implementation, and c) the TSF data. This component covers the testing of the TSF's abstract machine which is defined in Annex J as "the virtual or physical machine upon which the specific TSF implementation under evaluation executes."

Application Note: It is intended that the abstract machine test suite be run as part of all recovery actions identified in FPT_RCV_EXP.2.

5.5.2 Explicit: Configuration Change (FPT_CFG_EXP)

5.5.2.1 **Explicit: Configuration Change (FPT_CFG_EXP.1)**

FPT_CFG_EXP.1.1 The TSF shall provide [selection: *dynamic total configuration change capability*, *dynamic constrained configuration change capability, dynamic unconstrained configuration change capability, no configuration change capability*].

Application Note: It is the intent of this profile to not mandate that all separation kernel implementations provide a configuration change capability; such a capability is not required for all product types based on a separation kernel. However, this profile recognizes that any capability to change the configuration of the TOE must be implemented in a manner that preserves the security objectives. Therefore, should the TOE developer choose to implement a configuration change capability, that capability is a component of the TSF and must be fully addressed in the ST. Since the CC provides no mechanism to express optional requirements, this profile has chosen to employ the selection operation to articulate this choice to the TOE developer. Should the TOE developer select "no configuration change capability" in the above selection, then FPT_CFG_EXP.1.2 and FPT_CFG.1.3 do not apply..

FPT_CFG_EXP.1.2 The TSF shall enforce the following rules when changing the configuration of the TOE:

1) For the dynamic total configuration change capability:
   a) The TSF shall maintain a configuration vector set containing multiple configuration vectors;
   b) The TSF shall allow an authorized subject to select the next TSF internal vector from the TSF internal vector set;

Application Note: The term "next" is used in FPT_CFG_EXP to convey the temporal relationship that exists between the current TSF internal vector and the one selected or constructed by an authorized subject (i.e., the "next" TSF internal vector). It is not used to convey a predefined sequential ordering of TSF internal vectors.

Regarding condition 1b), the PP does not dictate that the designation of the next configuration and the execution of the change be a single, atomic event: the changeover could be immediate, or delayed. The authorized subject can choose to delay the changeover until the next initialization (e.g., see FPT_RST_EXP), if the TSF provides the means to export the choice of "next configuration" such that it is accessible by the TOE initialization mechanism.

Application Note: The PP does not dictate that the TOE transition to the halt state before the initialization process begins.

2) For the dynamic constrained configuration change capability:
   a) The TSF shall allow an authorized subject to specify new values for the following TOE configuration attributes [assignment: *list of TOE configuration attributes*], thus defining the next TSF internal vector;
   b) The TSF shall enforce the following mandatory constraints on all TOE configuration attributes [assignment: *list of mandatory constraints to changes of the TOE configuration*].
   c) For each TOE configuration attribute that may be changed, the TSF shall impose [selection: [assignment: *list of constraints specific to each attribute that can be changed*], *no constraints*] on changes to that attribute;

3) For the dynamic unconstrained configuration change capability:
   a) The TSF shall allow an authorized subject to change the following TOE configuration attributes [assignment: *list of TOE configuration attributes*], thus defining the next TSF internal vector;

Application Note: In item 2b), the mandatory constraints enforced by the TSF apply globally, i.e., to all attributes irrespective of constraints and conditions specified in item 2c). In item 2c), the constraints enforced by the TSF apply on a per-attribute basis.

FPT_CFG_EXP.1.3 When requested by an authorized subject, the TSF shall change the configuration data to the values specified in the next TSF internal vector.

FPT_CFG_EXP.1.4 The TSF shall preserve secure state during any change of TOE configuration.

Application Note: Refer to Section 2 for discussion of the configuration change definitions, concepts and options for implementation.

Application Note: The intent of this profile is to not restrict the dynamicity of the implemented configuration change capability. However, dynamic configuration change capabilities require that associated management controls assurances be provided to ensure that secure state is preserved and the core security properties of the TOE (as expressed by the Security Objectives) are continuously met.. The Security Target must fully address this concern in its functional and assurance requirements. It is beyond the scope of this Protection Profile to define acceptable examples of ST requirements that would guarantee the continuity of secure state during the course of dynamic configuration changes.

5.5.3 Explicit: Establishment of Secure State (FPT_ESS_EXP)

5.5.3.1 **Explicit: Establishment of Secure State (FPT_ESS_EXP.1)**

FPT_ESS_EXP.1.1 The TSF shall be established in a secure state as defined by the configuration vector.

FPT_ESS_EXP.1.2 The TSF shall enforce the Partitioned Information Flow Policy (PIFP) in accordance with the PIFP abstraction specified by the configuration data.

FPT_ESS_EXP.1.3 The TSF shall verify that it is in a secure state upon completion of the TOE initialization function and prior to authorizing any information flows governed by the Partitioned Information Flow Policy (PIFP).

Application Note: FPT_ESS_EXP.1.1 expresses the requirement that the TSF shall be established in a secure state – which is not a function of the TSF but is a function levied on the TOE. ADV_ARC_EXP expresses requirements for how that secure state must be achieved.

Application Note: FPT_ESS_EXP.1.2 expresses the requirement for the TSF to use the configuration data to establish the rules for PIFP enforcement. Note that the FDP_IFC/IFF requirements address only the existence of the enforcement function. FPT_ESS requires that the security policy enforced by that function shall be based on the configuration data.

Application Note: FPT_ESS_EXP1.3 expresses the need for the TSF to verify that it is in a secure state prior to allowing any information flows to occur.

5.5.4 Fail Secure (FPT_FLS)

5.5.4.1 Failure with Preservation of Secure State (FPT_FLS.1)

FPT_FLS.1.1 The TSF shall preserve a secure state when the following types of failures occur:

a) [assignment: *list of failures that are detected by tests defined in FPT_AMT.1 and FPT_TST_EXP.1*];

b) [assignment: *other failures in the TSF*].

Application Note: TSF failure modes vary and may include "hard" failures such as those associated with hardware failure or unrecoverable software errors, and "soft" failures such as intermittent hardware errors and recoverable software errors.

Application Note: The TSF is not expected to protect itself against all types of hardware errors. For example, a radiation induced change of a single bit in a memory access control register could result in an incorrect (but valid) memory location being accessed. This would not always be detected by the hardware.

5.5.5 Explicit: TOE Halt (FPT_HLT_EXP)

5.5.5.1 Explicit: TOE Halt (FPT_HLT_EXP.1)

FPT_HLT_EXP.1.1 When requested by [selection: *an authorized subject executing on the TOE, a trusted individual in the TOE non-IT environment, an authorized subject in the TOE IT environment*], the TSF shall halt the TOE.

FPT_HLT_EXP.1.2 The TSF shall preserve secure state when halting the TOE.

Application Note: The ability to halt the TOE is security-relevant. The intent of this requirement is to ensure that only authorized subjects or trusted individuals are able to halt the TOE. The PP authors recognize that this capability might be provided via a software or hardware interface to the TSF. The ability of the TSF to securely halt the TOE via this interface must be demonstrated.

5.5.6 Explicit: TOE Maintenance (FPT_MTN_EXP)

5.5.6.1 Explicit: TOE Maintenance (FPT_MTN_EXP.1)

FPT_MTN_EXP.1.1 When requested by an authorized subject, the TSF shall transition the TOE to maintenance mode.

Application Note: The TSF may or may not be in a secure state when this function is invoked.

FPT_MTN_EXP.1.2 When maintenance mode is entered from a secure state, the TSF shall continue to preserve secure state.

FPT_MTN_EXP.1.3 When the TSF is unable to preserve secure state after transitioning to maintenance mode from a secure state, the TSF shall halt the TOE.

5.5.6.2 Explicit: TOE Maintenance Secure (FPT_MTN_EXP.2)

FPT_MTN_EXP.2.1 When in maintenance mode, the TSF shall reject the request for any operations that would result in a violation of the TSP.

Application Note: It is acceptable, but not required, to prevent all operations and non-TSF invoked flows.

5.5.7 Explicit: Principle of Least Privilege (FPT_PLP_EXP)

5.5.7.1 Explicit: TSF Least Privilege (FPT_PLP_EXP.1)

FPT_PLP_EXP.1.1 The TSF shall enforce the TSP such that each internal function has no more access to TSF data and other internal TSF resources than that which is required for its assigned functionality.

Application Note: This SFR establishes the behavioral property that the TSF must exhibit with respect to the maximum set of privileges allocated to the various modules that comprise the functions specified by the set of SFRs contained in the ST. Refer to ADV_INT_EXP.3.16C for the assurance evidence that must be provided to substantiate that this required behavioral property exists in the TSF implementation.

Application Note: Achieving least privilege does not require separate domains for each TSF function.

5.5.8 Explicit: Trusted Recovery (FPT_RCV_EXP)

5.5.8.1 Explicit: Automated Recovery (FPT_RCV_EXP.2)

FPT_RCV_EXP.2.1 When the TSF determines that it is not in a secure state immediately after completion of TOE initialization or at any time while the TOE is in operational mode, the TSF shall attempt to recover the TOE to a secure state without further protection compromise based on the following: [assignment: *list of condition-action pair(s) where each condition is associated with one of the following action(s)* [selection: *initiate and complete recovery action while remaining in operational mode, initiate recovery action that results in a restart of the TOE without transitioning to maintenance mode, transition the TOE to maintenance mode and initiate recovery action while in maintenance mode, halt the TOE without initiating any recovery action*]].

FPT_RCV_EXP.2.2 When the TSF determines that it is unable to initiate or complete a recovery action that requires the TOE to remain in operational mode, the TSF shall [selection: *attempt to transition the TOE to maintenance mode*, *halt the TOE*].

FPT_RCV_EXP.2.3 When the TSF determines that it is unable to initiate or complete a recovery action that requires the TOE to restart without transitioning to maintenance mode, the TSF shall [selection: *attempt to transition the TOE to maintenance mode*, *halt the TOE*].

FPT_RCV_EXP.2.4 When the TSF determines that it is unable to initiate or complete a transition to maintenance mode or is unable to complete a recovery action after transitioning to maintenance mode, the TSF shall halt the TOE.

FPT_RCV_EXP.2.5 When the TSF determines that it is unable to proceed with any recovery action, the TSF shall attempt to halt the TOE.

Application Note: The TOE developer is to provide appropriate evidence and the evaluator is to confirm that secure state results from the recovery action identified.

Application Note: There is no requirement that the TSF alone supports the recovery action to transition from maintenance mode to a secure state in operational mode.

Application Note: The ST developer should select halting the TOE instead of transitioning the TOE to a maintenance mode if the TOE implementation cannot meet the requirement defined in FPT_MTN_EXP.2.

5.5.8.2 Function Recovery (FPT_RCV.4)

FPT_RCV.4.1 The TSF shall ensure that [assignment: *list of all failure scenarios, and for each listed scenario, the affected SFs*] have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

5.5.9 Explicit: TOE Restart (FPT_RST_EXP)

5.5.9.1 Explicit: TOE Restart (FPT_RST_EXP.1)

FPT_RST_EXP.1.1 When requested by an authorized subject, the TSF shall restart the TOE.

FPT_RST_EXP.1.2 The TSF shall preserve secure state during a restart of the TOE.

Application Note: A restart of the TOE will result in the execution of the TOE initialization mechanism. Therefore, the restart function can serve as the means by which a change in TOE configuration is executed (see FPT_CFG_EXP.1).

Application Note: The PP authors also recognize that a restart capability can be triggered via the sequence of HALT and START triggered from the TOE environment. It is not the intent of this requirement that the TOE have an atomic restart capability, the intent is that should the capability exist, then there is assurance that it can not be invoked by an unauthorized subject.

5.5.10 Reference Mediation (FPT_RVM)

5.5.10.1 Non-Bypassability of the TSP (FPT_RVM.1)

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

5.5.11 Domain Separation (FPT_SEP)

5.5.11.1 Complete Reference Monitor (FPT_SEP.3)

FPT_SEP.3.1 **Refinement:** The unisolated portion of the TSF shall **use hardware mechanisms to** maintain a security domain for its own execution that protects **the code and data of the unisolated portion of the TSF** from interference and tampering by untrusted subjects.[14]

Application Note: Examples of hardware mechanisms that might be used to support a protected security domain for the execution of the TSF include: privilege bits; rings; hardware mechanisms that support controlled entry points to domains; and a variety of memory management features.

FPT_SEP.3.2 The TSF shall enforce separation between the security domains of subjects in the TSC.

FPT_SEP.3.3 **Refinement:** The TSF shall maintain the part of the TSF that enforces the information flow control SFPs in a security domain for its own execution that protects **that part of the TSF** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to the TSP.[15]

Application Note: In this PP, there is no access control SFP, and the Partitioned Information Flow Policy is the only information flow control SFP.

Application Note: For FPT_SEP.3.3, it is not required that hardware mechanisms be used to separate the unisolated portion of the TSF from the part of the TSF that enforces the information flow control SFPs. Software separation mechanisms and appropriate architecture development evidence (see ADV_ARC_EXP.1.4C), supported by penetration testing (see AVA_VLA_EXP.4) and a justification for using the software separation mechanisms instead of hardware mechanisms, are sufficient to meet this requirement.

5.5.12 Time Stamps (FPT_STM)

5.5.12.1 Reliable Time Stamp (FPT_STM.1)

FPT_STM.1.1 **Refinement:** The TSF shall be able to provide reliable time stamps for its own use **that meet [assignment: *granularity/precision of time stamp*]**.[16]

Application Note: It is the responsibility of the ST developer to provide a definition and metric for the term "reliable time stamp" and to provide evidence that the implementation meets the defined definition and metric.

Application Note: It is not required that "time" be maintained or provided as "time of day". It is acceptable for the TOE to maintain and provide time as being "relative to" some TOE event (e.g.,, relative to TOE initialization), such that it is possible to determine the ordering of events as allowed by the precision of the chosen unit of time. Therefore, the native format in which the TOE keeps time for internal use is acceptable. For example, a monotonically increasing counter with a defined metric for each increment of the counter represents one acceptable implementation of this requirement.

Application Note: The time stamp definition and metric, and means to interpret the chosen time stamp format are to be documented in the administrative guidance for the TOE. The rationale in the ST should be used to substantiate the chosen definition and metric.

5.5.13 Explicit: TSF Self Test (FPT_TST_EXP)

5.5.13.1 **Explicit: TSF Testing (FPT_TST_EXP.1)**

FPT_TST_EXP.1.1 The TSF shall run a suite of self tests during start-up, periodically during normal operation, during recovery, at the request of an authorized subject, and [selection: *during configuration change,* [assignment: *other conditions under which self test occurs*]] to demonstrate the correct operation of the software portion of the TSF implementation.

Application Note: See Annex J of the CC, Part 2, for an explanation of the notion of TSF's implementation.

Application Note: The assignment statement is to be completed to express all additional periodic self-test execution capabilities implemented by the TSF. "During configuration change" should be selected if the TOE provides that capability.

Application Note: It is intended that TSF seft tests are run as part of all recovery actions identified in FPT_RCV_EXP.2.

FPT_TST_EXP.1.2 The TSF suite of self tests shall verify the integrity of TSF configuration data and [assignment: *list of additional TSF data upon which the TSF depends to enforce its security policies correctly*].

FPT_TST_EXP.1.3 The TSF suite of self tests shall verify the integrity of stored TSF executable code.

FPT_TST_EXP.1.4 The TSF shall provide the results of the self tests to authorized subjects in a form that allows assessment of the results.

Application Note: TSF self-test results include both results of tests that determine the correct operation of the software portions of the TSF and the results of integrity tests on TSF data and TSF executable code.

5.6 Resource Utilization (FRU)

5.6.1 Resource Allocation (FRU_RSA)

5.6.1.1 Minimum and Maximum Quotas (FRU_RSA.2)

FRU_RSA.2.1 **Refinement:** The TSF shall enforce maximum quotas of the following resources **for each partition, as defined by the configuration data:**

- **System memory: [assignment:** ***maximum amount of memory that each partition can use*****],**
- **Processing time: [assignment:** ***maximum amount of processing time allocated to a partition for a specified period of time*****].** [17]

FRU_RSA.2.2 **Refinement:** The TSF shall ensure the **availability to each partition** of minimum **quantities** of **the following resources, as defined by the configuration data:**

- **System memory: [assignment:** ***minimum amount of memory that will be provided to each partition*****],**
- **Processing time: [assignment:** ***minimum amount of processing time provided to a partition for a specified period of time*****].** [18]

Application Note: The enforcement of memory allocation by the TSF is at the granularity of the partition abstraction. That is, there is a fixed amount of memory, possibly spanning multiple distinct subject address spaces, allocated to each partition. The TSF is not required to enforce minimum or maximum quotas at the granularity of subjects.

Application Note: The enforcement of time allocation by the TSF is at the granularity of the partition abstraction. That is, there is a fixed amount of time over some specified period of time that is allocated for all subjects bound to a single partition.

Application Note: It is acceptable for the minimum allocation of processing time to be zero, implying that a there is no guarantee that any subject bound to that partition will be scheduled during the specified period of time.

5.6.2 Explicit: Predictable Resource Utilization by the TSF (FRU_PRU_EXP.1)

5.6.2.1 Explicit: TSF Predictable Resource Utilization (FRU_PRU_EXP.1.1)

FRU_PRU_EXP.1.1 The TSF shall exhibit predictable and bounded execution behavior with respect to its usage of processor time and memory resources.

Application Note: The TOE developer is to document the expectations for memory and processor usage by the TSF in completing ADV_ARC_EXP.1.5C.

End Notes

This section records the functional requirements where deletions of Common Criteria text were performed.

1 Modifications of CC text were performed in FAU_ARP.1.1. Rationale: 1) To ensure clarity in expressing the intent of this requirement, the words "least disruptive actions" were replaced by the words "actions to take" and 2) to provide specific information on the types of security violations that are valid for this particular TOE without extending the scope of the requirement, the words "a potential security violation" were replaced with "any failure of the tests defined in FPT_AMT.1 and FPT_TST.1".

FAU_ARP.1.1 **Refinement:** The TSF shall take [assignment: list of the ~~least disruptive~~ actions **to take**] upon detection of ~~a potential security violation~~ **any failure of the tests defined in FPT_AMT.1 and FPT_TST.1**.

2 Modifications of CC text were performed in FDP_IFC.2.1. Rationale: The phrase "and all operations that cause that information to flow to and from subjects covered by the SFP" was replaced by the phrase "for all possibleoperations that cause information to flow between subjects and exported resources." The refinement is intended to emphasis that the choice of abstraction applies to all operations.

FDP_IFC.2.1 **Refinement:** The TSF shall enforce the **Partitioned Information Flow SFP** on

- **All partitions**
- **All subjects**
- **All exported resources**
- ~~and all operations that cause that information to flow to and from subjects covered by the SFP~~ **for all possible operations that cause information to flow between subjects and exported resources.**

3 Modifications of CC text were performed in FDP_IFC.2.2. Rationale: The phrase "to flow to and from any subject in the TSC" was reworded as "to flow between any subject and any exported resource" to clarify that the information flow policy is enforced on all subjects and exported resources. References to the TSC were removed to eliminate redundancy, i.e., there are no subjects, resources and operations that are outside of the TSC.

FDP_IFC.2.2 **Refinement:** The TSF shall ensure that all operations that cause any information ~~in the TSC~~ to flow ~~to and from~~ **between** any subject ~~in the TSC~~ **and any exported resource** are covered by an information flow control SFP.

4 A modification of the US interpretation I-0407 text was performed in FDP_IFF.1.1. Rationale: 1) the words and operation "as a [selection: *Partition Abstraction, Least Privilege Abstraction*]" was added to allow the TOE developer to specify the granularity at which the TOE is capable of enforcing the PIFP, 2) The words "the

following types of subject and information security attributes" were changed to require the TSF to make policy decisions based on both the flows caused by an operation and the partition, subject and exported resource security attributes associated with that operation.

FDP_IFF.1.1-NIAP-0407 **Refinement:** The TSF shall enforce the **Partitioned Information Flow SFP as a [selection:** ***Partition Abstraction, Least Privilege Abstraction*****]** based on ~~the following types of subject and information security attributes~~ **the flow(s) caused by an operation, and the following types of partition, subject, and exported resource security attributes associated with the operation:**

5 Modifications of the US interpretation I-0407 text were performed in FDP_IFF.1.2. Rationale: 1) The words "an information flow between a controlled subject and controlled information via a controlled" were deleted and 2) the words "the following" were replaced by the words "for each flow associated with the operation, the following". The changes are to clarify that an operation is permitted only if all flows associated with that operation are permitted, and that the rules apply to all partitions.

FDP_IFF.1.2-NIAP-0407 **Refinement:** The TSF shall permit ~~an information flow between a controlled subject and controlled information via a controlled~~ **an** operation if ~~the following~~**, for each flow associated with the operation, the following** rules hold:

6 Modifications of CC text were performed in FDP_RIP.2.1. Rationale: The words "to" and "from" were deleted for readability and the words "all objects" were deleted because object is not the abstraction used for this PP.

FDP_RIP.2.1 **Refinement:** The TSF shall ensure that any previous information content of a resource is made unavailable upon the [selection: *allocation of the resource ~~to~~*, *deallocation of the resource ~~from~~*] ~~all objects~~.

7 A modification of CC text was performed in FMT_MOF.1.1(1). Rationale: The words "selection: determine the behaviour of, disable, enable, modify the behaviour of] the functions" were replaced with a precise statement of the behavioral requirement.

FMT_MOF.1.1(1) **Refinement:** The TSF shall restrict the ability to ~~[selection: determine the behaviour of, disable, enable, modify the behaviour of]~~ ~~the functions~~ **invoke a configuration change of the TOE** to **authorized subjects**.

8 A modification of CC text was performed in FMT_MOF.1.1(2). Rationale: The words "selection: determine the behaviour of, disable, enable, modify the behaviour of] the functions" were replaced with a precise statement of the behavioral requirement.

FMT_MOF.1.1(2) **Refinement:** The TSF shall restrict the ability to ~~[selection: determine the behaviour of, disable, enable, modify the behaviour of]~~ ~~the functions~~ **invoke a restart of the TOE** to **authorized subjects**.

9 A modification of CC text was performed in FMT_MOF.1.1(3). Rationale: The words "selection: determine the behaviour of, disable, enable, modify the behaviour of] the functions" were replaced with a precise statement of the behavioral requirement.

FMT_MOF.1.1(3) **Refinement:** The TSF shall restrict the ability to ~~[selection: determine the behaviour of, disable, enable, modify the behaviour of]~~ ~~the functions~~ **invoke a halt of the TOE** to **authorized subjects**.

10 A modification of CC text was performed in FMT_MOF.1.1(4). Rationale: The words "selection: determine the behaviour of, disable, enable, modify the behaviour of] the functions" were replaced with a precise statement of the behavioral requirement.

FMT_MOF.1.1(4) **Refinement:** The TSF shall restrict the ability to ~~[selection: determine the behaviour of, disable, enable, modify the behaviour of]~~ ~~the functions~~ **initiate TSF self-tests** to **authorized subjects**.

11 A modification of CC text was performed in FMT_MOF.1.1(5). Rationale: The words "selection: determine the behaviour of, disable, enable, modify the behaviour of] the functions" were replaced with a precise statement of the behavioral requirement.

FMT_MOF.1.1(5) **Refinement:** The TSF shall restrict the ability to ~~[selection: determine the behaviour of, disable, enable, modify the behaviour of]~~ ~~the functions~~ **invoke a transition of the TOE to maintenance mode**

to **authorized subjects**.

12 A modification of CC text was performed in FMT_MTD.3.1. Rationale: The word "secure" was changed to "valid" to indicate that this is intended to be a syntax check.

FMT_MTD.3.1 **Refinement:** The TSF shall ensure that only ~~secure~~ **valid** values are accepted for TSF data.

13 A modification of CC text was performed in FPT_AMT.1.1. Rationale: The selection "during initial start-up" was changed to "during start-up" to indicate that the tests are to run every time the TOE is started and the words "software portions of the" were added for clarity.

FPT_AMT.1.1 **Refinement:** The TSF shall run a suite of tests during ~~initial~~ start-up, periodically during normal operation, **during recovery**, **and [assignment: *other conditions under which abstract machine testing should occur*]** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the **software portions of the** TSF.

14 A modification of CC text was performed in FPT_SEP.3.1. Rationale: 1) The words "use hardware mechanisms to" were added to require the use of hardware mechanisms to maintain a protected security domain for the unisolated portion of the TSF; 2) The word "it" was replaced by the words "the code and data of the unisolated portion of the TSF" for clarity.

FPT_SEP.3.1 **Refinement:** The unisolated portion of the TSF shall **use hardware mechanisms to** maintain a security domain for its own execution that protects ~~it~~ **the code and data of the unisolated portion of the TSF** from interference and tampering by untrusted subjects.

15 A modification of CC text was performed in FPT_SEP.3.3. Rationale: The word "them" was replaced by the words "that part of the TSF" for clarity.

FPT_SEP.3.3 **Refinement:** The TSF shall maintain the part of the TSF that enforces the information flow control SFPs in a security domain for its own execution that protects ~~them~~ **that part of the TSF** from interference and tampering by the remainder of the TSF and by subjects untrusted with respect to the TSP.

16 A modification of CC text was performed in FPT_STM.1.1. Rationale: The words "that meet [assignment: granularity/precision of time stamp]" were added to require the ST author to specify the granularity/precision of the time stamps.

FPT_STM.1.1 **Refinement:** The TSF shall be able to provide reliable time stamps for its own use **that meet [assignment: *granularity/precision of time stamp*]**.

17 A modification of CC text was performed in FRU_RSA.2.1. Rationale: 1) The words "for each partition as defined by the configuration data" were added to make it clear that maximum quotas are assigned at the partition level; 2) The selection statement for "individual user, defined group of users" was deleted since the TOE does not support the concept of individual users or groups of users..

FRU_RSA.2.1 **Refinement:** The TSF shall enforce maximum quotas of the following resources ~~[assignment: controlled resources] that [selection: individual user, defined group of users] can use [selection: simultaneously, over a specified period of time]~~ **for each partition, as defined by the configuration data:**

- **System memory: [assignment: *maximum amont of memory that each partition can use*],**
- **Processing time: [assignment: *maximum amount of processing time allocated to a partition for a specified period of time*].**

18 A modification of CC text was performed in FRU_RSA.2.2. Rationale: 1) The phrase "provision of minimum quantity of each" was replaced by the words "availability to each partition, the minimum quantities of the following resources, as defined by the configuration data" to make it clear that minimum quotas are assigned at the partition level; 2) The selection statement for "individual user, defined group of users, subject" was deleted since the TOE does not support the concept of invidividual users or groups of users, and the subject is not the entity to which minimum quotas are assigned.

FRU_RSA.2.2 **Refinement:** The TSF shall ensure the ~~provision~~ **availability to each partition** of minimum ~~quantity of each [assignment: controlled resource] that is available for [selection: an individual user, defined~~

~~group of users, subjects] to use [selection: simultaneously, over a specified period of time]~~ **quantities of the following resources, as defined by the configuration data:**

- **System memory: [assignment:** ***minimum amount of memory that will be provided to each partition*****],**
- **Processing time: [assignment:** ***minimum amount of processing time provided to a partition for a specified period of time*****].**

6. TOE Security Assurance Requirements

126 This section contains the detailed security assurance requirements for Separation Kernels supporting systems in environments requiring high robustness. The requirements contained in this section are either selected from Part 3 of the CC or have been explicitly stated (with short names ending in "_EXP"). Table 6.1 lists the explicitly stated assurance components.

Table 6.1. Explicit Assurance Requirements

Explicit Component	Component Behavior Name
ADO_DEL_EXP.2	Detection of Modification
ADV_ARC_EXP.1	Architectural Design
ADV_CTD_EXP.1	Configuration Tool Design
ADV_FSP_EXP.4	Formal Functional Specification
ADV_HLD_EXP.4	Semiformal High Level Design
ADV_IMP_EXP.3	Verified Implementation of the TSF
ADV_INI_EXP.1	Trusted Initialization
ADV_INT_EXP.3	Minimization of Complexity
ADV_LLD_EXP.2	Semiformal Low Level Design
ADV_LTD_EXP.1	Load Tool Design
AGD_ADM_EXP.1	Administrator Guidance
AMA_AMP_EXP.1	Assurance Maintenance Plan
APT_PDF_EXP.1	Specified Platform Definition
APT_PSP_EXP.1	Complete Platform Specification
APT_PCT_EXP.1	Tested Platform Conformance
APT_PST_EXP.1	Comprehensive Platform Security Testing
APT_PVA_EXP.1	Comprehensive Platform Vulnerability Assessment
AVA_CCA_EXP.2	Systematic Covert Channel Analysis
AVA_VLA_EXP.4	Highly Resistant

127 The intended TOE environment and the value of information processed by this environment establish the need for the TOE to be evaluated at high robustness, which in terms of CC security assurance requirements, is higher assurance than that expressed by EAL4[8]. The set of security

[8] Refer to the "Mutual Recognition of Common Criteria Certificates" Section 1.3 to read conditions for the CC certificate to be mutually recognized for PPs with EALs higher than 4.

assurance requirements that define high robustness are summarized in Table 6.2. Note that flaw remediation (ALC_FLR) and maintenance of assurance (AMA_AMP_EXP) have also been chosen even though the CC chose not to assign these components to a specific EAL level.

Table 6.2. SKPP High Robustness Assurance Requirements Relative to EAL6

Assurance Class	Assurance Family	EAL6	SKPP
Configuration Management	ACM_AUT	**2**	2
	ACM_CAP	**5**	5
	ACM_SCP	3	3
Delivery and Operation	ADO_DEL_EXP	2	(2)
	ADO_IGS	1	1
Development (TSF)	ADV_ARC_EXP		**(1)**
	ADV_CTD_EXP		**(1)**
	ADV_FSP_EXP	3	**(4)**
	ADV_HLD_EXP	**4**	(4)
	ADV_IMP_EXP	**3**	(3)
	ADV_INI_EXP		**(1)**
	ADV_INT_EXP	**2**	**(3)**
	ADV_LLD_EXP	**2**	(2)
	ADV_LTD_EXP		**(1)**
	ADV_RCR_EXP	2	**3**
	ADV_SPM_EXP	3	3
Guidance Documents	AGD_ADM_EXP	1	(1)
	AGD_USR	1	1
Life cycle Support	ALC_DVS	**2**	2
	ALC_FLR		**3**
	ALC_LCD	2	2
	ALC_TAT	**3**	3
Maintenance of Assurance	AMA_AMP_EXP		**(1)**
Platform Assurance	APT_PDF_EXP		**(1)**
	APT_PSP_EXP		**(1)**
	APT_PCT_EXP		**(1)**
	APT_PST_EXP		**(1)**
	APT_PVA_EXP		**(1)**
Tests	ATE_COV	**3**	3
	ATE_DPT	2	**3**
	ATE_FUN	**2**	2
	ATE_IND	2	**3**
Vulnerability Assessment	AVA_CCA_EXP	**2**	(2)
	AVA_MSU	**3**	3
	AVA_SOF	1	1
	AVA_VLA_EXP	**4**	(4)

Parentheses indicate explicit or refined requirements; Bold in a column indicates an increase in assurance from the preceding EAL (i.e., bold items in the EAL6 column are increases relative to EAL5, and bold items in the SKPP column are increases relative to EAL6)

6.1 Configuration Management (ACM)

6.1.1 CM Automation (ACM_AUT)

6.1.1.1 Complete CM Automation (ACM_AUT.2)

ACM_AUT.2.1D The developer shall use a CM system.

ACM_AUT.2.2D The developer shall provide a CM plan.

ACM_AUT.2.1C The CM system shall provide an automated means by which only authorized changes are made to the TOE implementation representation, and to all other configuration items.

ACM_AUT.2.2C The CM system shall provide an automated means to support the generation of the TOE.

ACM_AUT.2.3C The CM plan shall describe the automated tools used in the CM system.

ACM_AUT.2.4C The CM plan shall describe how the automated tools are used in the CM system.

ACM_AUT.2.5C The CM system shall provide an automated means to ascertain the changes between the TOE and its preceding version.

ACM_AUT.2.6C The CM system shall provide an automated means to identify all other configuration items that are affected by the modification of a given configuration item.

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

6.1.2 CM Capabilities (ACM_CAP)

6.1.2.1 Advanced Support (ACM_CAP.5)

ACM_CAP.5.1D The developer shall provide a reference for the TOE.

ACM_CAP.5.2D The developer shall use a CM system.

ACM_CAP.5.3D The developer shall provide CM documentation.

ACM_CAP.5.1C The reference for the TOE shall be unique to each version of the TOE.

ACM_CAP.5.2C The TOE shall be labeled with its reference.

ACM_CAP.5.3C The CM documentation shall include a configuration list, a CM plan, an acceptance plan, and integration procedures.

ACM_CAP.5.4C The configuration list shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.5.5C The configuration list shall describe the configuration items that comprise the TOE.

ACM_CAP.5.6C The CM documentation shall describe the method used to uniquely identify the configuration items that comprise the TOE.

ACM_CAP.5.7C The CM system shall uniquely identify all configuration items that comprise the TOE.

ACM_CAP.5.8C The CM plan shall describe how the CM system is used.

ACM_CAP.5.9C The evidence shall demonstrate that the CM system is operating in accordance with the CM plan.

ACM_CAP.5.10C The CM documentation shall provide evidence that all configuration items have been and are being effectively maintained under the CM system.

ACM_CAP.5.11C The CM system shall provide measures such that only authorized changes are made to the configuration items.

ACM_CAP.5.12C The CM system shall support the generation of the TOE.

ACM_CAP.5.13C The acceptance plan shall describe the procedures used to accept modified or newly created configuration items as part of the TOE.

ACM_CAP.5.14C The integration procedures shall describe how the CM system is applied in the TOE manufacturing process.

ACM_CAP.5.15C The CM system shall require that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.5.16C The CM system shall clearly identify the configuration items that comprise the TSF.

ACM_CAP.5.17C The CM system shall support the audit of all modifications to the TOE, including as a minimum the originator, date, and time in the audit trail.

ACM_CAP.5.18C The CM system shall be able to identify the master copy of all material used to generate the TOE.

ACM_CAP.5.19C The CM documentation shall demonstrate that the use of the CM system, together with the development security measures, allow only authorized changes to be made to the TOE.

ACM_CAP.5.20C The CM documentation shall demonstrate that the use of the integration procedures ensures that the generation of the TOE is correctly performed in an authorized manner.

ACM_CAP.5.21C The CM documentation shall demonstrate that the CM system is sufficient to ensure that the person responsible for accepting a configuration item into CM is not the person who developed it.

ACM_CAP.5.22C The CM documentation shall justify that the acceptance procedures provide for an adequate and appropriate review of changes to all configuration items.

ACM_CAP.5.1E The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

6.1.3 CM Scope (ACM_SCP)

6.1.3.1 Development Tools CM Coverage (ACM_SCP.3)

ACM_SCP.3.1D The developer shall provide a list of configuration items for the TOE.

ACM_SCP.3.1C The list of configuration items shall include the following: implementation representation; security flaws; development tools and related information; and the evaluation evidence required by the assurance components in the ST.

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

6.2 Delivery and Operation (ADO)

6.2.1 Delivery (ADO_DEL)

6.2.1.1 **Explicit: Detection of Modification (ADO_DEL_EXP.2)**

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

Application Note: Delivery procedures must be provided for the entire TOE. It is acceptable to have different procedures that apply to parts of the TOE that are separately delivered. It is not acceptable for any part of the TOE to be delivered and for which no delivery procedures apply.

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

ADO_DEL_EXP.2.3D The developer shall use [selection: *cryptographic signature, cryptographic keyed-hash message authentication function*] technical measures to verify the integrity of the TOE or parts of it and for source authentication when delivering the TOE or parts of it to the user.

Application Note: For the case where the TOE is delivered in parts, it is the ST author's responsibility to identify the technical measures that apply to the separately delivered parts of the TOE.

Application Note: The requirements for the two technical measures given in this requirement are discussed below in ADO_DEL_EXP.2.5D and ADO_DEL_EXP.2.6D. The TOE developer can decide which of these measures is employed to meet the requirement for trusted delivery – the TOE developer is not required to employ both.

ADO_DEL_EXP.2.4D The developer shall use independent channels to deliver the TOE and to deliver the cryptographic keying materials used to verify the delivery of the TOE.

ADO_DEL_EXP.2.5D Technical measures that use cryptographic signature services shall employ Digital Signature algorithms in accordance with the NIST-approved *[selection:* Digital Signature Algorithm (DSA) with a key size (modulus) of 2048 bits or greater, RSA Digital Signature Algorithm (rDSA with odd e) with a key size (modulus) of 2048 bits or greater, Elliptic Curve Digital Signature Algorithm (ECDSA) with a key size of 256 bits or greater] that meets the following:

a) Case: Digital Signature Algorithm

FIPS PUB 186-2[9], Digital Signature Standard, for signature creation and verification processing; and ANSI Standard X9.42-2001, Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography for generation of the domain parameters[10];

b) Case: RSA Digital Signature Algorithm (with odd e)

ANSI X 9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography For The Financial Services Industry (rDSA)[11];

c) Case: Elliptic Curve Digital Signature Algorithm

ANSI X9.62-1998 (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: Elliptic Curve Digital Signature Algorithm (ECDSA)

Application Note: For elliptic curve-based schemes the key size refers to the $\log_2$ of the order of the base point. As the preferred approach for cryptographic signature, elliptic curves will eventually be required, once all the necessary standards and other supporting information are fully established.

ADO_DEL_EXP.2.6D Technical measures that use cryptographic keyed-hash message authentication functions shall employ a NIST-approved hash implementation of the Secure Hash algorithm and message digest size of at least 256 bits that meets FIPS PUB 198.

ADO_DEL_EXP.2.1C The delivery documentation shall describe all procedures that are necessary to maintain security when distributing versions of the TOE to the user.

[9] FIPS PUB 186-3 is under development. It will incorporate the signature creation and verification processing of FIPS PUB 186-2, and the generation of domain parameters of ANSI X9.42. FIPS PUB 186-3, once finalized and approved, will become the basis for this requirement.

[10] Any pseudorandom RNG used in these schemes for generating private values is to be seeded by a nondeterministic RNG.

[11] See previous footnote.

ADO_DEL_EXP.2.2C The delivery documentation shall describe how independent delivery channels are used to deliver the TOE and to deliver the cryptographic keying materials used to verify the delivery of the TOE.

ADO_DEL_EXP.2.3C The delivery documentation shall describe how the various procedures and technical measures provide for the detection of modifications, or any discrepancy between the developer's master copy and the version received at the user site.

Application Note: It is assumed that the "cryptographic seal" of the TOE code will be verified when the TOE code is received from the TOE developer and protected appropriately at the user's site prior to loading into non-volatile memory for inclusion into the hosting hardware. However, for IT environments that cannot guarantee physical protection, additional procedures to re-validate the integrity of the TOE code prior to loading should be provided by the IT environment.

ADO_DEL_EXP.2.4C The delivery documentation shall describe how the various procedures and technical measures allow detection of attempts to masquerade as the developer, even in cases in which the developer has sent nothing to the user's site.

ADO_DEL_EXP.2.5C The delivery documentation shall contain evidence demonstrating that each cryptographic signature service and each cryptographic keyed-hash message authentication function utilized is NIST approved.

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

ADO_DEL_EXP.2.2E The evaluator shall determine that the various procedures and technical measures provided result in a trusted delivery.

6.2.2 Installation, Generation and Start-Up (ADO_IGS)

6.2.2.1 Installation, Generation and Start-Up Procedures (ADO_IGS.1)

Application Note: This section is intended to address the requirements for configuring the TOE to be in a TOE Evaluated Configuration (TEC). Requirements for administrator guidance to correctly use TOE mechanisms (e.g., boot, initialization) to achieve an initial secure state are addressed in AGD_ADM.

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

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

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

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

6.3 Development (ADV)

6.3.1 Architectural Design with Domain Separation and Non-Bypassability (ADV_ARC)

6.3.1.1 Explicit: Architectural Design (ADV_ARC_EXP.1)

Application Note: This component contains two types of requirements for architecture assurance evidence: 1) requirements for specific properties and characteristics that must be present in the architecture, and 2) requirements for the evidence provided to demonstrate that the TSF architecture exhibits the properties and characteristics.

Application Note: The architecture design required by this component is at the level of the functional specification and high-level design documentation. The TSF internals description required by the ADV_INT_EXP.3 component is at the level of TSF module documentation.

ADV_ARC_EXP.1.1D The developer shall provide the architectural design of the TSF.

ADV_ARC_EXP.1.1C The descriptive information contained in the architectural design shall be at a level of detail commensurate with the description of the SFR-enforcing abstractions described in the TOE high level design documentation.

ADV_ARC_EXP.1.2C The architectural design shall demonstrate that the security domains maintained by the TSF are consistent with the SFRs**.**

Application Note: Of particular interest are SFRs FDP_IFC.2, FDP_IFF_1, FPT_SEP.3 as stated in this profile. Should the Security Target include additional relevant SFRs, they also are candidates for the stated justification.

ADV_ARC_EXP.1.3C The architectural design shall justify that the TSF protects itself from interference and tampering.

ADV_ARC_EXP.1.4C The architectural design shall justify that the TSF prevents bypass of the SFR-enforcing functionality.

ADV_ARC_EXP.1.5C The architectural design shall document the resources required by the TSF for its execution, to include providing the bounds for TSF usage requirements for processor time and memory.

Application note: Suspending a service request to implement a scheduling policy is not to be construed as nondeterministic or unpredictable TOE behavior.

ADV_ARC_EXP.1.6C The architectural design shall identify the hardware, firmware, and software portions of the TSF.

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

6.3.2 Configuration Tool Design (ADV_CTD)

6.3.2.1 **Explicit: Configuration Tool Design (ADV_CTD_EXP.1)**

ADV_CTD_EXP.1.1D The developer shall provide a configuration vector generation and validation capability.

ADV_CTD_EXP.1.2D The developer shall provide configuration vector generation and validation documentation.

ADV_CTD_EXP.1.3D The configuration vector generation and validation capability shall present configuration vectors in a human-readable form such that 1) the semantics of the vectors are clear and understandable, and 2) the completeness and accuracy of the intended operational configuration can be validated.

ADV_CTD_EXP.1.4D The configuration vector generation and validation capability shall be able to convert the configuration vectors from a human-readable form into a machine-readable form, and vice versa, such that the semantics of the data are preserved.

ADV_CTD_EXP.1.5D The configuration vector generation and validation capability shall be able to place an integrity seal on generated configuration vectors.

ADV_CTD_EXP.1.1C The presentation of the descriptive information contained in the configuration vector generation and validation documentation shall be in informal style at a level of abstraction and detail as required in the TOE high level design document.

ADV_CTD_EXP.1.2C The configuration vector generation and validation documentation shall explain the semantics for the expression of the human-readable form of a generated configuration vector such that the completeness and accuracy of a generated configuration vector can be verified.

ADV_CTD_EXP.1.3C The configuration vector generation and validation documentation shall define the format of the machine-readable form of a generated configuration vector, and shall explain how to interpret the machine-readable form of a generated configuration vector.

ADV_CTD_EXP.1.4C The configuration vector generation and validation documentation shall provide instructions for placing integrity seal on generated configuration vectors.

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

ADV_CTD_EXP.1.2E The evaluator shall determine that configuration vectors generated by the configuration vector generation and validation tool is an accurate instantiation of the intent, and shall verify that the configuration generation validation tool properly places a cryptographic seal on generated configuration vectors.

6.3.3 Functional Specification (ADV_FSP)

6.3.3.1 Explicit: Formal Functional Specification (ADV_FSP_EXP.4)

ADV_FSP_EXP.4.1D The developer shall provide a functional specification.

ADV_FSP_EXP.4.2D The developer shall provide a formal presentation of the functional specification of the TSF.

Application note: The developer is required to provide two separate documents, a Functional Specification and a precise expression of the contents of the functional specification, i.e., the Formal Presentation of the Functional Specification. The formalism of the functional specification is required to support modeling and verification requirements, i.e., to correlate the TSFI with the security policy and model.

ADV_FSP_EXP.4.1C The functional specification shall completely represent the TSF.

ADV_FSP_EXP.4.2C The functional specification shall describe the TSFI using a semi-formal style.

ADV_FSP_EXP.4.3C The functional specification shall describe the purpose and method of use for all TSFI.

ADV_FSP_EXP.4.4C The functional specification shall identify and describe all parameters associated with each TSFI.

ADV_FSP_EXP.4.5C The functional specification shall describe all operations associated with each TSFI.

ADV_FSP_EXP.4.6C The functional specification shall describe all exceptions, error messages and effects that may result from an invocation of each TSFI.

ADV_FSP_EXP.4.7C The functional specification shall describe all exceptions, error messages and effects contained in the TSF implementation that are not associated with the invocation of any TSFI.

ADV_FSP_EXP.4.8C The formal presentation of the functional specification of the TSF shall describe the TSFI using a formal style, supported by informal, explanatory text where appropriate.

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

ADV_FSP_EXP.4.2E The evaluator shall determine that the functional specification is an accurate and complete instantiation of the TOE security functional requirements.

6.3.4 High-Level Design (ADV_HLD)

6.3.4.1 Explicit: Semiformal High-Level Explanation (ADV_HLD_EXP.4)

ADV_HLD_EXP.4.1D The developer shall provide the high-level design of the TOE.

ADV_HLD_EXP.4.1C The presentation of the high-level design of the TSF shall be in semiformal style, supported by informal, explanatory text where appropriate.

ADV_HLD_EXP.4.2C The presentation of the high-level design of the runtime non-TSF portions of the TOE shall be in informal style.

Application Note: "Runtime non-TSF portions of the TOE" refers to the TOE components that are executable during runtime but are not part of the TSF. These exclude the configuration tool, the TOE initialization function and the TOE loader.

ADV_HLD_EXP.4.3C The high-level design shall be internally consistent.

ADV_HLD_EXP.4.4C The high-level design shall describe the structure of the TOE in terms of subsystems.

ADV_HLD_EXP.4.5C The high-level design shall identify all subsystems of the TSF, and designate them as either SFR-enforcing or SFR-supporting subsystems.

ADV_HLD_EXP.4.6C The high-level design shall provide a description of each subsystem of the TSF.

ADV_HLD_EXP.4.7C The high-level design shall provide a description of the interactions between the subsystems of the TSF.

Application Note: The goal of describing the interactions between the SFR-enforcing components and other components is to help provide the reader with a better understanding of how the TSF performs it functions. These interactions do not need to be characterized at the implementation level (e.g., parameters passed from one routine in a component to a routine in a different component; global variables; hardware signals (e.g., interrupts) from a hardware component to an interrupt-handling component), but the data elements identified for a particular component that are going to be used by another component should be covered in this discussion. Any control relationships between components (e.g., a component responsible for configuring a rule base for a firewall system and the component that actually implements these rules) should also be described.

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

ADV_HLD_EXP.4.2E The evaluator shall determine that the high-level design is an accurate and complete instantiation of all TOE security functional requirements.

6.3.5 Implementation Representation (ADV_IMP)

6.3.5.1 **Explicit: Structured Implementation of the TSF (ADV_IMP_EXP.3)**

Application Note: Implementation representation refers to source code for software, and Very High Speed Integrated Circuit Hardware Description Language (VHDL) or its equivalent for firmware and hardware. Implementation refers to the output of compiled source code and compiled VHDL (or its equivalent).

ADV_IMP_EXP.3.1D The developer shall make available, the implementation representation for the entire TSF.

ADV_IMP_EXP.3.2D The developer shall provide the tools and their associated instructions that are used to transform the implementation representation into the implementation.

ADV_IMP_EXP.3.1C The implementation representation shall unambiguously define the TSF to a level of detail such that the TSF can be generated without further design decisions.

ADV_IMP_EXP.3.2C The implementation representation shall be identical in form and content, as that used by the development personnel.

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

ADV_IMP_EXP.3.2E The evaluator shall determine that the implementation representation, when transformed to the implementation using the developer-provided tools and instructions, is identical to the implementation used in testing activities.

6.3.6 Trusted Initialization (ADV_INI)

6.3.6.1 **Explicit: Trusted Initialization (ADV_INI_EXP.1)**

ADV_INI_EXP.1.1D The developer shall provide a TOE initialization function.

Application Note: The TOE initialization function brings the software portion of the TSF and TSF data into the TSF security domain and establishes the TSF in a secure state consistent with the configuration vector that defines the configuration data. The following "D" elements define the security-relevant capabilities of the TOE initialization function.

ADV_INI_EXP.1.2D The TOE initialization function shall establish the TSF in a secure state consistent with the configuration vector that defines the configuration data.

Application Note: The TOE initialization function is not part of the TSF. Therefore, the assurances associated with the secure state of the TSF require an assurance argument for both the initialization function (with focus on that portion that establishes the TSF in a secure state) and the TSF; either alone is not sufficient. Refer to FPT_ESS_EXP.1 for the requirements levied on the TSF to verify that it is in a secure state when the initialization function completes.

ADV_INI_EXP.1.3D The TOE initialization function shall verify the integrity of TSF code and data prior to establishing the TSF in a secure state.

ADV_INI_EXP.1.4D The TOE initialization function shall detect and respond to errors and failures during initialization such that the TOE either successfully completes initialization or is halted.

Application Note: This requirement is intended to provide assurance that the initialization code is capable of detecting and handling anomalies during the boot process, which precedes the establishment of a secure state.

ADV_INI_EXP.1.5D The TOE initialization function shall not be able to arbitrarily interact with the TSF after TOE initialization completes.

Application Note: This requirement is intended to provide assurance that the initialization function, which is not defined as part of the TSF, is not able to interact with the TSF in a manner inconsistent with the design of the TSF, once initialization completes.

Should the design of the TOE be such that any portion of the initialization function is utilized during TOE reconfiguration or as part of trusted recovery, then it must be demonstrated that the initialization function will only execute when commanded by the TSF.

Finally, if due to design decisions, portions of the initialization function are part of the TSF, then those portions would be subjected to all TSF assurances, in addition to these initialization assurances. Note, however, that these initialization assurances are intended to be a subset of the TSF assurances.

ADV_INI_EXP.1.6D The TOE initialization function shall establish the TSF security domain and shall bring the software portion of the TSF implementation and TSF data into the TSF security domain.

ADV_INI_EXP.1.7D The TOE initialization function shall be designed and implemented such that in conjunction with the TSF no other component executing on the TOE is able to establish the TSF in a secure state consistent with the configuration vector.

ADV_INI_EXP.1.8D The TOE initialization function shall be designed and implemented such that it is able to protect itself from tampering by other components executing on the TOE.

ADV_INI_EXP.1.9D The components of the TOE initialization function shall be designed and implemented using modular decomposition.

ADV_INI_EXP.1.10D The developer shall provide a functional specification of the TOE initialization function.

ADV_INI_EXP.1.11D The developer shall provide the design of the TOE initialization function.

ADV_INI_EXP.1.12D The developer shall test the TOE initialization function and document the results.

ADV_INI_EXP.1.13D The developer shall provide TOE initialization function test documentation.

ADV_INI_EXP.1.1C The TOE initialization functional specification shall completely represent the TOE initialization function.

ADV_INI_EXP.1.2C The TOE initialization functional specification shall describe the purpose and method of use of all TOE initialization function interfaces.

ADV_INI_EXP.1.3C The TOE initialization functional specification shall describe all parameters associated with each TOE initialization function interface.

ADV_INI_EXP.1.4C The TOE initialization functional specification shall describe all operations associated with each TOE initialization function interface.

ADV_INI_EXP.1.5C The TOE initialization functional specification shall describe all exceptions, error messages and effects associated with each TOE initialization function interface.

ADV_INI_EXP.1.6C The TOE initialization design shall identify all components of the TOE initialization function and shall designate each component as relevant to establishment of the TSF in a secure state or un-related to establishment of the TSF in a secure state.

ADV_INI_EXP.1.7C The TOE initialization design shall describe the structure of the TOE initialization function in terms of the identified components.

ADV_INI_EXP.1.8C The TOE initialization design shall identify the hardware, firmware, and software portions of the TOE initialization components.

ADV_INI_EXP.1.9C The TOE initialization design shall describe how the components of the TOE initialization function work together to establish the TSF in a secure state consistent with the configuration vector.

ADV_INI_EXP.1.10C The TOE initialization design shall describe how the TOE initialization function verifies the integrity of the TSF code and data.

ADV_INI_EXP.1.11C The TOE initialization design shall describe how the TOE initialization function detects and responds to errors, and shall contain a definition and description of all errors associated with the TOE initialization function.

ADV_INI_EXP.1.12C The TOE initialization design shall demonstrate that the TOE initialization function will not arbitrarily interact with the operation of the TSF after TOE initialization completes.

ADV_INI_EXP.1.13C The TOE initialization design shall demonstrate that no other component executing on the TOE is able to establish the TSF in a secure state consistent with the configuration vector.

ADV_INI_EXP.1.14C The TOE initialization design shall demonstrate how the TOE initialization function protects itself from tampering by other components executing on the TOE.

ADV_INI_EXP.1.15C The TOE initialization design shall describe the structure of the TOE initialization function in terms of component modularization.

ADV_INI_EXP.1.16C The TOE initialization design shall justify the inclusion of components that do not support initialization of the TOE or the establishment of the TSF in a secure state.

ADV_INI_EXP.1.17C The presentation of the TOE initialization functional specification and TOE initialization design shall be in informal style.

ADV_INI_EXP.1.18C The TOE initialization test documentation shall consist of test procedure descriptions, expected test results and actual test results.

ADV_INI_EXP.1.19C The TOE initialization test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing the TOE initialization function.

ADV_INI_EXP.1.20C The TOE initialization test results shall demonstrate that the TOE initialization function behaves as specified.

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

ADV_INI_EXP.1.2E The evaluator shall determine that the TOE initialization function design is sufficient to ensure that the TOE initialization function: (a) correctly establishes the TSF in a secure state while preserving the integrity of TSF data and code, and (b) halts the TOE if anomalies prevent establishment of the TSF in a secure state.

ADV_INI_EXP.1.3E The evaluator shall execute all tests in the TOE initialization test documentation to verify the developer test results.

ADV_INI_EXP.1.4E The evaluator shall conduct independent tests of the TOE initialization function to confirm that the TOE initialization function behaves as specified.

ADV_INI_EXP.1.5E The evaluator shall determine that other components executing on the TOE can neither circumvent nor tamper with the TOE initialization function.

6.3.7 TSF Internals (ADV_INT)

6.3.7.1 **Explicit: Minimization of Complexity (ADV_INT_EXP.3)**

ADV_INT_EXP.3.1D The developer shall design and implement the TSF using modular decomposition.

ADV_INT_EXP.3.2D The developer shall use sound software engineering principles to achieve the modular decomposition of the TSF.

ADV_INT_EXP.3.3D The developer shall design the TSF modules such that they exhibit good internal structure and are not overly complex, with limited exceptions.

ADV_INT_EXP.3.4D The developer shall design all TSF modules such that they exhibit only functional, sequential, communicational, or temporal cohesion, with limited exceptions.

ADV_INT_EXP.3.5D The developer shall design all TSF modules such that they exhibit only call or common coupling, with limited exceptions.

ADV_INT_EXP.3.6D The developer shall implement the TSF modules using coding standards that result in good internal structure that is not overly complex.

ADV_INT_EXP.3.7D The developer shall design and implement the TSF in a layered fashion that minimizes interactions between the layers of the design.

ADV_INT_EXP.3.8D The developer shall design and implement the TSF such that interactions between layers are initiated from a higher layer in the hierarchy down to the next layer in the hierarchy, with limited exceptions.

ADV_INT_EXP.3.9D The developer shall design and implement the modules of the TSF such that they are simple enough to be analyzed.

ADV_INT_EXP.3.10D The developer shall ensure that functions whose purpose is not relevant for enforcing or supporting the SFRs are excluded from the TSF modules.

ADV_INT_EXP.3.11D The developer shall design and implement the TSF in such a way that the principle of least privilege is achieved with respect to TSF modules as required by FPT_PLP_EXP.

ADV_INT_EXP.3.12D The developer shall provide a TSF internals description.

ADV_INT_EXP.3.1C The TSF internals description shall describe the process used for modular decomposition.

ADV_INT_EXP.3.2C The TSF internals description shall identify all the modules of the TSF.

ADV_INT_EXP.3.3C The TSF internals description shall describe how the TSF design is a reflection of the modular decomposition process.

ADV_INT_EXP.3.4C The TSF internals description shall provide a justification, on a per-module basis, of any deviation from the coding standards governing module internal structure and complexity.

ADV_INT_EXP.3.5C The TSF internals description shall include a coupling analysis that describes intermodule coupling for all TSF modules.

ADV_INT_EXP.3.6C The TSF internals description shall include a cohesion analysis that describes the types of cohesion for all TSF modules.

ADV_INT_EXP.3.7C The TSF internals description shall provide a justification, on a per module basis, for any coupling or cohesion exhibited by modules of the TSF, other than those permitted.

ADV_INT_EXP.3.8C The TSF internals description shall describe the layering architecture and shall describe the services that each layer provides.

ADV_INT_EXP.3.9C The TSF internals description shall describe the methodology used to determine the layering architecture.

ADV_INT_EXP.3.10C The TSF internals description shall identify all modules associated with each layer of the TSF.

ADV_INT_EXP.3.11C The TSF internals description shall describe all interactions between layers of the TSF.

ADV_INT_EXP.3.12C The TSF internals description shall provide a justification of interactions that are initiated from a lower layer to a higher layer.

ADV_INT_EXP.3.13C The TSF internals description shall provide a justification for all modules of the TSF that contain unused or redundant code.

ADV_INT_EXP.3.14C The TSF internals description shall describe how the entire TSF has been designed and implemented to minimize complexity.

ADV_INT_EXP.3.15C The TSF internals description shall justify the inclusion of any non-security relevant modules in the TSF.

ADV_INT_EXP.3.16C The TSF internals description shall describe how the entire TSF has been designed and implemented to achieve the principle of least privilege.

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

ADV_INT_EXP.3.2E The evaluator shall verify, through direct examination of a sample of TSF modules, that cohesion and coupling between TSF modules is consistent with the TSF internals description.

ADV_INT_EXP.3.3E The evaluator shall verify, through direct examination of a sample of TSF modules, that the design and implementation of the TSF modules is consistent with the TSF internals description about minimization of complexity.

ADV_INT_EXP.3.4E The evaluator shall determine that the TSF modules design and implementation is sufficient to support the principle of least privilege.

ADV_INT_EXP.3.5E The evaluator shall confirm that the modules of the TSF are simple enough to be analyzed.

6.3.8 Low-level Design (ADV_LLD)

6.3.8.1 **Explicit: Semi-Formal Low-Level Design (ADV_LLD_EXP.2)**

ADV_LLD_EXP.2.1D The developer shall provide the low-level design of the TSF.

ADV_LLD_EXP.2.1C The presentation of the low-level design shall be semi-formal style, supported by informal, explanatory text where appropriate.

ADV_LLD_EXP.2.2C The low-level design shall be internally consistent.

ADV_LLD_EXP.2.3C The low-level design shall describe the TSF in terms of modules, identifying each TSF module and designating each TSF module as SFR-enforcing, SFR-supporting, or non-security relevant.

ADV_LLD_EXP.2.4C The low-level design shall identify and describe data that are common to more than one module.

ADV_LLD_EXP.2.5C The low-level design shall describe each module in terms of its purpose, method of use, interfaces provided to invoke the module, return values from those interfaces, and methods used to invoke and dependencies on other modules.

Application Note: "Methods used to invoke other modules" includes all forms of direct interaction with other modules. This includes called interfaces, interaction via a message mailbox or other inter-module communication method, etc.

ADV_LLD_EXP.2.6C The low-level design shall describe each module in terms of all exceptions, error messages and effects that may result from the execution of the module.

ADV_LLD_EXP.2.7C The low-level design shall provide an algorithmic description for each module detailed enough to represent the TSF implementation.

Application Note: An algorithmic description contains sufficient detail such that two different programmers would produce functionally-equivalent code, although data structures, programming methods, etc. may differ.

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

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

6.3.9 Load Tool Design (ADV_LTD)

6.3.9.1 **Explicit: Load Tool Design (ADV_LTD_EXP.1)**

ADV_LTD_EXP.1.1D The developer shall provide a TOE loader design.

ADV_LTD_EXP.1.2D The developer shall provide a TOE loader capability.

ADV_LTD_EXP.1.3D The TOE loader capability shall be able to transfer the machine-readable software portion of the TSF implementation and configuration vector set, either together or separately, into a form that is accessible by the TOE initialization function.

Application note: See Load Function in Glossary of Terms.

ADV_LTD_EXP.1.4D The TOE loader capability shall preserve the integrity of the software portion of the TSF implementation and configuration vector set during the transfer process.

ADV_LTD_EXP.1.1C The presentation of the descriptive information contained in the TOE loader design shall be in informal style at a level of abstraction and detail as required in the TOE high level design document.

ADV_LTD_EXP.1.2C The TOE loader design shall describe how the TOE loader capability performs the transfer of the machine-readable software portion of the TSF implementation and configuration vector set into a form that is accessible by the TOE initialization function.

ADV_LTD_EXP.1.23 The TOE loader design shall describe the protection mechanisms used by the TOE loader capability such that it is able to preserve the integrity of the TSF implementation and configuration vector set during all aspects of the transfer process.

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

ADV_LTD_EXP.1.2E The evaluator shall determine that the TOE loader design provides sufficient evidence to support the conclusion that the TOE loader capability preserves the integrity of the machine-readable portions of the TSF implementation and configuration vector set when they are transferred into a form accessible by the TOE initialization function.

6.3.10 Representation Correspondence (ADV_RCR)

6.3.10.1 Formal Correspondence Demonstration (ADV_RCR.3)

ADV_RCR.3.1D The developer shall provide an analysis of correspondence between all adjacent pairs of TSF representations that are provided.

ADV_RCR.3.2D For those corresponding portions of representations that are formally specified, the developer shall prove that correspondence.

ADV_RCR.3.1C For each adjacent pair of provided TSF representations, the analysis shall prove or demonstrate that all relevant security functionality of the more abstract TSF representation is correctly and completely refined in the less abstract TSF representation.

ADV_RCR.3.2C For each adjacent pair of provided TSF representations, where portions of one representation are semiformally specified and the other at least semiformally specified, the demonstration of correspondence between those portions of the representations shall be semiformal.

ADV_RCR.3.3C For each adjacent pair of provided TSF representations, where portions of both representations are formally specified, the proof of correspondence between those portions of the representations shall be formal.

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

ADV_RCR.3.2E The evaluator shall determine the accuracy of the proofs of correspondence by selectively verifying the formal analysis.

6.3.11 Security Policy Modeling (ADV_SPM)

6.3.11.1 Formal TOE Security Policy Model (ADV_SPM.3)

ADV_SPM.3.1D The developer shall provide a TSP model.

ADV_SPM.3.2D **Refinement**: The developer shall demonstrate **correspondence between the functional specification and the TSP model and shall** prove correspondence between the **formal presentation of the** functional specification and the TSP model. **1**

ADV_SPM.3.1C The TSP model shall be formal.

ADV_SPM.3.2C The TSP model shall describe the rules and characteristics of all policies of the TSP that can be modeled.

ADV_SPM.3.3C The TSP model shall include a rationale that demonstrates that it is consistent and complete with respect to all policies of the TSP that can be modeled.

ADV_SPM.3.4C The demonstration of correspondence between the TSP model and the functional specification shall show that all of the security functions in the functional specification are consistent and complete with respect to the TSP model.

ADV_SPM.3.5C **Refinement:** The demonstration of correspondence between the TSP model and the functional specification shall be semiformal. **2**

ADV_SPM.3.6C **Refinement:** The proof of correspondence between the TSP model and the **formal presentation of the** functional specification shall be formal. **3**

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

6.4 Guidance Documents (AGD)

6.4.1 Administrator Guidance (AGD_ADM)

6.4.1.1 **Explicit: Administrator Guidance (AGD_ADM_EXP.1)**

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

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

Application Note: Administrators of the TOE are the "authorized administrators" as defined in the Glossary.

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

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

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

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

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

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

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

AGD_ADM_EXP.1.9C The administrator guidance shall document procedures necessary for the correct generation and validation of the TSF configuration vectors.

AGD_ADM_EXP.1.10C The administrator guidance shall document procedures to restrict the authorizations and information flows granted to each subject to be only those required for its assigned functionality.

AGD_ADM_EXP.1.11C The administrator guidance shall describe the Partitioned Information Flow Policy abstractions supported by the TOE, and shall document constraints and procedures for assigning the correct abstractions to partitions, and the allocation of subjects and exported resources to partitions based upon the abstractions supported by partitions.

Application Note: The SKPP defines two forms of abstraction for the Partitioned Information Flow Policy: 1) Partition, 2) Least Privilege. Refer to Section 2 for discussion of "Partitions and the Partitioned Information Flow Policy (PIFP)".

AGD_ADM_EXP.1.12C The administrator guidance shall document procedures necessary to securely load the TSF code and configuration vectors.

Application Note: See Load function in Figure 2-1.

AGD_ADM_EXP.1.13C The administrator guidance shall document procedures necessary for using the initialization function to bring the TSF into an initial secure state.

AGD_ADM_EXP.1.14C The administrator guidance shall describe the audit record structure in sufficient detail such that the audit data can be properly interpreted.

AGD_ADM_EXP.1.15C The administrator guidance shall document the time stamp definition and metric, and the means to interpret the chosen time stamp format.

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

6.4.2 User Guidance (AGD_USR)

6.4.2.1 User Guidance (AGD_USR.1)

AGD_USR.1.1D The developer shall provide user guidance.

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

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

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

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

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

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

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

6.5 Life Cycle Support (ALC)

6.5.1 Development Security (ALC_DVS)

6.5.1.1 Sufficiency of Security Measures (ALC_DVS.2)

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

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

Application Note: Protecting "the confidentiality and integrity of the TOE design and implementation" means that there are confidentiality and integrity rules in place and the rules are enforced. The level of protection is TOE-specific and it is acceptable to have confidentiality rules that do not require confidentiality protection, i.e., "negative" rules.

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

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

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

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

6.5.2 Flaw Remediation (ALC_FLR)

6.5.2.1 Systematic Flaw Remediation (ALC_FLR.3)

ALC_FLR.3.1D The developer shall provide flaw remediation procedures addressed to TOE developers.

ALC_FLR.3.2D The developer shall establish a procedure for accepting and acting upon all reports of security flaws and requests for corrections to those flaws.

ALC_FLR.3.3D The developer shall provide flaw remediation guidance addressed to TOE users.

ALC_FLR.3.1C The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

ALC_FLR.3.2C The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

ALC_FLR.3.3C The flaw remediation procedures shall require that corrective actions be identified for each of the security flaws.

ALC_FLR.3.4C The flaw remediation procedures documentation shall describe the methods used to provide flaw information, corrections and guidance on corrective actions to TOE users.

ALC_FLR.3.5C The flaw remediation procedures shall describe a means by which the developer receives from TOE users reports and enquiries of suspected security flaws in the TOE.

ALC_FLR.3.6C The procedures for processing reported security flaws shall ensure that any reported flaws are corrected and the correction issued to TOE users.

ALC_FLR.3.7C The procedures for processing reported security flaws shall provide safeguards that any corrections to these security flaws do not introduce any new flaws.

ALC_FLR.3.8C The flaw remediation guidance shall describe a means by which TOE users report to the developer any suspected security flaws in the TOE.

ALC_FLR.3.9C The flaw remediation procedures shall include a procedure requiring timely responses for the automatic distribution of security flaw reports and the associated corrections to registered users who might be affected by the security flaw.

ALC_FLR.3.10C The flaw remediation guidance shall describe a means by which TOE users may register with the developer, to be eligible to receive security flaw reports and corrections.

ALC_FLR.3.11C The flaw remediation guidance shall identify the specific points of contact for all reports and enquiries about security issues involving the TOE.

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

6.5.3 Life Cycle Definition (ALC_LCD)

6.5.3.1 Standardized Life-Cycle Model (ALC_LCD.2)

ALC_LCD.2.1D The developer shall establish a life-cycle model to be used in the development and maintenance of the TOE.

ALC_LCD.2.2D The developer shall provide life-cycle definition documentation.

ALC_LCD.2.3D The developer shall use a standardized life-cycle model to develop and maintain the TOE.

ALC_LCD.2.1C The life-cycle definition documentation shall describe the model used to develop and maintain the TOE.

ALC_LCD.2.2C The life-cycle model shall provide for the necessary control over the development and maintenance of the TOE.

ALC_LCD.2.3C The life-cycle definition documentation shall explain why the model was chosen.

ALC_LCD.2.4C The life-cycle definition documentation shall explain how the model is used to develop and maintain the TOE.

ALC_LCD.2.5C The life-cycle definition documentation shall demonstrate compliance with the standardized life-cycle model.

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

6.5.4 Tools and Techniques (ALC_TAT)

6.5.4.1 Compliance with Implementation Standards – All Parts (ALC_TAT.3)

ALC_TAT.3.1D The developer shall identify the development tools being used for the TOE.

ALC_TAT.3.2D The developer shall document the selected implementation-dependent options of the development tools.

ALC_TAT.3.3D The developer shall describe the implementation standards for all parts of the TOE.

ALC_TAT.3.1C All development tools used for implementation shall be well-defined.

Application Note: The development tools include the compiler and linker used to generate the TOE.

ALC_TAT.3.2C The documentation of the development tools shall unambiguously define the meaning of all statements used in the implementation.

ALC_TAT.3.3C The documentation of the development tools shall unambiguously define the meaning of all implementation-dependent options.

Application Note: This documentation includes the compiler options used during the generation of the TOE.

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

ALC_TAT.3.2E The evaluator shall confirm that the implementation standards have been applied.

6.6 Ratings Maintenance (AMA)

6.6.1 Assurance Maintenance Plan (AMA_AMP)

6.6.1.1 **Explicit: Assurance Maintenance Plan (AMA_AMP_EXP.1)**

Application Note: Target of Maintenance (TOM) is defined as the subject of the assurance maintenance process, comprising an evaluated TOE together with any changes to the associated assurance baseline.

AMA_AMP_EXP.1.1D The developer shall provide an Assurance Maintenance Plan.

AMA_AMP_EXP.1.1C The Assurance Maintenance Plan shall identify the assurance baseline.

AMA_AMP_EXP.1.2C The Assurance Maintenance Plan shall characterize the changes to the assurance baseline that are covered by the plan.

AMA_AMP_EXP.1.3C The Assurance Maintenance Plan shall describe the planned TOM release-cycle.

AMA_AMP_EXP.1.4C The Assurance Maintenance Plan shall identify the planned schedule of assurance maintenance audits and the conditions for the end of maintenance.

Application Note: The end of maintenance occurs when it is no longer feasible or practical to maintain the TOM under maintenance assurance [8].

AMA_AMP_EXP.1.5C The Assurance Maintenance Plan shall justify the planned schedule of assurance maintenance audits and the conditions for the end of maintenance.

AMA_AMP_EXP.1.6C The Assurance Maintenance Plan shall identify the processes for assigning and ensuring currency of knowledge of individual(s) assuming the role of security analyst.

AMA_AMP_EXP.1.7C The Assurance Maintenance Plan shall define the relationship between the security analyst and the development of the evidence.

AMA_AMP_EXP.1.8C The Assurance Maintenance Plan shall identify the conceptual, technical, and evaluation qualifications of the individual(s) identified as the security analyst.

Application Note: In AMA_AMP_EXP.1.8C, conceptual qualification refers to the security analyst's understanding of security concepts relevant to the TOM.

AMA_AMP_EXP.1.9C The Assurance Maintenance Plan shall describe the procedure specifying the method by which changes to the assurance baseline will be identified.

AMA_AMP_EXP.1.10C The Assurance Maintenance Plan shall describe the procedures to be applied to the TOM to maintain the assurance established for the certified TOE.

AMA_AMP_EXP.1.11C The Assurance Maintenance Plan shall describe the controls and mechanisms implemented to ensure that the procedures documented in the Assurance Maintenance Plan are followed.

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

6.7 Platform Assurance (APT)

Application Note: The requirements in this class are intended to be applied to commercial-off-the-shelf (COTS), mass-produced, non-specialized, third-party platform components. These requirements replace a subset of the assurance requirements defined in a number of ADV, ATE and AVA families. Assurance requirements defined in other classes and families that are not addressed in this APT class still apply.

6.7.1 Platform Definition (APT_PDF)

6.7.1.1 Explicit: Specified Platform Definition (APT_PDF_EXP.1)

APT_PDF_EXP.1.1D The developer shall supply platform definition documentation.

Application Note: The platform definition documentation does not need to be a single document. In addition to TOE-specific documentation such as the platform component security analysis, the platform definition documentation may also include separate vendor documentation for the different components that comprise the TOE platform. It is the responsibility of the TOE developer to ensure that vendor documentation, if used, can satisfy the content requirements defined in this family.

APT_PDF_EXP.1.2D The developer shall provide the platform definition documentation to potential end-users of the product under terms no more restrictive than the security target.

APT_PDF_EXP.1.1C The platform definition documentation shall identify the types of commercial-of-the-shelf, mass-produced, non-specialized, third party platform components that comprise the platform for the TOE.

APT_PDF_EXP.1.2C The platform definition documentation shall specify the rules for assembling platform components into a valid platform for the TOE.

APT_PDF_EXP.1.3C The platform definition documentation shall include a platform component security analysis for each type of platform component to indicate the capabilities of the component and how the component capabilities interact with the TOE.

APT_PDF_EXP.1.4C The platform definition documentation shall identify component interface specifications provided by platform component manufacturers for the external platform interfaces and the internal platform interfaces, including interfaces between platform components that define the interface and behavior of each valid platform component.

APT_PDF_EXP.1.5C The platform component security analysis shall characterize each platform component type in terms of allowable variations in functional parameters.

APT_PDF_EXP.1.6C The platform component security analysis shall describe the effect of the full range of allowed functional parameter variations on the TOE.

APT_PDF_EXP.1.7C The platform component security analysis shall identify any platform components that are directly responsible for implementing any part of any SFR.

APT_PDF_EXP.1.8C The references to each component interface shall be sufficiently precise to allow the specifications to be obtained by a third party.

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

APT_PDF_EXP.1.2E The evaluator shall verify that the platform definition identifies all types of platform components that may be required to construct a valid platform for the TOE.

APT_PDF_EXP.1.3E The evaluator shall verify that the rules for platform assembly allow construction of valid platforms for the TOE.

APT_PDF_EXP.1.4E The evaluator shall verify that the platform configuration(s) used for testing are constructed in accordance with the platform definition.

APT_PDF_EXP.1.5E The evaluator shall confirm that all relevant SFRs are addressed in the platform component security analysis for each platform component type.

APT_PDF_EXP.1.6E The evaluator shall confirm that all security mechanisms implemented in platform components that are depended on by the software portion of the TOE are correctly identified in the applicable ADV_HLD and ADV_LLD documentation.

APT_PDF_EXP.1.7E The evaluator shall confirm that component interface specifications are identified for all platform components.

APT_PDF_EXP.1.8E The evaluator shall select a subset of the component interface specifications and shall verify that they provide adequate information to support design and testing of component compatibility.

6.7.2 Platform Specification (APT_PSP)

6.7.2.1 **Explicit: Complete Platform Specification (APT_PSP_EXP.1)**

APT_PSP_EXP.1.1D The developer shall identify the specifications for all external platform interfaces.

APT_PSP_EXP.1.2D The developer shall supply a complete specification of all external platform interfaces.

APT_PSP_EXP.1.3D The developer shall supply a complete specification of all internal platform interfaces.

APT_PSP_EXP.1.4D The developer shall supply a complete specification of all platform component interfaces that are not external and are not used by the TOE.

APT_PSP_EXP.1.1C The external platform interface specification shall identify invocation methods, parameters, expected results, and error conditions for all external platform interfaces.

APT_PSP_EXP.1.2C The external platform interface specification shall provide an argument that all external platform interfaces are included in the specification.

APT_PSP_EXP.1.3C The internal platform interface specification shall identify invocation methods, parameters, expected results, and error conditions for all internal platform interfaces.

APT_PSP_EXP.1.4C The internal platform interface specification shall provide an argument that all internal platform interfaces are included in the specification.

APT_PSP_EXP.1.5C The internal platform interface specification shall provide an argument that all platform component interfaces that are not external and are not used by the TOE are included in the specification.

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

6.7.3 Platform Conformance Testing (APT_PCT)

6.7.3.1 Explicit: Tested Platform Conformance (APT_PCT_EXP.1)

APT_PCT_EXP.1.1D For each type of commercial-of-the-shelf, mass-produced, non-specialized, third party platform component, the developer shall describe acceptance test procedures that demonstrate that a particular platform component is compatible with the platform definition.

APT_PCT_EXP.1.1C The acceptance test procedures shall verify that the particular platform component operates successfully when used as a component of the TOE.

Application Note: Vendor-provided tests and test procedures may be used to meet this requirement.

APT_PCT_EXP.1.2C The acceptance test procedures shall explicitly test all platform security features on which the TSF depends, as identified in the platform component security analysis.

Application Note: Since the platform component security analysis is TOE-specific, the tests to verify the platform security features are expected to be developed by the TOE developer.

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

APT_PCT_EXP.1.2E The evaluator shall verify that the acceptance test procedure has been successfully followed for the platform components used in the TOE configuration(s) that are tested.

6.7.4 Platform Security Testing (APT_PST)

6.7.4.1 Explicit: Comprehensive Platform Security Testing (APT_PST_EXP.1)

APT_PST_EXP.1.1D The developer shall supply tests to verify correct operation of all external platform interfaces, and those internal platform interfaces used by the TOE.

APT_PST_EXP.1.1C The platform security tests shall define the test procedures and expected results for each tested interface.

APT_PST_EXP.1.2C The platform security tests shall include an argument that the test coverage of applicable platform interfaces is complete.

APT_PST_EXP.1.3C The platform security tests shall verify correct security operation of at least one instance of each interface and/or interface parameter that is manipulable by an untrusted subject.

APT_PST_EXP.1.4C The platform security tests for the internal platform interfaces used by the TOE shall verify correct security operation of the platform component feature(s) that implement those feature(s).

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

APT_PST_EXP.1.2E The evaluator shall observe execution of the platform security tests and verify that the correct test results are obtained.

APT_PST_EXP.1.3E The evaluator shall confirm that the claimed test coverage for internal platform interfaces used by the TOE is complete with respect to usage of interfaces described in the applicable ADV_HLD and ADV_LLD documentation.

6.7.5 Platform Vulnerability Assessment (APT_PVA)

6.7.5.1 Explicit: Comprehensive Platform Vulnerability Assessment (APT_PVA_EXP.1)

APT_PVA_EXP.1.1D The developer shall consider all external platform interfaces, and those internal platform interfaces used by the TOE in performing the vulnerability assessment as specified in AVA_VLA_EXP.4.

APT_PVA_EXP.1.2D The developer shall provide platform vulnerability assessment documentation.

APT_PVA_EXP.1.1C The platform vulnerability assessment documentation shall describe the disposition of considered vulnerabilities.

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

APT_PVA_EXP.1.2E The evaluator shall consider all external platform interfaces, and those internal platform interfaces used by the TOE in performing the vulnerability assessment.

6.8 Testing (ATE)

6.8.1 Coverage (ATE_COV)

6.8.1.1 Rigorous Analysis of Coverage (ATE_COV.3)

ATE_COV.3.1D The developer shall provide an analysis of the test coverage.

ATE_COV.3.1C The analysis of the test coverage shall demonstrate the correspondence between the tests identified in the test documentation and the TSF as described in the functional specification.

ATE_COV.3.2C The analysis of the test coverage shall demonstrate that the correspondence between the TSF as described in the functional specification and the tests identified in the test documentation is complete.

ATE_COV.3.3C The analysis of the test coverage shall rigorously demonstrate that all external interfaces of the TSF identified in the functional specification have been completely tested.

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

6.8.2 Depth (ATE_DPT)

6.8.2.1 Testing: Implementation Representation (ATE_DPT.3)

ATE_DPT.3.1D The developer shall provide the analysis of the depth of testing.

ATE_DPT.3.1C The depth analysis shall demonstrate that the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with its high-level design, low-level design and implementation presentation.

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

6.8.3 Functional Tests (ATE_FUN)

6.8.3.1 Ordered Functional Testing (ATE_FUN.2)

ATE_FUN.2.1D The developer shall test the TSF and document the results.

ATE_FUN.2.2D The developer shall provide test documentation.

ATE_FUN.2.1C The test documentation shall consist of test plans, test procedure descriptions, expected test results and actual test results.

ATE_FUN.2.2C The test plans shall identify the security functions to be tested and describe the goal of the tests to be performed.

ATE_FUN.2.3C The test procedure descriptions shall identify the tests to be performed and describe the scenarios for testing each security function. These scenarios shall include any ordering dependencies on the results of other tests.

ATE_FUN.2.4C The expected test results shall show the anticipated outputs from a successful execution of the tests.

ATE_FUN.2.5C The test results from the developer execution of the tests shall demonstrate that each tested security function behaved as specified.

ATE_FUN.2.6C The test documentation shall include an analysis of the test procedure ordering dependencies.

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

6.8.4 Independent Testing (ATE_IND)

6.8.4.1 Independent Testing – Complete (ATE_IND.3)

ATE_IND.3.1D The developer shall provide the TOE for testing.

ATE_IND.3.1C The TOE shall be suitable for testing.

ATE_IND.3.2C The developer shall provide an equivalent set of resources to those that were used in the developer's functional testing of the TSF.

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

ATE_IND.3.2E The evaluator shall test a subset of the TSF as appropriate to confirm that the TOE operates as specified.

ATE_IND.3.3E The evaluator shall execute all tests in the test documentation to verify the developer test results.

6.9 Vulnerability Assessment (AVA)

6.9.1 Covert Channel Analysis (AVA_CCA)

6.9.1.1 **Explicit: Systematic Covert Channel Analysis (AVA_CCA_EXP.2)**

AVA_CCA_EXP.2.1D The developer shall conduct a search for inter-partition covert channels with respect to the Partitioned Information Flow Policy.

AVA_CCA_EXP.2.2D The developer shall provide covert channel analysis documentation.

AVA_CCA_EXP.2.1C The analysis documentation shall identify covert channels and estimate their capacity.

AVA_CCA_EXP.2.2C The analysis documentation shall describe the procedures used for determining the existence of covert channels, and the information needed to carry out the covert channel analysis.

AVA_CCA_EXP.2.3C The analysis documentation shall describe all assumptions made during the covert channel analysis.

AVA_CCA_EXP.2.4C The analysis documentation shall describe the method used for estimating channel capacity, based on worst case scenarios.

AVA_CCA_EXP.2.5C The analysis documentation shall describe the worst case exploitation scenario for each identified covert channel.

AVA_CCA_EXP.2.6C The analysis documentation shall provide evidence that the method used to identify covert channels is systematic.

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

AVA_CCA_EXP.2.2E The NSA evaluator shall confirm that the results of the covert channel analysis show that the TOE meets its functional requirements.

AVA_CCA_EXP.2.3E The NSA evaluator shall selectively validate the covert channel analysis through testing.

6.9.2 Misuse (AVA_MSU)

6.9.2.1 Analysis and Testing for Insecure States (AVA_MSU.3)

AVA_MSU.3.1D The developer shall provide guidance documentation.

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

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

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

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

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

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

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

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

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

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

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

6.9.3 Strength of TOE Security Functions (AVA_SOF)

6.9.3.1 Strength of TOE Security Function Evaluation (AVA_SOF.1)

Application Note: This PP contains no security functions for which a strength of function claim is appropriate. However, should additional security functions for which a strength of function claim is appropriate be included in a security target claiming conformance to this PP, then the following AVA_SOF criteria applies.

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

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

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

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

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

6.9.4 Vulnerability Analysis (AVA_VLA)

6.9.4.1 **Explicit: Highly Resistant (AVA_VLA_EXP.4)**

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

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

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

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

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

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

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

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

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

AVA_VLA_EXP.4.2E The NSA evaluator shall perform an independent vulnerability analysis.

AVA_VLA_EXP.4.3E The NSA evaluator shall perform independent penetration testing.

AVA_VLA_EXP.4.4E The NSA evaluator shall determine that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.

End Notes

This section records the assurance requirements where deletions of Common Criteria text were performed.

The modifications described in the following End Notes all relate to the functional specification of the TSF and its formal presentation. Refer to the Application Notes of ADV_FSP_EXP.4.2D for an explanation of the relationship between these two evaluation artifacts.

1 Modifications of CC text were performed in ADV_SPM.3.2D. Rationale: 1) The words “correspondence between the functional specification and the TSP model and shall” were added, 2) the words “or prove, as appropriate,” were changed to “prove” and 3) the words “formal presentation of the” were added. The changes are to clarify that two separate correspondences are required between the functional specification and the TSP model, and between the formal presentation of the functional specification and the TSP model.

ADV_SPM.3.2D **Refinement:** The developer shall demonstrate **correspondence between the functional specification and the TSP model and shall** ~~or~~ prove~~, as appropriate,~~ correspondence between the **formal presentation of the** functional specification and the TSP model.

2 Modifications of CC text were performed in ADV_SPM.3.5C. Rationale: The words "Where the functional specification is semiformal," were deleted for readability since ADV_FSP_EXP.4.2C requires that the functional specification be written in a semi-formal style.

ADV_SPM.3.5C **Refinement:** ~~Where the functional specification is semiformal,~~ The demonstration of correspondence between the TSP model and the functional specification shall be semiformal.

3 Modifications of CC text were performed in ADV_SPM.3.6C. Rationale: The words "Where the functional specification is formal," were deleted and 2) the words "formal presentation of the" were added. The changes are for readability since ADV_FSP_EXP.4.2D and ADV_FSP_EXP.4.9C require a formal presentation of the functional specification.

ADV_SPM.3.6C **Refinement:** ~~Where the functional specification is formal,~~ The proof of correspondence between the TSP model and the **formal presentation of the** functional specification shall be formal.

7. Rationale

128 This section provides the rationale for the selection, creation, and use of security objectives and requirements as defined in sections 4 and 5, respectively.

7.1 Security Objectives derived from Threats

129 Each of the identified threats to security is addressed by one or more security objectives. Table 7.1 below provides the mapping from security objectives to threats, as well as a rationale that discusses how the threat is addressed. Definitions are provided (in italics) below each threat and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.1. Mapping of Security Objectives to Threats

Threat	Objectives Addressing Threat	Rationale
T.ADMIN_ERROR *An administrator may incorrectly install or configure the TOE (including the misapplication of the protections afforded by the PIFP), or install a corrupted TOE resulting in ineffective security mechanisms.*	O.ADMIN_GUIDANCE *The TOE will provide administrators with the necessary information for secure management of the TOE.* O.INSTALL_GUIDANCE *The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.*	To mitigate this threat, administrative personnel must have available to them correct guidance governing the installation and use of the TOE. O.ADMIN_GUIDANCE requires that the necessary information to securely manage the TOE be provided to administrators of the TOE. O.INSTALL_GUIDANCE requires that the appropriate information to securely install and maintain TOE security be provided as part of the delivered TOE. This threat is about TOE misconfiguration by a qualified admininstrator. TOE misconfiguration by an unqualified administrator is addressed separately by the environmental assumption A.SUBJECT_ALLOCATION (see Section 7.3).
T.ALTERED_DELIVERY *The TOE may be corrupted or otherwise modified during delivery such that the on-site version does not match the master distribution version.*	O.TRUSTED_DELIVERY *The integrity of the TOE must be protected during the initial delivery and subsequent updates, and verified to ensure that the on-site version matches the master distribution version.*	To mitigate this threat, O.TRUSTED_DELIVERY requires integrity protection of the TOE. Checking the integrity of the TOE during initial delivery and subsequent updates is sufficient to determine if the TOE is corrupted or modified.

Threat	Objectives Addressing Threat	Rationale
T.CONFIGURATION_CHANGE *The lack of TSF-enforced constraints on the ability of an authorized subject to invoke or dictate how the TOE is reconfigured may result in the TOE transitioning to an insecure (unknown, inconsistent, etc) state.*	O.CONFIGURATION_CHANGE *The TOE will support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.* O.MANAGE *The TOE will provide all the functions necessary to support the administrative users and authorized subjects in their management of the TOE security functions and configuration data, and restrict these functions from use by unauthorized subjects.*	This threat exists in relation to an organizational security policy for configuration change (see P.CONFIGURATION_CHANGE). For a TOE that supports online configuration change capability, there must be protections in place to ensure that the change in configuration results in the configuration specified, and that secure state is maintained for the duration of the configuration change process. O.CONFIGURATION_CHANGE mitigates this threat by requiring the TOE to provide specific means for authorized subjects to select or redefine the TOE configuration. O.MANAGE mitigates this threat by requiring the TOE to provide management functions accessible to authorized subjects and restricted from access by unauthorized subjects.
T.CONFIGURATION_INTEGRITY *The TOE may be placed in a configuration that is not consistent with that of the configuration vector due to the improper loading of the configuration vector or incorrect use of the configuration vector during TOE initialization.*	O.CORRECT_INIT *The TOE will provide mechanisms to correctly transfer the software portion of the TSF implementation and TSF data into the TSF's security domain and to correctly establish the TOE in an operational configuration consistent with the configuration vector that defines the configuration data.* O.CORRECT_LOAD *The TOE will provide procedures and mechanisms to correctly convert the software portion of the TSF implementation and/or configuration vectors into a TOE-usable form.*	The failure to establish the TSF in the intended configuration that is consistent with the intent may occur if the TOE load and initialization functions do not have sufficient checks-and-balances to maintain the integrity of the configuration vector during the TOE load process and during TOE initialization. In addition, it is necessary to ensure correct interpretation and use of the configuration vector during the initialization of the TOE. O.CORRECT_INIT mitigates this threat by requiring the TOE to provide an initialization function that correctly transfers the software portion of the TSF implementation and TSF data into the TSF's security domain and to correctly establish the TOE in an operational configuration consistent with the configuration vector that defines the configuration data. O.CORRECT_LOAD mitigates this threat by ensuring that the load function accurately prepares the software portion of the TSF implementation and TOE configuration vectors for use by the TOE initialization function.

Threat	Objectives Addressing Threat	Rationale
T.COVERT_CHANNEL_EXPLOIT *An unauthorized information flow may occur between partitions as a result of covert channel exploitation.*	O.COVERT_CHANNEL_ANALYSIS *The TOE will undergo appropriate covert channel analysis by NSA to demonstrate that the TOE satisfies covert channel mitigation metrics.* OE.COVERT_CHANNELS *If the TOE has covert storage and/or timing channels, then all subjects executing on that TOE will, relative to the IT assets to which they have access, have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.*	Unauthorized information flow may occur between partitions as a result of covert channel exploitation. O.COVERT_CHANNEL_ANALYSIS mitigates this threat by validating the vendor's covert channel analysis through testing and analysis. OE.COVERT_CHANNELS mitigates this threat by requiring that subjects capable of exploiting covert channels are trusted not to do so. See rationale for A.COVERT_CHANNELS.
T.DENIAL_OF_SERVICE *A malicious subject may block others from system resources (e.g., system memory, persistent storage, and processing time) via a resource exhaustion attack.*	O.RESOURCE_ALLOCATION *The TOE will provide mechanisms that enforce constraints on the allocation of exported TOE resources.* O.BOUNDED_EXECUTION *The TOE will exhibit predictable and worst-case bounded execution behavior.*	The need to share resources (e.g., system memory, and processing time) between subjects introduces the potential for one subject to not be able to obtain the number of resources it requires to perform its function. Additionally, the TSF internally may contribute to a denial-of-service, observable at the TSFI, due to its unbounded use of resources. O.RESOURCE_ALLOCATION contributes to mitigation of this threat by requiring the TOE to enforce the allocation of system resources to partitions according to the constraints in the configuration data. These constraints include the allocation of minimum and maximum quotas for consumable resources. O.BOUNDED_EXECUTION contributes to mitigation of this threat by requiring that the TSF has predictable execution properties to include a worst-case execution property that remains within defined bounds.
T.INCORRECT_CONFIG *The configuration vectors are not an accurate and complete description of the operational configuration of the TOE as used by an organization.*	O.CORRECT_CONFIG *The TOE will provide procedures and mechanisms to generate the configuration vectors such that they accurately describe the operational configuration of the TOE as used by an organization.*	Since the policy enforcement functions of the TSF depend on the correctness of the TSF data, and the TOE's initialization mechanism generates the TSF data from the configuration vector used during initialization of the TOE, it is important that the mechanisms used to generate the configuration vectors are subjected to analysis and testing with developmental assurance commensurate with the rest of the TOE. O.CORRECT_CONFIG mitigates this threat by requiring the mechanisms used to generate the configuration vector (e.g., a configuration vectors generation tool) be included as part of the TOE and within the scope of the TOE evaluation.

Threat	Objectives Addressing Threat	Rationale
T.INCORRECT_LOAD *The software portion of the TSF implementation and/or configuration vectors are not correctly converted into a TOE-useable form.*	O.CORRECT_LOAD *The TOE will provide procedures and mechanisms to correctly convert the software portion of the TSF implementation and/or configuration vectors into a TOE-useable form.*	O.CORRECT_LOAD mitigates this threat by requiring the mechanisms used to convert the software portion of the TSF implementation and/or configuration vectors into a form that is usable by the TOE initialization mechanism be included as part of the TOE and within the scope of the TOE evaluation.

Threat	Objectives Addressing Threat	Rationale
T.INSECURE_STATE *The TOE may be placed in an insecure state as a result of an erroneous initialization, halt, reconfiguration or restart, transition to maintenance mode, or as a result of an unsuccessful recovery from a system failure or discontinuity.*	O.INIT_SECURE_STATE *The TOE will provide mechanisms to transition the TSF to an initial secure state without protection compromise.* O.RECOVERY_SECURE_STATE *The TOE will provide procedures and/or mechanisms, which can be used in the event of failure, faults, or discontinuity, to preserve secure state and to transition the TSF back to a secure state without protection compromise.* O.CORRECT_TSF_OPERATION *The TOE will provide a runtime self-test capability.* *The TOE will provide the means for an authorized subject to invoke and obtain the results of the self-test.* *The TOE will take action in response to any failure of a runtime self-test capability.* O.TRANSITION *The TOE will provide the capabilities for an authorized subject to restart the TOE, halt the TOE and transition the TOE into maintenance mode.* O.TSF_INTEGRITY *The TOE will verify the integrity of the TSF code and data.* O.SECURE_STATE *The TOE will preserve secure state during an execution session.*	Any change in TOE state or change in TOE mode of operation presents the possibility for the TOE to be placed in an insecure state. To mitigate this threat, it is necessary to ensure that the notion of secure state exists and is preserved throughout all TOE transitions between states and between modes. The combination of the following objectives mitigate this threat: O.INIT_SECURE_STATE requires the TOE to provide the mechanisms to initialize the TSF into an initial secure state during TOE initialization. O.SECURE_STATE requires the TOE to ensure that secure state is preserved during an execution session. O.RECOVERY_SECURE_STATE requires the TOE to provide procedures and/or mechanisms to ensure recovery without further protection compromise. The TOE developer is required to list the specific recovery condition(s) that the TOE may be placed in an insecure state and, for each condition, the associated recovery action to be taken by the TSF. The TSF is required to attempt to halt the TOE if it is unable to proceed with any recovery action. O.CORRECT_TSF_OPERATION requires runtime self-tests to be performed to demonstrate the correct operation of the TSF's implementation (hardware and software). This includes providing the means for an authorized subject to invoke the execution of self-tests at its discretion and for such an authorized subject to obtain the results of the self-test for analysis and possible response action. This objective also requires the TOE to take action upon the detection of any self-test failure. The action to be taken is ST-specific. O.TRANSITION requires the TOE to provide an authorized subject the ability to perform the following operations to facilitate trusted recovery: restart the TOE, halt the TOE and transition the TOE into maintenance mode. O.TSF_INTEGRITY requires the TOE to verify the integrity of the TSF code and data during normal operation.

Threat	Objectives Addressing Threat	Rationale
T.LEAST_PRIVILEGE *The design and implementation of the TSF internals may not suffice to limit the damage resulting from accident, error or unauthorized use.*	O.INTERNAL_LEAST_PRIVILEGE *The entire TSF will be structured to achieve the principle of least privilege among TSF modules.*	The application of the principle of least privilege to the TSF internal design and implementation minimizes the damage posed by any threat that results in erroneous behavior within the TSF. O.INTERNAL_LEAST_PRIVILEGE requires that the TSF be structured such that the principle of least privilege is applied to the internal software architecture and implementation of the TSF. Supporting the principle of least privilege in the internals of the TSF limits the damage that can result from accident, error or unauthorized use.
T.POOR_DESIGN *Unintentional or intentional errors in requirements specification or design of the TOE may occur, leading to flaws that may be exploited by a malicious subject.*	O.CHANGE_MANAGEMENT *The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled by trusted individuals throughout the TOE's development.* O.SOUND_DESIGN *The TOE will be designed using sound design principles and techniques which will be accurately documented.* *The TOE design will be completely and accurately documented.* O.VULNERABILITY_ANALYSIS_TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	Intentional or unintentional errors may occur in the requirements specification, design or development of the TOE. To address this threat, O.SOUND_DESIGN requires sound design principles and techniques that help prevent flaws in the TOE's design by eliminating errors in the logic, and for the design to be completely and accurately documented. This provides the evaluation team with the means to independently reach the same conclusions as the development team with regard to the ability of the TSF to adequately mitigate defined threats and enforce defined policies while meeting its security functional requirements. In addition, O.CHANGE_MANAGEMENT addresses this threat by requiring all changes to the TOE and its development evidence be analyzed, tracked and controlled by trusted individuals throughout the development cycle. To verify that there are no intentional or unintentional errors introduced in the design, O.VULNERABILITY_ANALYSIS_TEST demonstrates that the design of the TOE is resistant to attacks that exercise these design flaws and development errors.

Threat	Objectives Addressing Threat	Rationale
T.POOR _IMPLEMENTATION *Unintentional or intentional errors in implementation of the TOE design may occur, leading to flaws that may be exploited by a malicious subject.*	O.CHANGE_MANAGEMENT *The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled by trusted individuals throughout the TOE's development.* O.FUNCTIONAL_TESTING *The TOE will undergo independent security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.SOUND_IMPLEMENTATION *The implementation of the TOE will be an accurate instantiation of its design.* O.VULNERABILITY_ANALYSIS_TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	Intentional or unintentional errors may occur when implementing the design of the TOE. To address this threat, O.SOUND_IMPLEMENTATION ensures that the implementation is an accurate representation of the design. To ensure that an accurate representation of the design is maintained, O.CHANGE_MANAGEMENT ensures that all changes to the TOE and its development evidence are analyzed, tracked and controlled by trusted individuals throughout the development cycle. To ensure that errors have not been introduced, O.FUNCTIONAL_TESTING validates that the TSF satisfies the security functional requirements. To further demonstrate that vulnerabilities are not present, O.VULNERABILITY_ANALYSIS_TEST adds confidence that the TOE is not susceptible to attack despite being an accurate and complete instantiation of the design.

Threat	Objectives Addressing Threat	Rationale
T.POOR_TEST *Lack of or insufficient evaluation and runtime tests to demonstrate that all TOE security functions operate correctly (including in a fielded TOE) may result in incorrect TOE behavior being undiscovered.*	O.CORRECT_TSF_OPERATION *The TOE will provide a runtime self-test capability.* *The TOE will provide the means for an authorized subject to invoke and obtain the results of the self-test.* *The TOE will take action in response to any failure of a runtime self-test capability.* O.FUNCTIONAL_TESTING *The TOE will undergo independent security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.VULNERABILITY_ANALYSIS_TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	Design analysis determines that documented design of the TOE satisfies its security functional requirements. In order to ensure the TOE's design is correctly realized in its implementation, the appropriate level of functional testing of the TOE's security mechanisms must be performed during the evaluation of the TOE. TOE testing also includes the ability of the TOE to execute a suite of tests as necessary during runtime to ensure that the TSF continuously operates correctly. O.FUNCTIONAL_TESTING ensures that independent functional testing is performed to demonstrate the TSF satisfies the security functional requirements and the TOE's security mechanisms operate as documented. While functional testing serves an important purpose, it does not ensure the TSFI cannot be used in unintended ways to circumvent the TOE's security policies. O.VULNERABILITY_ANALYSIS_TEST addresses this concern by requiring that vulnerability analysis and penetration testing be performed. This objective provides a measure of confidence that the TOE does not contain security flaws that may not be identified through functional testing. While these testing activities are a necessary activity for successful completion of an evaluation, this testing activity does not address the concern that the TOE continues to operate correctly and enforce its security policies during normal operation. Some level of testing must be available to ensure the TOE's security mechanisms continue to operate correctly once the TOE is fielded. O.CORRECT_TSF_OPERATION ensures that once the TOE is installed at a customer's location, a TSF self-testing capability exists to provide end users the confidence that the TOE's security policies continue to be enforced. The ability for an authorized subject to obtain the results of TSF self-tests allows recovery action to be initiated from outside of the TSF. This objective also ensures that the TOE takes action upon the detection of any self-test failure. The action to be taken is ST-specific.

Threat	Objectives Addressing Threat	Rationale
T.TSF_COMPROMISE *A malicious subject may cause TSF data or executable code to be inappropriately accessed (viewed, modified, or deleted).*	O.REFERENCE_MONITOR *The TOE will provide a reference validation mechanism responsible for the enforcement of the TSP.* *The reference validation mechanism will execute in its own security domain.* *The reference validation mechanism must be tamper proof, its enforcement functions must be always invoked, and its design and implementation must be of size and complexity small enough to be subject to analysis and tests, the completeness of which can be assured.*	O.REFERENCE_MONITOR addresses the threat of tampering with or destruction of TSF software and TSF data (when the TSF is executing). It ensures that the TSF maintains a security domain for its own execution that protects it from interference and tampering.
T.UNAUTHORIZED_ACCESS *A subject may gain access to resources or TOE security management functions for which it is not authorized according to the TOE security policy.*	OE.PHYSICAL *Physical security will be provided for the TOE by the non- IT environment commensurate with the value of the IT assets protected by the TOE.* O.ACCESS *The TOE will ensure that subjects gain only authorized access to exported resources.* O.AUTHORIZED_SUBJECT *The TOE will ensure that only authorized subjects are allowed to access restricted services.* O.RESIDUAL_INFORMATION *The TOE will ensure that any information contained in a protected resource is not released to subjects when the resource is reallocated.* O.SUBJECT_ISOLATION *The TOE will provide mechanisms to protect each subject from unauthorized interference by other subjects.* O.MANAGE *The TOE will provide all the functions necessary to support the administrative users and authorized subjects in their management of the TOE security functions and configuration data, and restrict these functions from use by unauthorized subjects.* O.TRANSITION *The TOE will provide the capabilities for an authorized subject to restart the TOE, halt the TOE and transition the TOE into maintenance mode.*	Unauthorized users may physically tamper with the TOE hardware to gain unauthorized access to TOE resources. To mitigate this threat, OE.PHYSICAL establishes physical controls that restrict physical access to the TOE to only authorized personnel. Within the computing environment, O.ACCESS only allows that subjects can gain access only to those exported resources for which they are authorized, and O.AUTHORIZED_SUBJECT only allows subjects to gain access to restricted services for which they are authorized. The potential for unauthorized access via an information flow in violation of the TOE security policy occurs when hardware resources are deallocated from one subject and allocated to another. O.RESIDUAL_INFORMATION mitigates this aspect of the threat by requiring the TOE to ensure that unauthorized access to the residual information contained in a resource, once disassociated with one subject, is not accessible when the resource is allocated to another subject. At the same time, O.SUBJECT_ISOLATION provides mechanisms to enforce domain separation to protect each subject from unauthorized interference by other subjects. To counter the threat that unauthorized subjects could gain access to TOE security management functions, O.MANAGE and O.TRANSITION together require the TOE to restrict access to these management functions to authorized subjects only.

7.2 Objectives derived from Security Policies

130 Each of the identified security policies is addressed by one or more security objectives. Table 7.2 below provides the mapping from security objectives to security policies, as well as a rationale that discusses how the policy is addressed. Definitions are provided (in italics) below each policy and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.2. Mapping of Security Objectives to Security Policies

Security Policy	Objectives Addressing Policy	Rationale
P.ACCOUNTABILITY *The TOE shall provide the capability to make available information regarding the occurrence of security relevant events.*	O.AUDIT_GENERATION *The TOE will provide the capability to detect, generate and export audit records for security relevant auditable events.*	This policy requires the TOE to detect and make available information associated with security relevant events. Such information can aid the analysis/debugging of security-related errors, and provides the means for authorized subjects to take action in response of security relevant events. O.AUDIT_GENERATION enforces this policy by requiring the TOE to detect the occurrence of security relevant events and to generate audit data associated with those events.
P.CONFIGURATION_CHANGE *The TOE shall support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.*	O.CONFIGURATION_CHANGE *The TOE will support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.*	This policy is driven by operational needs to change the configuration of the TOE. It requires the TOE to support a static configuration change capability. Additionally, this policy allows the TOE to provide the means to perform a configuration change of the TOE whereby an authorized subject is able to select or redefine the configuration vector to be used when the TOE is started, restarted, or reconfigured in the absence of a start or restart. A TOE that implements any variation of an on-line configuration change capability in accordance with this policy introduces a configuration-change-related threat. Refer to T.CONFIGURATION_CHANGE for the mapping and rationale for this threat. O.CONFIGURATION_CHANGE enforces this policy by requiring the TOE to support a static configuration change capability, and makes provisions for the TOE to provide support for a configuration change capability invoked by an authorized subject.

Security Policy	Objectives Addressing Policy	Rationale
P.CRYPTOGRAPHY *The TOE shall use NSA approved cryptographic mechanisms.*	O.CRYPTOGRAPHY *The TOE will use NIST FIPS-validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, handling, and storage of keys) and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).*	The use of cryptographic mechanisms for which their correctness and/or strength are not fully understood presents a vulnerability that, if exploited, could undermine all aspects of the TOE's ability to meet its objectives. To mitigate this threat, O.CRYPTOGRAPHY enforces this policy by requiring that the TOE employ cryptographic solutions that at a minimum have been validated by NIST FIPS processes and which employ NSA approved methods for key management and for cryptographic operations.
P.INDEPENDENT_TESTING *The TOE shall undergo independent testing.*	O.FUNCTIONAL_TESTING *The TOE will undergo independent security functional testing that demonstrates the TSF satisfies the security functional requirements.* O.VULNERABILITY_ANALYSIS_TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	This policy requires the TOE to undergo independent testing to verify that the implementation is an accurate instantiation of the requirements and to provide additional confidence that in meeting its requirements, the TOE is sufficiently resistant to the capabilities of attackers with high attack potential, motivation, expertise and resources. O.FUNCTIONAL_TESTING demonstrates the TSF satisfies the appropriate security functional requirements. O.VULNERABILITY_ANALYSIS_TEST requires the TOE to undergo vulnerability analysis and penetration testing to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.
P.RATINGS_MAINTENANCE *A plan for procedures and processes to maintain the TOE's rating shall be in place to maintain the TOE's rating once it is evaluated.*	O.RATINGS_MAINTENANCE *Procedures and processes to maintain the TOE's rating will be documented.*	This policy requires the TOE developer to provide a plan that documents the procedures and processes to maintain the evaluated rating that is ultimately awarded to the TOE. O.RATINGS_MAINTENANCE satisfies this policy by requiring the TOE developer to provide the required rating maintenance plan.

Security Policy	Objectives Addressing Policy	Rationale
P.SYSTEM_INTEGRITY *The TOE shall provide the ability to periodically validate its correct operation.*	O.CORRECT_TSF_OPERATION *The TOE will provide a runtime self-test capability.* *The TOE will provide the means for an authorized subject to invoke and obtain the results of the self-test.* *The TOE will take action in response to any failure of a runtime self-test capability.*	This policy requires the TOE to periodically test itself to provide some measure of confidence that the TOE is operating in accordance with its security policies. O.CORRECT_TSF_OPERATION supports this policy by requiring the TOE to provide a capability to test the TSF to demonstrate the correct operation of the TSF in its operational environment. The ability for an authorized subject to obtain the results of TSF self-tests enables validation of TOE correct operation from outside of the TSF. This objective further supports this policy by requiring the TOE to take action upon the detection of any self-test failure. The action to be taken is ST-specific.
P.USER_GUIDANCE *The TOE shall provide documentation regarding the correct use of the TOE security features.*	O.USER_GUIDANCE *The TOE shall provide users with the necessary information for secure use of the TOE.*	This policy requires that the TOE documentation provide adequate information for the secure use and operation of the TOE. O.USER_GUIDANCE satisfies this policy by requiring that the necessary user information be provided.
P.VULNERABILITY _ANALYSIS_AND_TEST *The TOE shall undergo independent vulnerability analysis and penetration testing by NSA to demonstrate that the TOE is resistant to an attacker possessing a high attack potential.*	O.VULNERABILITY_ANALYSIS_ TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	O.VULNERABILITY_ANALYSIS_TEST satisfies this policy by ensuring that an independent vulnerability analysis is performed on the TOE and penetration testing based on that analysis is performed. Having an independent party perform the analysis helps ensure objectivity and eliminates preconceived notions of the TOE's design and implementation that may otherwise affect the thoroughness of the analysis. The level of analysis and testing requires that an attacker with a high attack potential cannot compromise the TOE's ability to enforce its security policies.

7.3 Objectives derived from Assumptions

131 Each of the identified security assumptions is addressed by one or more security objectives. Table 7.3 below provides the mapping from security objectives to security assumptions, as well as a rationale that discusses how the assumption is addressed. Definitions are provided (in italics) below each assumption and security objective so the PP reader can reference these without having to go back to sections 3 and 4.

Table 7.3. Mapping of Security Objectives to Assumptions

Assumption	Objectives Addressing Assumption	Rationale
A.PHYSICAL *It is assumed that the non-IT environment provides the TOE with appropriate physical security commensurate with the value of the IT assets protected by the TOE.*	OE.PHYSICAL *Physical security will be provided for the TOE by the non-IT environment commensurate with the value of the IT assets protected by the TOE.*	OE.PHYSICAL addresses this assumption by requiring the non-IT environment to provide physical security for the TOE that is commensurate with the value of the IT assets protected by the TOE.
A.SUBJECT_ALLOCATION *It is assumed that a properly trained trusted individual will create configuration vectors such that, for those partitions to which subjects are allocated, each partition is allocated one or more subjects (i.e., subjects with homogeneous access requirements, or subjects with heterogeneous access requirements) that are appropriate for the policy abstraction supported by the TOE.*	OE.SUBJECT_ALLOCATION *A properly trained trusted individual will create configuration vectors such that, for those partitions to which subjects are allocated, each partition is allocated one or more subjects (i.e., subjects with homogeneous access requirements, or subjects with heterogeneous access requirements) that are appropriate for the policy abstraction supported by the TOE.*	OE.SUBJECT_ALLOCATION addresses this assumption by requiring a trusted individual to allocate subjects to partitions such that each partition's configuration is appropriate for the policy abstraction supported by the TOE. Having trust in an individual requires proper training to ensure they fully understand the consequences of their actions and can be relied upon to take correct action in performing their duties..
A.COVERT_CHANNELS *If the TOE has covert storage and/or timing channels, then for all subjects executing on that TOE, it is assumed that relative to the IT assets to which they have access, those subjects will have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those channels.*	OE.COVERT_CHANNELS *If the TOE has covert storage and/or timing channels, then all subjects executing on that TOE will, relative to the IT assets to which they have access, have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.*	The purpose of the assumption is to provide insight to the risk associated with "system-level" use of the TOE, and to identify the assurance action that should be taken to ensure that the risk is properly understood and mitigated. The SKPP allows storage and timing covert channels to exist in the TOE implementation. These channels allow any subject (e.g., application program) to violate the TSP. Therefore, if a TOE has covert channels, and it is deployed in an environment which cannot tolerate the risk to the IT assets associated with those channels, then the applications configured to run on the TOE should be of sufficient assurance (e.g., through CC evaluation or some other means) that they can be trusted to not exercise the covert channels. The assurance should be commensurate with the value of the IT assets, at least, and in the case of the threat environment targeted by this PP, would be the same as the TOE (viz., EAL6+).
A.TRUSTED_FLOWS *For any subject configured to have unrestricted access in multiple policy equivalence classes, it is assumed that the subject is trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which*	OE.TRUSTED_FLOWS *For each configuration of the TOE, a partial order of the flows that are allowed between policy equivalence classes will be identified. Any subject allowed by the configuration data to cause information flow that is contrary to the partial order will*	The purpose of the assumption is to provide insight to the risk associated with "system-level" use of the TOE, and to identify the assurance action that should be taken to ensure that the risk is properly understood and mitigated.

Assumption	Objectives Addressing Assumption	Rationale
it has access.	*be trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.*	Denning [9] has shown that all security policies that restrict information flow can be represented as a lattice policy. The SKPP describes a class of products for managing information flow (in addition to least privilege) in its applications. Therefore, requirements in the SKPP based on an assumption of a lattice-bounded application-level policy are not restrictive to its general interpretation. A significant property of lattice flow policies is that they do not allow two-way flows between equivalence classes. OE.TRUSTED_FLOWS addresses this assumption by requiring that a subject capable of causing information flow in violation of the partial ordering of information flows between partitions be trusted with assurance commensurate with the value of the IT assets in all partitions to which it has access. The "partial ordering" requirement addresses a significant characteristic of the class of systems represented by this protection profile. Partitions between which flows occur in violation of the partial ordering result in a logical equivalence class of information in those partitions, since all information can be shared between the partitions. In some cases, flows between partitions in violation of the partial ordering are useful when constructing an application, if it can be assured that only certain information is permitted to flow in violation of the partial ordering. If a subject has insufficient assurance, then it may be assumed to cause unintended flows between the partitions. While the subject-to-resource flow controls can be used to prevent inter-partition flows otherwise allowed by the partition-to-partition flow rules, it is generally intractable to determine which information in a partition will be (e.g., transitively) allowed to flow into another partition once the flow is allowed by a partition-to-partition flow rule and a subject-to-resource flow rule (e.g., to support a guard or downgrader application). Therefore, if such an inter-partition flow were allowed, a requirement of the

Assumption	Objectives Addressing Assumption	Rationale
		environment is that the subject (e.g., application) have a level of trust that is adequate to protect the information in both the source and the destination partitions.
A.TRUSTED_INDIVIDUAL *It is assumed that any individual allowed to perform procedures upon which the security of the TOE may depend is trusted with assurance commensurate with the value of the IT assets.*	OE.TRUSTED_INDIVIDUAL *Any individual allowed to perform procedures upon which the security of the TOE may depend must be trusted with assurance commensurate with the value of the IT assets.*	OE.TRUSTED_INDIVIDUAL addresses this assumption by requiring that any individual who is allowed to perform procedures that affect the security of the TOE be trusted with assurance commensurate with the value of the IT assets. This requirement is allocated to the non-IT environment because there are no Identification & Authentication requirements for the TOE.

7.4 Requirements Rationale

132 Each of the TOE security objectives identified in section 4 is addressed by one or more security requirements. Table 7.4 below provides the mapping from security requirements to security objectives, as well as a rationale that discusses how the security objective is met. Definitions are provided (in italics) below each security objective so the PP reader can reference these without having to go back to section 4.

Table 7.4. Mapping of Security Requirements to Objective

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
O.ACCESS *The TOE will ensure that subjects gain only authorized access to exported resources.*	FDP_IFC.2 FDP_IFF.1 FPT_RVM.1	This objective requires the TOE to manage resources that it controls such that subjects can only gain access to those resources that they are permitted to use. The combination of FDP_IFC.2, FDP_IFF.1 and FPT_RVM.1 satisfies this objective. FDP_IFC.2 requires the TSF to enforce the Partition Information Flow Control policy on all partitions, subjects and exported resources and all operations that cause information to flow between partitions and to and from all subjects. FDP_IFF.1 specifies the policy rules for the selected PIFP abstraction to be enforced by the TSF and the security attributes used by the enforcement rules. For the Partition Abstraction, the Partition Information Flow Control policy rule requires the TSF to permit an information flow between partitions only if the mode of the flow associated with the requested operation is explicitly authorized by the configuration data. For the Least Privilege Abstraction, the Partition Information Flow Control policy rule requires the TSF to permit an information flow between a subject

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		and an exported resource if the mode of that flow is explicitly authorized by the configuration data. The PIFP rules apply to all partitions and resources for a given execution session. There is no need to support a heterogeneous configuration of PIFP abstractions for a given execution session because the Least Privilege Abstraction is a superset of the Partition Abstraction. FPT_RVM.1 requires that the TSF makes policy decisions on all attempts to access the TOE resources. Without assurance that this non-bypassability requirement is being met, the TSF could not be relied upon to completely and continuously enforce the Partitioned Information Flow policy.
O.ADMIN_GUIDANCE *The TOE will provide administrators with the necessary information for secure management of the TOE.*	ADO_IGS.1 AGD_ADM_EXP.1	ADO_IGS.1 requires the developer to provide the procedures necessary to securely install and start-up an instance of the exact evaluated configuration of the TOE. AGD_ADM_EXP.1 requires the developer to provide administrative guidance to configure and administer the TOE securely for the IT environment within which it is intended to operate. The necessary information for secure management of the TOE includes instructions on proper use of the administrative functions, warnings about functions and privileges that should be controlled, assumptions regarding user behavior, correct settings of security parameters, and security requirements for the IT environment.
O.AUDIT_GENERATION *The TOE will provide the capability to detect, generate and export audit records for security relevant auditable events.*	FAU_ARP.1 FAU_GEN.1 FAU_SAR_EXP.1 FAU_SEL_EXP.1 FPT_STM.1	The FAU_ARP.1 requirement is intended to ensure that some action is taken upon the failure of the tests associated with either FTP_AMT.1or FTP_TST.1. The ST author is required to specify the action to be taken. FAU_GEN.1 defines the set of auditable events for which the TOE must be capable of generating audit records. For each specified auditable event, this requirement defines the minimum amount of data associated with that event that must also be recorded. This requirement establishes the minimum level of data that must be recorded for any additional audit events that are specified in the ST by the TOE developer. FAU_SAR_EXP.1 requires the TSF to export audit data in a form that supports audit data analysis by an authorized subject. FAU_SEL_EXP.1 requires the TSF to generate audit records for those auditable events that have actually been selected to be audited, based on attributes associated with each audit event. This provides flexibility in detecting only those events that are deemed necessary by site policy, thus reducing the amount of resources consumed by the audit mechanism. To support the post-runtime analysis of audit data the FAU_GEN.1 requirement associates a time attribute with each recorded event. FPT_STM.1 requires the TSF to provide a reliable time stamps such that there is confidence in the integrity of the sequencing and time relationships between audit events

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		based upon the time attribute recorded with each event.
O.AUTHORIZED_SUBJECT *The TOE will ensure that only authorized subjects are allowed to access restricted services.*	FMT_MOF.1 FMT_MSA_EXP.1 FMT_MTD.1 FMT_MCD_EXP.1	Multiple iterations of the base FMT_MOF.1 component requires the TSF to prevent unauthorized subjects from invoking specific TSF functions that control TOE behavior, result in TOE state changes, and that provide access to TSF data. FMT_MSA_EXP.1 requires the TSF to assign authorizations to subjects as specified by the configuration data. Multiple iterations of FMT_MTD.1 require the TSF to restrict access to specified TSF data. FMT_MCD_EXP.1 requires the TSF to prevent any modification to the configuration data The combination of FMT_MOF.1, FMT_MTD.1, FMT_MCD_EXP.1 and FMT_MSA_EXP.1 ensures that only authorized subjects, where that authorization is explicitly defined by the configuration data, are able to access restricted TOE services.
O.BOUNDED_EXECUTION *The TOE will exhibit predictable and worst-case bounded execution behavior.*	FRU_PRU_EXP.1 ADV_ARC_EXP.1	FRU_PRU_EXP.1 establishes metrics defining the behavior of the TSF, in terms of its predictable use of processor resources and its maximum use of memory resources. These metrics provide the basis for the analysis required by ADV_ARC_EXP.1 ADV_ARC_EXP.1 requires the developer to present the architecture design in a manner that demonstrates the metrics for use of processor and memory resources are being met.
O.CHANGE_MANAGEMENT *The configuration of, and all changes to, the TOE and its development evidence will be analyzed, tracked, and controlled by trusted individuals throughout the TOE's development.*	ACM_AUT.2 ACM_CAP.5 ACM_SCP.3 ALC_DVS.2 ALC_FLR.3 ALC_LCD.2 ALC_TAT.3	This objective is satisfied by the following Configuration Management (CM) and Life Cycle (LC) requirements. ACM_AUT.2 requires the TOE developer to have a CM plan and use a CM system that provides an automated means to enforce controls on changes made to all configuration items that comprise the TOE, and that supports the generation of the TOE. This requirement also requires the developer to describe in the CM plan the automated tools used in the CM system and how those tools are used in the CM system. Thus, ACM_AUT.2 aids in understanding how the CM system enforces the control over changes made to the TOE. ACM_CAP.5 requires the developer to describe in the CM plan how changes to the TOE and its evaluation deliverables are managed by the CM system. The CM system is required to operate in accordance with the CM plan and provide the capability to control who on the development staff can make changes to the TOE and its developed evidence. Furthermore, the CM system is required to enforce separation of duties (e.g., developers cannot be part of the CM staff), clearly identify the configuration items that comprise the TSF, and support the audit of modifications to the TOE. In addition to the CM plan and CM system, the developer is also required to provide a list of uniquely identified configuration

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		items that comprise the TOE, an acceptance plan and integration procedures. The configuration list is used by the CM system to control unauthorized modification, addition, or deletion of the TOE configuration items, and by the integration procedures to ensure that the TOE is generated correctly. The acceptance plan describes how modified or newly created configuration items are reviewed and accepted as part of the TOE. The developer is required to justify that the acceptance procedures provide for an adequate and appropriate review of all changes to the TOE. This requirement satisfies the "analyzed" aspect of this objective. ACM_SCP.3 is necessary to define what items must be under the control of the CM system. This requirement ensures that the TOE implementation representation, design documentation, test documentation (including the executable test suite), user and administrator guidance, CM documentation, security flaws, and development tools (and related information) are tracked by the CM system. ALC_DVS.2 requires the developer to describe the security measures used in the development environment to ensure the integrity and confidentiality of the TOE. Furthermore, the developer must also provide evidence that these security measures are followed by the development team, and justify that these measures provide the necessary level of protection. The physical, procedural, and personnel security measures the developer uses provides an added level of control over who and how changes are made to the TOE and its associated evidence. ALC_FLR.3 requires the developer to track and correct flaws in the TOE that have been discovered either through developer actions (e.g., developer testing) or by others. In addition to correcting discovered flaws, the flaw remediation process used by the developer must also ensure that new flaws are not created while fixing the discovered flaws. The developer is also required to support timely automatic distribution of security flaw reports and associated corrections, and to inform users who might be affected by the discovered flaws in a timely manner. ALC_LCD.2 requires the developer to use a standardized life-cycle model that describes the procedures, tools and techniques used in the development and maintenance of the TOE. Procedural aspects such as design methods, code or documentation reviews, how changes to the TOE are reviewed and accepted or rejected will add assurance for the TOE at the time of the initial evaluation and during its maintenance phases. The developer is required to explain why the particular life cycle model was chosen and how it is used, and to demonstrate that the life cycle documentation is compliant with the life cycle model. ALC_TAT.3 ensures that all the tools and techniques used during the development and maintenance of the TOE are well defined including the selected implementation-dependent options of the development tools. It also requires the developer to establish

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		implementation standards for all parts of the TOE. This will mitigate the risk of using ill-defined, inconsistent or incorrect development tools and techniques.
O.CONFIGURATION_CHANGE *The TOE will support the capability to perform a static configuration change. The TOE may also provide the capability for an authorized subject to select or redefine the configuration vector to be used upon TOE startup, TOE restart or TOE reconfiguration.*	FMT_MOF.1 FMT_MSA_EXP.1 FMT_SMF.1 FPT_CFG_EXP.1 FPT_ESS_EXP.1 FPT_RST_EXP.1	This objective requires the TOE to support a static configuration change capability. It also requires that, should the TOE provide an on-line configuration change capability, then the TOE is to allow only authorized subjects to make the selection or redefinition of the next configuration vector. This objective is met through combination of the following SFRs: FMT_MOF.1 requires the TSF to restrict the access to the TOE configuration change function to those subjects that are explicitly authorized. FMT_MSA_EXP.1 requires the TSF to base the assignment of subjects authorizations for TOE configuration change on attributes contained in the configuration data. FMT_SMF.1 requires the TSF to implement TOE security management capability to change the TOE configuration. FPT_CFG_EXP.1 provides the option for the TOE to support dynamic total, dynamic-constrained and dynamic-unconstrained reconfiguration capabilities. Each of these options has unique semantics with regards to the manner in which a new TSF internal vector can be selected or changed by an authorized subject. FPT_ESS_EXP.1 requires that the TOE be established in a secure state based on the configuration vector that is referenced during initialization. If the configuration vector is changed by some offline process, the TOE utilizes that changed configuration vector to initialize the TOE. FPT_RST_EXP.1 requires the TOE to have the capability for an authorized subject to restart the TOE which will result in a configuration change if a new configuration vector was selected prior to the restart.
O.CORRECT_CONFIG *The TOE will provide procedures and mechanisms to generate the configuration vectors such that they accurately describe the operational configuration of the TOE as used by an organization.*	ADV_CTD_EXP.1 AGD_ADM_EXP.1	ADV_CTD_EXP.1 requires the TOE developer to provide a configuration tool that generates configuration vectors that can be verified as being an accurate description of the intended TOE operational configuration as used by an organization. AGD_ADM_EXP.1 requires the developer to provide administrator guidance for the correct use of the configuration vector generation tool.
O.CORRECT_INIT *The TOE will provide mechanisms to correctly transfer the software portion of the TSF implementation and TSF data into the TSF's security domain and to correctly*	FPT_ESS_EXP.1 ADV_INI_EXP.1 AGD_ADM_EXP.1	This objective requires capabilities and assurances in regards to the integrity of the initialization mechanism. This objective is addressed by the combination of implemented capability and procedural controls. FPT_ESS_EXP.1 requires that the TSF be able to determine that it is in a secure state prior to allowing any information flows as

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
establish the TOE in an operational configuration consistent with the configuration vector that defines the configuration data.		authorized by the Partitioned Information Flow Policy. ADV_INI_EXP.1 requires the TOE developer to provide an initialization mechanism that establishes a security domain for the TSF, and to provide an initialization design that demonstrates how the software portion of the TSF implementation and TSF data are correctly transferred into the TSF security domain. Furthermore, it requires the TOE to be established in an operational configuration consistent with the configuration vector that defines the configuration data. AGD_ADM_EXP.1 requires the developer to provide administrator guidance for the proper use of the initialization mechanism.
O.CORRECT_LOAD *The TOE will provide procedures and mechanisms to correctly convert the software portion of the TSF implementation and/or configuration vectors into a TOE-useable form.*	ADV_LTD_EXP.1 AGD_ADM_EXP.1	ADV_LTD_EXP.1 requires the developer to provide a TOE load capability and a description of its design. The TOE load capability ensures that the software portion of the TSF implementation and data are correctly converted into a form that is accessible by the TOE initialization mechanism. AGD_ADM_EXP.1 requires the developer to provide administrator guidance for the proper use of the load mechanism.
O.CORRECT_TSF_OPERATION *The TOE will provide a runtime self-test capability.* *The TOE will provide the means for an authorized subject to invoke and obtain the results of the self-test.* *The TOE will take action in response to any failure of a runtime self-test capability.*	FAU_ARP.1 FMT_MOF.1 FMT_MSA_EXP.1 FMT_SMF.1 FPT_AMT.1 FPT_TST_EXP.1	This objective requires capabilities to periodically test the TSF and the abstract machine that underlies the TSF, to interpret the results of such tests, and to manage the use of those tests. FAU_ARP.1 ensures that the TOE takes action upon the detection of any failure of the tests associated with FPT_AMT.1 and FPT_TST1. FMT_MOF.1 requires the TSF to prevent a subject from invoking TSF self-test unless that subject has been granted authorization to do so. FMT_MSA_EXP.1 requires the TSF to assign the authorization for running self-tests to subjects as specified by the configuration data. FMT_SMF.1 requires the TSF to implement TOE security management capability for an authorized subject to invoke the self-test and to obtain the results of the self-test. FPT_AMT.1 provides the means to discover any failures in the hardware security mechanisms upon which the TSF is dependent, and therefore, could render the TSF ineffective in enforcing its security policies. This requirement requires the TSF to test the hardware security mechanisms during the initial start-up and also periodically during normal operation. FPT_TST_EXP.1 requires the TSF to run a suite of self-tests to verify the software portions of the TSF. This requirement explicitly specifies that the TSF self tests be run during the initial start-up, but leaves the conditions under which the self tests should occur during normal operation to be specified by the ST author. This allows the ST author to tailor the testing requirements to be appropriate to conditions of the TSF's normal

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		operation. Subjects with the appropriate authorization can invoke the TSF self-test and can obtain the results of the self-test. The tests required by FPT_AMT.1 and FPT_TST_EXP.1 collectively verify the correct operation and integrity of all three parts of the TSF, i.e., the TSF's underlying abstract machine, the TSF's implementation and the TSF's data.
O.COVERT_CHANNEL_ANALYSIS *The TOE will undergo appropriate covert channel analysis by NSA to demonstrate that the TOE satisfies covert channel mitigation metrics.*	FDP_IFF.3 AVA_CCA_EXP.2	The scope for elimination of covert channels is defined by FDP_IFF.3 and the objectives of the covert channel analysis are defined by AVA_CCA_EXP.2. FDP_IFF.3 requires an upper bound on the bandwidth associated with any identified covert storage and timing channels. AVA_CCA_EXP.2 requires the developer to perform a systematic search for inter-partition covert channels. It also requires the developer to document the covert channel analysis and provide the documentation as evaluation evidence. Since all subjects assigned to a partition are of the same equivalence class, a search for intra-partition covert channels is not needed. A systematic search, as opposed to an informal search, is necessary because it is important that the covert channels be identified in a structured and repeatable way to aid the validation of the covert channel analysis. AVA_CCA_EXP.2 also requires the NSA evaluator to confirm the results of the covert channel analysis and to selectively validate the covert channel analysis through testing. This will afford additional assurance evidence to support a high robustness evaluation.
O.CRYPTOGRAPHY *The TOE will use NIST FIPS-validated cryptography as a baseline with additional NSA-approved methods for key management (i.e., generation, access, distribution, destruction, handling, and storage of keys) and for cryptographic operations (i.e., encryption, decryption, signature, hashing, key exchange, and random number generation services).*	ADO_DEL_EXP.2	ADO_DEL_EXP.2 requires the TOE developer to select from a specific list of validated cryptographic mechanisms when determining how to implement the means to ensure the integrity of the TOE when it is transmitted from the TOE development facility to the end user.
O.FUNCTIONAL_TESTING *The TOE will undergo independent security functional testing that demonstrates the TSF satisfies the security functional requirements.*	APT_PCT_EXP.1 APT_PST_EXP.1 ATE_COV.3 ATE_DPT.3 ATE_FUN.2 ATE_IND.3	Rationales for APT explicit requirements are described in Appendix F. ATE_COV.3, ATE_DPT.3 and ATE_FUN.2 impose testing requirements on the developer to create and document the security test suite. ATE_IND.3 levies requirements on the evaluation team to independently verify the testing results. The combination of these requirements satisfies this objective. ATE_COV.3 requires the developer to provide an analysis of the test coverage to demonstrate that the TSF and TSF interfaces are

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		completely addressed by the developer's test suite. While this requirement does not require exhaustive testing of the TSF, it does impose rigorous testing of the TSF interfaces to ensure that the TSF interfaces meet their security functional requirements. This component also requires an independent confirmation of the completeness of the test suite. ATE_DPT.3 requires the developer to provide an analysis of the depth of the functional testing to demonstrate that the TSF is implemented and operates as specified by its high-level design and low-level design. This component complements ATE_COV.3 by ensuring that the developer takes into account the high-level and low-level design when developing their test suite. ATE_FUN.2 requires the developer to test the TSF and to provide documentation of the results. The ordering of execution of independent functional tests is required to be loop-free. The developer's test documentation must include an analysis of the test procedure ordering dependencies to demonstrate the testing is not circular. The developer must provide sufficient test documentation, i.e., the test plan, test procedures, and test results, to support independent verification of the test results and test coverage analysis. ATE_IND.3 requires the developer to provide the evaluator with the TOE and testing materials for independent testing. The developer must provide the same testing materials that were used by the developer to perform the developer's functional testing. These must include, minimally, test suite executables and source code, and machine-readable test documentation. ATE_IND.3 also levies testing requirements on the evaluator to verify the developer's test results by re-testing all tests performed by the developer, and to develop and run additional evaluator-developed tests that exercise the TOE in areas that are not well demonstrated by the developer's test suite. By repeating all of the developer's tests and running the evaluator-developed test suite, the evaluator can demonstrate that the TSF satisfies all security functional requirements as required by this objective.
O.INIT_SECURE_STATE *The TOE will provide mechanisms to transition the TSF to an initial secure state without protection compromise.*	FIA_ATD_EXP.1 FIA_USB_EXP.1 FPT_ESS_EXP.1 ADV_INI_EXP.1	Abstractly, the TOE consists of two distinct sets of functions: initialization functions and runtime functions. Initialization functions only execute during start-up and are not relied upon for security enforcement after the TOE is fully initialized. This protection profile defines the initialization functions as being outside the TSF because they are not relied upon for the enforcement of the TSP. Runtime functions, on the other hand, are relied upon, either directly or indirectly, to correctly enforce the TSP once the TSF is established in a secure state. FIA_ATD_EXP.1 defines the set of attributes to be contained in the configuration data that fully describe the configuration and

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		behavior of the TSF. FIA_USB_EXP.1 requires that the TSF properly translate and associate (bind) the attributes described in the configuration data to the runtime resources to which they are allocated. FPT_ESS_EXP.1 requires that the TSF be established in a secure state as defined by the attributes in the configuration data, that the TSF enforces the Partitioned Information Flow Policy as defined by the configuration data, and that no information flows are allowed to occur until the TSF determines that the above conditions have been met. ADV_INI_EXP.1 requires the TOE developer to provide an initialization mechanism that 1) brings the TSF to the secure state defined by the configuration data such that there is no protection comprise, 2) either completes successfully or halts due to unrecoverable errors, and 3) can not subvert the TSF after it completes and the TSF is in operational mode and enforcing the TSP.
O.INSTALL_GUIDANCE *The TOE will be delivered with the appropriate installation guidance to establish and maintain TOE security.*	ADO_DEL_EXP.2 ADO_IGS.1	This objective is satisfied by the documentation requirements of the trusted delivery and secure installation and start-up functions. ADO_DEL_EXP.2 requires the developer to describe the procedures and technical measures that the developer put in place to: 1) detect modifications during transit, 2) detect any discrepancy between the developer's master version and the delivered version, and 3) detect any attempts to masquerade as the developer. ADO_DEL_EXP.2 requires the developer to provide cryptographic mechanisms to protect the integrity of the TOE during delivery. ADO_DEL_EXP.2 also requires the developer to follow the developer-prescribed delivery procedures. After verifying that the TOE delivery from the developer is the right version and tamper-free, the user is responsible to configure and install the TOE in accordance with the TOE's intended use before running it. ADO_IGS.1 requires the developer to provide the guidance on how to use the installation and start-up procedures to install and start-up an instance of the TOE that was evaluated. ADO_IGS.1 further requires the evaluator to verify that if the procedures are used as described, they will result a secure installation and start-up of the TOE.
O.INTERNAL_LEAST_PRIVILEGE *The entire TSF will be structured to achieve the principle of least privilege among TSF modules.*	FPT_PLP_EXP.1 ADV_INT_EXP.3	FPT_PLP_EXP.1 establishes the requirement for TSF functions to be allocated the minimum required access to TSF data and TSF resources. The requirement for assurances associated with support for the principle of least privilege within the TSF was added as part of the explicit ADV_INT_EXP.3 component because the existing ADV_INT.3 component does not address the need to apply the principle of least privilege to the design and structure of the TSF.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		ADV_INT_EXP.3 expanded the scope of ADV_INT.3 by requiring the developer to design and structure the TSF such that the principle of least privilege can be achieved. Together with layering and minimization, least privilege imposes modularity on the implementation, thus making it more understandable. In combination, these provide a greater level of confidence in the analysis of the correctness of the implementation. ADV_INT_EXP.3 also requires the evaluator to confirm that the TSF has been internally structured to achieve least privilege among TSF modules.
O.MANAGE *The TOE will provide all the functions necessary to support the administrative users and authorized subjects in their management of the TOE security functions and configuration data, and restrict these functions from use by unauthorized subjects.*	FMT_MOF.1 FMT_MSA_EXP.1 FMT_MSA_EXP.3 FMT_MTD.1 FMT_MTD.3 FMT_MCD_EXP.1 FMT_SMF.1	This objective requires the TOE to provide adequate functions to securely manage the TSF, its behavior, and TSF data satisfies this objective. These functional requirements specifically support this objective by requiring the TSF to implement appropriate and sufficient TOE security management functions. FMT_MOF.1 requires the TSF to ensure that only authorized subjects are able to invoke the management functions. FMT_MSA_EXP.1 requires the TSF to assign authorizations to subjects, where the configuration data is the only source that can specify those authorizations. FMT_MSA_EXP.3 requires the TSF to use restrictive default values for the attributes contained in configuration data, and, requires that only the configuration is able to change the defaults. FMT_MTD.3 requires the TSF to perform syntax check on all TSF data. For example, the values that are accepted as valid must fall within the defined range. FMT_MCD_EXP.1 disallows modification of configuration data. FMT_MTD.1 places restrictions such that only authorized subjects are able to access specified TSF data. FMT_SMF.1 requires the TSF to implement TOE security management capabilities that are accessible to authorized subjects. The management capabilities include changing the TOE configuration if the TOE provides a configuration change capability, halting the TOE, restarting the TOE, invoking TSF self-test, and transitioning the TOE to maintenance mode.
O.RATING_MAINTENANCE *Procedures and processes to maintain the TOE's rating will be documented.*	AMA_AMP_EXP.1	The AMA family of requirements is incorporated into this PP to ensure the TOE developer has procedures and mechanisms in place to maintain the evaluated rating that is ultimately awarded the TOE. These requirements are somewhat related to the ACM family of requirements in that changes to the TOE and its evidence must be managed, but the AMA requirements ensure the appropriate level of analysis is performed on any changes made to the TOE to ensure the changes do not affect the TOE's ability to enforce its security policies. AMA_AMP_EXP.1 requires the developer to develop an assurance maintenance plan that describes how the assurance gained from an evaluation will be maintained, and that any

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		changes to the TOE will be analyzed to determine the security impact, if any, of the changes that are made. This requirement mandates the developer assign personnel to fulfill the role of a security analyst that is responsible for ensuring the changes made to the TOE will not adversely impact the TOE and that it will continue to maintain its evaluation rating.
O.RECOVERY_SECURE_STATE *The TOE will provide procedures and/or mechanisms that can be used in the event of failure, faults, or discontinuity, to preserve secure state and to transition the TSF back to a secure state without protection compromise.*	FPT_FLS.1 FPT_MTN_EXP.1 FPT_MTN_EXP.2 FPT_RCV.4 FPT_RCV_EXP.2	This objective requires the TSF to maintain secure state despite the presence of faults or failures and to prevent operations that constitute a violation of the TSP during the periods of recovery from specific faults or failures. The objective applies to both operational and maintenance mode. The following SFRs, in combination, enable the TSF to meet this objective: FPT_FLS.1 requires that by design, the TSF is able to fail securely, i.e., to preserve a secure state, when a specific set of failures are detected by TSF self-test. FPT_MTN_EXP.1 requires the TSF to be able to transition to maintenance mode when directed to do so by an authorized subject and to preserve secure state when the transition to maintenance mode is from a secure state. The TSF is further required to halt the TOE if the TSF is unable to preserve secure state after transitioning to maintenance mode from a secure state. FPT_MTN_EXP.2 requires that while in maintenance mode, the TSF prevents controlled operations from occurring if the TSF is unable to assure that a protection compromise will not occur by allowing the controlled operation to occur. FPT_RCV.4 requires the TSF to ensure that all security functions that are affected by the ST-defined failure scenarios can recover to a consistent and secure state if a ST-defined failure scenario is encountered during their execution. The ST author is required to specify the affected security functions and the list of failure scenarios from which the TSF is required to complete a full security function recovery. FPT_RCV_EXP.2 requires the TSF, in all cases where it detects an insecure state while the TOE is in operational mode, to take some action to recover the TOE to a secure state without further protection compromise. The TSF is also required to initiate recovery for the case where it detects an insecure state after the completion of the TOE initialization function. The ST author is required to list the specific recovery conditions that can be detected and to specify the associated recovery action for each condition taken by the TSF. Transistion to maintenance mode may be an acceptable recovery action – depending on the condition in the context of a specific TOE – and this transition can occur directly as part of the execution session or can be a mode that the TSF transitions to during initialization of the TOE. The TSF is required to attempt to halt the TOE if it is unable to proceed with any recovery action.
O.REFERENCE_MONIT	FPT_FLS.1	This objective requires the TSF to implement a reference

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
OR *The TOE will provide a reference validation mechanism responsible for enforcement of the TSP.* *The reference validation mechanism will execute in its own security domain.* *The reference validation mechanism must be tamper proof, its enforcement functions must be always invoked, and its design and implementation must be of size and complexity small enough to be subject to analysis and tests, the completeness of which can be assured.*	FPT_MTN_EXP.1 FPT_MTN_EXP.2 FPT_RCV_EXP.2 FPT_RVM.1 FPT_SEP.3 ADV_ARC_EXP.1 ADV_INT_EXP.3	validation mechanism (RVM). This objective is met by a set of SFRs and two SARs. FPT_FLS.1 requires the TSF to preserve secure state, when possible, in the event of a limited set of failures or service discontinuities. This property of the TSF ensures that the reference validation mechanism continues to function in a manner consistent with the information flow policy requirements. FPT_MTN_EXP.1 requires the TSF to preserve secure state when the transition to maintenance mode is from a secure state. FPT_MTN_EXP.2 requires that while in maintenance mode, the TSF prevents controlled operations from occurring if the TSF is unable to assure that a protection compromise will not occur by allowing the controlled operation to occur. FPT_RCV_EXP.2 requires the TSF to attempt to recover the TOE to a secure state when the TSF determines that it is not in a secure state immediately after completion of TOE initialization or at any time while the TOE is in operational mode. The TOE developer is to state the specific recovery action to take for each specified recovery condition. This ensures that the reference validation mechanism cannot be bypassed even in the event of non-recoverable failures. FPT_RVM.1 requires the TSF to enforce the TSP on all services and exported resources such that the enforcement functions are always invoked; it is not possible to bypass the enforcement mechanism to gain access to services and exported resources. FPT_SEP.3 requires the TSF to maintain three different types of security domains during runtime: 1) a separate domain for partition information flow control enforcement functions, 2) one or more separate domains for the remainder of the TSF that does not enforce the flow control SFPs, and 3) one or more separate domains for the non-TSF portions of the TOE, i.e., the subjects in the TSC. The SFP enforcement functions are the most important functions provided by the TSF, thus it is necessary to separate them from the less-critical portion of the TSF. The separation between the TSF and the non-TSF portion of the TOE is also necessary so that the non-TSF portion cannot interfere with the operation of the TSF. ADV_ARC_EXP.1 requires evidence to be provided that describes how the TSF protects itself from interference and tampering and how the TSF prevents bypass of the security enforcement functions. ADV_INT_EXP.3 requires the TSF be implemented such that its size and complexity is suitable for rigorous analysis methods that yield conclusive results.
O.RESIDUAL_INFORM	FDP_RIP.2	FDP_RIP.2 satisfies this objective by requiring that when an exported resource is reallocated, the TSF must ensure that no

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
ATION *The TOE will ensure that any information contained in a protected resource is not released to subjects when the resource is reallocated.*		unauthorized access to the residual information from the previous allocation is possible. Removal of residual information must occur at the point of deallocation or allocation. The ST author is to complete the selection to reflect the behavior of the implementation.
O.RESOURCE_ALLOCATION *The TOE will provide mechanisms that enforce constraints on the allocation of exported TOE resources.*	FRU_RSA.2	FRU_RSA.2 mandates that allocation limits be enforced for the minimum and maximum amount of memory and processing time available to a partition. Allocation requirements for system memory are based on the minimum and maximum simultaneous memory usage by each individual partition at any given time. Allocation limits on processing time are based on the minimum and maximum CPU usage by each individual partition over a specific time interval. A refinement to the wording of the choices provided by the CC in the selection operation was made. The allocation is made to the partition, which is inclusive of subjects and exported resources (there are no 'users' in the context of the SK allocation of resources).
O.SECURE_STATE *The TOE will preserve secure state during an execution session.*	FPT_CFG_EXP.1 FPT_ESS_EXP.1 FPT_FLS.1 FPT_HLT_EXP.1 FPT_MTN_EXP.1 FPT_MTN_EXP.2 FPT_RCV.4 FPT_RCV_EXP.2 FPT_RST_EXP.1	An execution session is the set of states from initialization to shutdown or restart of the TOE, and includes both operational and maintenance mode. This objective is met by the set of SFRs that require preservation of secure state. FPT_CFG_EXP.1 requires the TSF to preserve secure state during any change to the TOE configuration. FPT_ESS_EXP.1 requires the TSF to determine that it is established in a secure state prior to authorizing any information flows governed by the implemented Partitioned Information Flow Policy abstractions. Since no information flows are allowed to occur until after initialization completes, the TSF remains in a secure state throughout the TOE initialization process. FPT_FLS.1 requires the TSF to fail securely, i.e., to preserve a secure state, when failures are detected by the TSF self-tests and ST-specific conditions, if any. FPT_HLT_EXP.1 requires the TSF to preserve secure state during the transition from runtime to the halt state. FPT_MTN_EXP.1 requires the TSF to preserve secure state when the transition to maintenance mode is from a secure state. FPT_MTN_EXP.2 requires that while in maintenance mode, the TSF prevents controlled operations from occurring if the TSF is unable to assure that a protection compromise will not occur by allowing the controlled operation to occur. FPT_RCV.4 requires the TSF to ensure that all security functions that are affected by a ST-defined failure scenario either complete successfully or if a ST-defined failure scenario is encountered during their execution can recover to a consistent and secure state. Failure scenarios and the affected security functions are

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		ST-specific; therefore specification of the failure scenarios and the affected functions is left as an open assignment for the ST author. FPT_RCV_EXP.2 requires the TSF to attempt to recover the TOE to a secure state after the TSF detects that it is not in a secure state. The ST author is required to specify the detectable recovery conditions and associated recovery action to be taken by the TSF. The ST author is also required to specify the action that the TSF will take if a specified recovery action is not completed. One of the possible actions is for the TSF to take is to enter a maintenance mode that allows the TOE to return to a secure state. It is assumed that the IT environment provides adequate protection against unauthorized access to the maintenance mode. The TSF is required to attempt to halt the TOE if the TSF is unable to proceed with any recovery action. FPT_RST_EXP.1 requires the TSF to preserve secure state during a TOE restart, which may or may not include a halt state.
O.SOUND_DESIGN *The TOE will be designed using sound design principles and techniques which will be accurately documented.* *The TOE design will be completely and accurately documented.*	ADV_ARC_EXP.1 ADV_FSP_EXP.4 ADV_HLD_EXP.4 ADV_INI_EXP.1 ADV_INT_EXP.3 ADV_LLD_EXP.2 ADV_RCR.3 ADV_SPM.3 APT_PDF_EXP.1 APT_PSP_EXP.1 AVA_SOF.1	This objective is achieved by imposing developmental requirements on the design of the TSF and non-TSF components of the TOE, and on the analysis of the security functions for which strength of function claims are made. Rationales for ADV explicit requirements are described in Section 7.6. Rationales for APT explicit requirements are described in Appendix F. Since there are no strength-of-function claims associated with the security functions contained in this PP, the AVA_SOF.1 requirement only applies to the ST-specific security functions for which a strength-of-function claim is appropriate. For these security functions, this requirement ensures that the TOE developer has performed a strength-of-function analysis to ensure that these security functions meet or exceed the following: the overall minimum strength level defined in this PP (see Section 7.7) and the specific strength of function metric defined the ST.
O.SOUND_IMPLEMENTATION *The implementation of the TOE will be an accurate instantiation of its design.*	ADV_HLD_EXP.4 ADV_IMP_EXP.3 ADV_INT_EXP.3 ADV_LLD_EXP.2 ADV_RCR.3 ALC_DVS.2 ALC_FLR.3 APT_PDF_EXP.1 APT_PCT_EXP.1 APT_PSP_EXP.1 APT_PST_EXP.1	This objective is achieved by imposing developmental requirements on the implementation of the TSF and non-TSF components of the TOE to ensure that the TOE implementation is correctly created as specified by the TOE design. Rationales for ADV explicit requirements are described in Section 7.6. ALC_DVS.2 requires the developer to describe all security measures they employ to ensure the integrity and confidentiality of the TOE are maintained. In addition to showing the evidence that these security measures are followed during the development and maintenance of the TOE, the developer is also required to justify that these security measures provide the necessary level of protection. Although confidentiality may not be an issue for

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
	APT_PVA_EXP.1 ATE_COV.3 ATE_DPT.3 ATE_FUN.2 ATE_IND.3 AVA_CCA_EXP.2 AVA_VLA_EXP.4	some TOE implementations, the physical, procedural, and personnel security measures the developer uses provides an added level of assurance that the integrity of the TOE implementation is appropriately maintained. ALC_FLR.3 supports this objective by requiring the developer to track and correct flaws in the TOE, and to provide safeguards that new flaws are not created while fixing the discovered flaws. Rationales for APT explicit requirements are described in Appendix F. ATE_COV.3, ATE_DPT.3 and ATE_FUN.2 require the developer to test the TSF and analyze the test coverage as well as the depth of testing. These requirements provide the assurance that the TOE security functional requirements are correctly implemented and that the TOE implementation is a correct instantiation of both high-level design and low-level design. ATE_IND.3 provides added assurance on the rigor of the testing by requiring the evaluator to develop and run separate test suite in addition to re-testing all tests performed by the developer. The correctness of the TOE implementation can be demonstrated by a successful execution of these tests by the evaluator. Requiring the TOE to be assessed for the existence of exploitable covert channels and vulnerabilities also satisfies this objective. AVA_CCA_EXP.2 requires the developer to perform a systematic search for inter-partition covert channels. The NSA evaluator is required to confirm the results of the covert channel analysis and to selectively validate the analysis through testing. See O.COVERT_CHANNEL_ANALYSIS for the rationale on why a thorough inter-partition covert channel analysis is important. AVA_VLA_EXP.4 component is intended to provide the necessary level of confidence that vulnerabilities do not exist in the TOE that could cause the security policies to be violated. AVA_VLA_EXP.4 requires the developer to perform a systematic search for potential vulnerabilities in all the TOE deliverables, and to provide a justification that the analysis completely addresses the TOE deliverables. AVA_VLA_EXP.4 was refined to require that, in addition to the independent penetration testing and analysis performed by the evaluator, a second set of penetration testing and analysis be independently performed by the NSA evaluator. The two levels of independent testing and analysis helps to ensure that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.
O.SUBJECT_ISOLATION *The TOE will provide mechanisms to protect each subject from unauthorized*	FDP_IFC.2 FDP_IFF.1 FPT_SEP.3	This objective requires the TOE to establish security domains for subjects where each subject is completely isolated from every other subject. This complete isolation is the default configuration that is established by the TSF. Where flows between subjects are specified by the configuration data and mediated by the TSF

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
interference by other subjects.		throughout the execution session, the scope of this objective expands and must also ensure that no unauthorized information flows can occur, which may result in one subject interfering with another. FDP_IFC.2 and FDP_IFF.1 combine to define the scope of the partitioned information flow policy to be enforced by the TSF, and the rules implemented by the TSF to enforce the policy. This enforcement capability of the TSF ensures that strict isolation of a subject is preserved where no flows to/from the subject are allowed, and ensures that only authorized information flows as specified by the configuration data are allowed. FPT_SEP.3 satisfies this objective by requiring the TSF to enforce separation between the security domains of all subjects in the TSC, thus ensuring that subjects cannot access or manipulate other subject's services and resources in violation of the TSP. The security domain of a subject includes the services and exported resources that the particular subject is allowed to use.
O.TRANSITION *The TOE will provide the capabilities for an authorized subject to restart the TOE, halt the TOE and transition the TOE into maintenance mode.*	FMT_MOF.1 FMT_MSA_EXP.1 FMT_SMF.1 FPT_CFG_EXP.1 FPT_HLT_EXP.1 FPT_RST_EXP.1 FPT_MTN_EXP.1 FPT_MTN_EXP.2	FMT_MOF.1 requires the TSF to restrict the access to the TOE halt, TOE restart, and TOE transition to maintenance mode functions to those subjects that are explicitly authorized to invoke those functions. FMT_MSA_EXP.1 requires the TSF to base the assignment of subject's authorizations for TOE halt, TOE restart, and the TOE transition to maintenance mode on attributes contained in the configuration data. FMT_SMF.1 requires the TSF to implement TOE security management capabilities that include TOE halt, TOE restart and transition of the TOE to maintenance mode. FPT_CFG_EXP.1 requires an authorized subject to invoke a change in TOE configuration should the TOE provide a configuration change capability. FPT_HLT_EXP.1 requires the TOE to have the capability for an authorized subject to halt the TOE. FPT_RST_EXP.1 requires the TOE to have the capability for an authorized subject to restart the TOE. FPT_MTN_EXP.1 requires the TSF to preserve secure state when the transition to maintenance mode is from a secure state. FPT_MTN_EXP.2 requires that while in maintenance mode, the TSF prevents controlled operations from occurring if the TSF is unable to assure that a protection compromise will not occur by allowing the controlled operation to occur.
O.TRUSTED_DELIVERY *The integrity of the TOE must be protected during the initial delivery and subsequent updates, and verified to ensure*	ADO_DEL_EXP.2	ADO_DEL_EXP.2 requires the developer to provide cryptographic signature services and cryptographic hashing functions to protect the integrity of the TOE when distributing versions of the TOE to a user's site. ADO_DEL_EXP.2 also requires the developer to use independent channels to deliver the TOE code and to deliver the cryptographic keying materials used

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
that the on-site version matches the master distribution version.		to verify the distribution of the code. Cryptographic integrity check mechanisms increase assurance, i.e., prevents forging the checksum.
O.TSF_INTEGRITY *The TOE will verify the integrity of the TSF code and data.*	FMT_MOF.1 FMT_MSA_EXP.1 FPT_TST_EXP.1	FMT_MOF.1 requires the TSF to prevent a subject from being able to invoke TSF self-test unless that subject has been granted authorization to do so. FMT_MSA_EXP.1 requires the TSF to assign authorizations to subjects for the purpose of invoking TSF self-test and obtaining the results of those self-tests as specified by the configuration data. FPT_TST_EXP.1 requires the TSF to verify the integrity of TSF configuration data and TSF executable code loaded in memory. If the TSF software or TSF configuration data is corrupted, the TSF may not correctly enforce its security policies. In addition to the TSF configuration data, the ST author is required to specify the testing of other TSF data that the TSF depends on to enforce its security policies.
O.USER_GUIDANCE *The TOE will provide users with the necessary information for secure use of the TOE.*	AGD_USR.1	AGD_USR.1 satisfies this objective by requiring the developer to document the functions, interfaces and warnings available to non-administrative users of the TOE. AGD_USR.1 further requires the developer to describe all user responsibilities and assumptions necessary for secure use of the TOE.
O.VULNERABILITY_ANALYSIS_TEST *The TOE will undergo independent vulnerability analysis and penetration testing by NSA to demonstrate the design and implementation of the TOE does not allow attackers with high attack potential to violate the TOE's security policies.*	APT_PVA_EXP.1 AVA_CCA_EXP.2 AVA_MSU.3 AVA_SOF.1 AVA_VLA_EXP.4	Rationales for APT explicit requirements are described in Appendix F. AVA_CCA_EXP.2 requires both the developer and evaluator to perform a systematic search for inter-partition covert channels. See O.COVERT_CHANNEL_ANALYSIS for the rationale on why it is important to perform a thorough search for these covert channels. AVA_MSU.3 satisfies this objective by requiring the developer to provide complete, clear, consistent and reasonable administrator and user guidance documents, and to perform an analysis for any vulnerability that might be caused by unclear documentation. AVA_MSU.3 further requires the evaluator to perform independent testing to check if the provided guidance document would enable an administrator or user, with proper training, to determine if the TOE is configured correctly or incorrectly. AVA_SOF.1 requires the developer to perform an analysis of the strength of the functions on the ST-specific security functions for which a strength-of-function claim is appropriate. Security functions are implemented by security mechanisms and thus, the strength-of-function analysis is to be performed at the level of the security mechanisms. The results of the analysis can be used to determine if these ST-specific security functions have the ability to counter the anticipated threats in DoD high robustness environments. See O.SOUND_DESIGN for additional

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
		information on the required analysis. AVA_VLA_EXP.4 requires the developer 1) to perform a systematic search for vulnerabilities, 2) to document the disposition of the identified vulnerabilities, 3) and to show evidence that the identified vulnerabilities cannot be exploited in the intended environment for the TOE and that the TOE is resistant to obvious penetration attacks. AVA_VLA_EXP.4 requires the NSA evaluator to conduct independent penetration testing and an independent vulnerability analysis to ensure that the TOE is resistant to penetration attacks performed by an attacker possessing a high attack potential.

7.5 TOE Environment Requirements Rationale

133 Each of the environment security objectives identified in section 4 are addressed by one or more security requirements. Table 7.5 below provides the mapping from security requirements to security objectives, as well as a rationale that discusses how the security objective is met. Definitions are provided (in italics) below each security objective so the PP reader can reference these without having to go back to section 4.

Table 7.5. Mapping of Security Requirements for TOE Environment to Objectives

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
OE_PHYSICAL *Physical security will be provided for the TOE by the non-IT environment commensurate with the value of the IT assets protected by the TOE.*	N/A	TOE environment requirements that address this objective are outside the scope of this PP. A TOE built to conform to this PP may be vulnerable to physical attack such that the TOE is unable to protect the IT assets.
OE.SUBJECT_ALLOCATION *A properly trained trusted individual will create configuration vectors such that, for those partitions to which subjects are allocated, each partition is allocated one or more subjects (i.e., subjects with homogeneous access requirements, or subjects with heterogeneous access requirements) that are appropriate for the policy abstraction supported by the TOE.*	N/A	TOE environment requirements that address this objective are outside the scope of this PP. The trusted individual responsible for configuring the TOE must be trained to fully understand the policy abstraction(s) to be enforced by the TOE in order to correctly create the configuration vectors.

Objectives from Policies/Threats	Requirements Meeting Objectives	Rationale
OE.COVERT_CHANNELS *If the TOE has covert storage and/or timing channels, then all subjects executing on that TOE will, relative to the IT assets to which they have access, have assurance sufficient to outweigh the risk that they will violate the security policy of the TOE by using those covert channels.*	N/A	TOE environment requirements that address this objective are outside the scope of this PP. Covert storage and timing channels allowed to exist on a system are a threat to the assets to which the subjects able to exercise the covert channels and thus violate the security policy of the TOE have access. Assurance must be provided that Trojan horses and other application malware cannot attack IT assets via these covert channels.
OE.TRUSTED_FLOWS *For each configuration of the TOE, a partial order of the flows that are allowed between policy equivalence classes will be identified. Any subject allowed by the configuration data to cause information flow that is contrary to the partial order will be trusted at least with assurance commensurate with the value of the IT assets in all equivalence classes to which it has access.*	N/A	TOE environment requirements that address this objective are outside the scope of this PP. See rationale for A.TRUSTED_FLOWS.
OE.TRUSTED_INDIVIDUAL *Any individual allowed to perform procedures upon which the security of the TOE may depend must be trusted with assurance commensurate with the value of the IT assets.*	N/A	TOE environment requirements that address this objective are outside the scope of this PP. See rationale for A.TRUSTED_INDIVIDUAL.

7.6 Explicit Requirements Rationale

134 Explicit components have been included in this protection profile because the Common Criteria requirements were found to be insufficient as stated. This section includes the rationale for using explicit requirements for both the TOE and the IT environment.

7.6.1 Explicit TOE Functional Requirements

Table 7.6. Rationale for Explicit TOE Functional Requirements

Explicit Component	Rationale
FAU_SAR_EXP.1	The CC component FAU_SAR.1 was written with the expectation that authorized administrators will interact directly with the TSF to review and

Explicit Component	Rationale
	to respond to audited events. Considering that the separation kernel provides no means for direct interaction with any users, the explicit component FAU_SAR_EXP.1, derived from FAU_SAR.1, requires that recorded audit events be made available to authorized subjects. FAU_SAR_EXP.1 requires the TSF to export audit records such that audit data analysis on the information contained in the audit records can be conducted by subjects authorized to access the recorded audit information. Note that explicit assurance component AGD_ADM_EXP.1 requires the TOE developer to provide documentation of the structure of recorded audit events to aid development of analysis capabilities.
FAU_SEL_EXP.1	The CC component FAU_SEL.1 provides a selection for specifying the user identity, subject identity, object identity, host identify or event type attributes to be used to determine the auditable events that are actually audited during an execution session. Considering that the separation kernel mediates information flows between subjects and resources, the base CC FAU_SEL.1 component was modified to replace the attribute selection statement with an explicit list of attributes used by the TSF to generate audit events (subject identity, resource identity, event type, event success, event failure).
FIA_ATD_EXP.1	The base CC FIA_ATD.1 component addresses the definition of security attributes associated with individual users, where those attributes are used to enforce the TSP. The security attributes used by the separation kernel to enforce the TSP are associated with partitions and exported resources. These security attributes are contained in the configuration vector which is transformed into configuration data once the TSF has been initialized. The explicit component (FIA_ATD_EXP.1) is based on the CC FIA_ATD.1 component. It is iterated to define the specific set of security attributes that must be maintained in the configuration data for partitions, subjects, and exported resources.
FIA_USB_EXP.1	The base CC FIA_USB.1 component addresses the binding of an individual user's security attributes (as defined by FIA_ATD.1) to a subject acting on behalf of that user. The security attributes used by the separation kernel are not defined for users and are not bound to users. The explicit component (FIA_USB_EXP.1) is based on the CC FIA_USB.1 component. It is iterated to require the TSF to associate security attributes maintained in the configuration data to partitions, to subjects, and to exported resources. It also requires the TSF to apply specific rules that govern which and how configuration data security attributes are associated to partitions, subjects, and exported resources.
FMT_MSA_EXP.1	The CC FMT_MSA.1 component requires the TSF to have the ability to assign authorizations to the roles that are bound to individual users, and to enable those roles to perform security-relevant functions associated with access control and information flow control security policies. The separation kernel does not support the concept of users and roles. The separation kernel allows authorized subjects to interact with the TSF via the TSFI, based upon the subject authorizations defined by the configuration data. The explicit component (FMT_MSA_EXP.1) is based on the CC FMT_MSA.1 component and requires the TSF to assign authorizations to subjects as specified by the configuration data, and to

Explicit Component	Rationale
	provide no other means by which authorizations can be assigned to subjects.
FMT_MSA_EXP.3	The CC FMT_MSA.3 component requires the TSF to assign a restrictive default value for each attribute for which a value was not assigned by the configuration data. The function of ensuring that appropriate restrictive values are used to configure the TSF is a function normally performed at runtime by an authorized administrator. Given that the TSF must be able operate without such interaction, it is necessary for the TSF to have this ability to take appropriate action in the event of an unassigned security-relevant attribute. The separation kernel obtains attributes for all aspects of secure operation of the TOE (e.g., self test execution, audit parameters, configuration parameters) from the configuration data. The explicit component (FMT_MSA_EXP.3) is based on the portion of the CC FMT_MSA.3.1 element that addresses capability to assign restrictive default values. However, the 3.2 element is not required for this PP, as this TOE has no notion of authorization for the purpose of over-riding default values.
FMT_MCD_EXP.1	The common criteria FMT_MTD.1 component provides for the TSF to restrict the ability of authorized roles to perform operations on TSF data; it *allows* some operations on TSF data subject to restrictions specified. The separation kernel relies on the configuration data to define the initial secure state of the TSF for an execution session, and to define the behavioral properties of the TSF with respect to Partitioned Information Flow Policy enforcement and other security relevant operations that occur during an execution session. The TSFs sole dependency on the configuration data requires that modification of the configuration data be prevented to ensure that secure state be maintained throughout and between execution sessions. The explicit component FMT_MCD_EXP.1 is introduced to require the TSF to prevent modification to the configuration data. Note that authorized modification of the configuration data by the TSF, in response to input from authorized subjects, can occur. This is accomplished by the initialization and configuration change functions. The mechanism for such modifications is the alteration of the TSF internal vector set, and is an operation restricted to authorized subjects via use of designed TSFI. Furthermore, the degree to which configuration data can be changed is a function of the configuration change capability implemented by the TOE. This can range from a relatively simple capability to select one of many predefined static configurations to a more complex capability with an interface having a number of functions and parameters for a dynamic unconstrained configuration change. Refer to FPT_CFG_EXP.1 for requirements governing the authorized modification of the TSF internal vector set.
FPT_CFG_EXP.1	The requirement for the TOE to be reconfigurable is not a security requirement. As a result, the CC does not provide components that address the capability to change the TOE configuration. However, once the decision is made to implement a configuration change capability, then it becomes necessary to ensure that the TSF preserves secure state

Explicit Component	Rationale
	throughout the execution of the configuration change operation. It is envisioned that the separation kernel will be used in environments where changing the configuration of the TOE will be required. The explicit component, FPT_CFG_EXP.1, establishes one mandatory requirement and several optional requirements that govern "by design" changes to the TOE configuration through the interaction of authorized subjects with the TSF. As a mandatory requirement, the TOE is to provide support for the static configuration change (Note that this capability requires no support from the TSF as the configuration vector used to define the TSF internal vector can be changed off-line. Reference Section 3 for a discussion of this capability). As optional requirements, the TOE developer may specify the requirements for dynamic total, dynamic-constrained and dynamic-unconstrained on-line configuration change capabilities. Additionally, FPT_CFG_EXP.1 defines the requirements for the ways in which an authorized subject can select and change TSF internal vectors, and requires the TSF to preserve secure state throughout the execution of each of the configuration change capabilities implemented.
FPT_ESS_EXP.1	Since the initialization of the TOE and the establishment of the TSF in its initial secure state is not a requirement levied on the TSF, the CC does not provide components that establish requirements for secure initialization Of the TOE. The autonomous nature of the separation kernel requires trust in the initialization of the TOE and the establishment of the TSF into an initial secure state as defined by the configuration vector. Additionally, the trusted initialization mechanism must be relied on to establish the execution environment for the TSF and to transform the configuration vector into configuration data useable by the TSF. It is necessary that trust in the initialization function be established (Refer to ADV_INI_EXP.1 for those requirements) and that the TSF be able to determine that it has been properly placed in an initial secure state, per the configuration vector used during initialization, prior to allowing any information flows to occur. The explicit component FPT_ESS_EXP.1 defines the requirements for the TSF to be established in an initial secure state as defined by the configuration vector used during the initialization of the TOE; requires the TSF to determine that it is in a secure state once initialization completes; and requires that the TSF will not authorize any information flows until it has determined that it is in an initial secure state.
FPT_HLT_EXP.1	The CC does not provide components addressing the capability for an authorized subject to halt the TOE. It is envisioned that the separation kernel will be used in environments where an authorized subject can command the TOE to halt. The explicit component FPT_HLT_EXP.1 requires the TSF to halt the TOE when directed to do so by an authorized subject, and for the TSF to preserve secure state throughout the process of halting the TOE.
FPT_MTN_EXP.1	Since the TOE could transition to maintenance mode from either a secure halted state or a secure operational state, or from a non-secure operational state (as a result of the failure to complete recovery from a failure or service discontinuity), it was necessary to introduce this explicit component to address security issues related to transitioning to the

Explicit Component	Rationale
	maintenance mode. The explicit component FPT_MTN_EXP.1 requires the TSF to provide the capability for an authorized subject to request the TOE be transitioned into maintenance mode, and, requires the TSF to preserve secure state during the transition to maintenance mode if the TOE was in a secure state when the transition was requested. FPT_MTN_EXP.1.3 requires the TSF to halt the TOE if the TSF is unable to preserve secure state after transitioning to maintenance mode from a secure state.
FPT_MTN_EXP.2	FMT_MTN_EXP.1.1 requires that the TOE be able to transition to maintenance mode for the purpose of conducting and completing maintenance action on the TSF. In addition the ST author can specify in FPT_RCV_EXP.2 that the TOE is to transition to maintenance mode as a recovery action. The transition to maintenance mode can occur from either a secure or insecure state. It is necessary to ensure that, once in maintenance mode, no TSF-mediated action takes place unless the TSF is able to properly mediate that request. The explicit component FPT_MNT_EXP.2 requires the TSF to be capable of rejecting requests for controlled operations that would result in violations of the TSP while the TSF is undergoing maintenance, thereby preventing any violation of the TSP while the TSF is in a potentially insecure state.
FPT_PLP_EXP.1	The CC does not provide components addressing the capability for the TSF to execute in such a manner that the TSF internally supports the principle of least privilege. Additionally, it is inappropriate to address this requirement solely as an assurance requirement (which is the approach taken by the CC for many behavioral/property requirements, which when implemented, do not result in a TSFI). Assurance requirements exist to dictate that which must be done to provide confidence that the TOE meets some set of properties. It is therefore necessary to both define a functional property of the TSF that governs how it is expected to behave and to define corresponding assurance requirements that govern the evidence produced and the activities performed to acquire confidence that the specified functional behavior is obtained. The explicit component FPT_PLP_EXP.1 requires the TSF to have the property that each TSF function requires no more access to TSF data and other internal TSF resources than the minimum required. It should be noted that FPT_PLP_EXP.1 is considered by the authors as being the same type of requirement as Domain Separation (FPT_SEP) and Non-bypassability of the TSF (FPT_RVM). Both FPT_SEP and FPT_RVM are appropriately requirements that express functional properties to be exhibited by the TOE during execution, and existence and correctness of those properties must be demonstrated.
FPT_RCV_EXP.2	The CC component FPT_RCV_EXP.2 requires the TSF to take specific recovery action when the TSF detects that it is not in a secure state. FPT_RCV_EXP.2 also requires the TSF to attempt to halt the TOE if the TSF is unable to proceed with any recovery action.. Considering this requirement in the context of the separation kernel, it was recognized that the transition to maintenance mode could occur from the operational state or from the halt state. Additionally, it was recognized that an appropriate response to the failure to complete a recovery operation

Explicit Component	Rationale
	(e.g., simply to halt the TOE or to restart the TOE in the same or a different configuration) is best left as an implementation option for the TOE developer. The explicit component FPT_RCV_EXP.2, based on FPT_RCV.2, requires the TSF to attempt to recover the TOE to a secure state without further protection comprise when the TSF determines that it is not in a secure state after TOE initialization or at any time while the TOE is in operational mode. The TOE developer is required to specify a list of recovery conditions and their associated recovery actions. The TOE developer must also specify the recovery action to be taken by the TSF in the event the TSF is unable to initiate or complete a recovery that requires the TOE to remain in operational mode or to restart without transitioning to maintenance mode. Refer to the explicit component FPT_MTN_EXP.1 which addresses the transition to maintenance mode when directed to do so by an authorized subject.
FPT_RST_EXP.1	The CC does not provide components addressing the capability for an authorized subject to restart the TOE. It is envisioned that the separation kernel will be used in environments where an authorized subject can command the TSF to restart the TOE, either as a means to clear fault indications or for the purpose of changing the TOE configuration. The explicit component FPT_RST_EXP.1 requires the TSF to restart the TOE when directed to do so by an authorized subject, and for the TSF to preserve secure state throughout the process of restarting the TOE.
FPT_TST_EXP.1	The CC component FPT_TST.1 is written in terms of authorized user interaction with the TSF for conducting TSF self-tests and in verifying the integrity of the stored TSF code and TSF data. Invocation of self tests by an authorized entity, whether a user or a subject, external to the TSF, provides no guarantee that such self-tests will occur. Because the separation kernel is expected to be used in embedded as well as more typical systems, it was recognized that the TSF must have the capability to independently determine when to conduct a self-test and to carry out such self-tests. In addition, the TSF provides the means for an authorized subject to request that a TSF self-test be executed. The results generated by TSF self-tests should be made available to authorized subjects in a form that allows the subjects to assess and respond to the results. The explicit component FPT_TST_EXP.1 requires the TSF to run a suite of TSF self-tests to verify the correct operation of the software portions of the TSF implementation. Additionally, the TSF is required to provide the results of the TSF self-test to authorized subjects in a form that allows the authorized subject to assess the results.
FRU_PRU_EXP.1	The separation kernel is the foundational component upon which other executing resource-constrained processes will be dependent. Just as it was necessary to require the TSF internals to execute while supporting the Principle of Least Privilege for purposes of damage limitation, it was recognized that the TSF must meet its SFRs while executing in a predictable manner with respect to its use of internal resources. By doing so, the TSF is better able to mitigate erroneous behavior that results in

Explicit Component	Rationale
	unbounded use of memory or processor resources, regardless of whether such behavior is intentionally triggered by attack against the TSF, or the result of device failure. In effect, the TSF is better able to mitigate denial of service with respect to TSF internal execution. The CC does not provide components addressing the capability for the TSF to execute in a predictable manner with respect to its use of memory and processor resources. Additionally, it is inappropriate to address this requirement solely as an assurance requirement (which is the approach taken by the CC for many behavioral/property requirements, which when implemented, do not result in a TSFI). Assurance requirements exist to dictate that which must be done to provide confidence that the TOE meets some set of properties. It is therefore necessary to both define a functional property of the TSF that governs how it is expected to behave and to define corresponding assurance requirements that govern the evidence produced and the activities performed to acquire confidence that the specified functional behavior is obtained. The explicit component FPT_PRU_EXP.1 requires the TSF to execute within the bounds of a worst-case usage of memory scenario, and to execute within the bounds of a normal and worst-case processor-use scenario.

7.6.2 Explicit TOE Assurance Requirements

Table 7.7. Rationale for Explicit TOE Assurance Requirements

Explicit Component	Rationale
ADO_DEL_EXP.2	The CC ADO_DEL.2 component did not specify the use of NIST-approved cryptographic signature algorithms and keyed-hash message authentication functions to support trusted delivery of the TOE. ADO_DEL_EXP.2 was created based on the base CC component ADV_DEL.2. Elements .3D through .6D, and .2C through .4C were added to require the developer to provide documentation for trusted delivery and to demonstrate the use of NIST-validated cryptographic mechanisms in support of their trusted delivery processes. The requirement for independent delivery channels for the TOE and for keying materials provides additional assurance against undetected tampering. Element .2E was added to require the evaluator to determine that the procedures asserted as constituting a trusted delivery mechanism are sufficiently strong security mechanisms for distributing the TOE.
ADV_ARC_EXP.1	The CC does not contain a consolidated set of requirements for assurance evidence to address the TOE architecture and the manner in which it contributes to the ability of the TSF to enforce the TSP and to protect itself from tampering. The CC addresses architecture issues in the form of SFRs (FPT_SEP, FPT_RVM) and SARs (ADV_INT). For high robustness, it was recognized that new assurance criteria was necessary to require

Explicit Component	Rationale
	assurance evidence specific to the TSF architecture and its ability to protect itself, to support the principle of least privilege for the purpose of damage limitation, and to prevent TSF-internal denial of service by executing in a predictable manner. ADV_ARC_EXP.1 was created as a new ADV family to require specific evidence for establishing assurance in the TSF architecture. ADV_ARC_EXP.1 requires the developer to provide an architecture design of the TSF. Additionally ADV_ARC_EXP.1 requires evidence that the TSF is able to execute within defined bounds for its internal use of processor and memory resources as a means to reduce or eliminate the potential for a successful denial of service originating from within the TSF, and to enable system integrators to know the amount of system resources that have to be allocated to the TOE when they develop the configuration data. Several assurance elements in ADV_ARC_EXP.1 are related to SFRs. This is intentional; it is appropriate to precisely define the testable desired behavior of the TOE in terms of functional requirements and to then precisely define the assurances required (combination of evidence, analysis, and third-party IV&V) to determine that the desired behavior is achieved by the implementation.
ADV_CTD_EXP.1	During the initialization of the TOE, a configuration vector is used to determine the initial secure state of the TSF. Once the TSF has been established in its initial secure state, the TSF enforces maintains secure state by enforcing the TSP for the duration of the execution session. Therefore, it is the configuration vector that defines secure state for all TSP states during the execution session. The configuration vector is critical to the ability of the TSF to properly enforce the organizational security policy governing inter-partition information flows. The configuration vector is generated by a configuration vector generation and validation capability, i.e., the configuration tool. The configuration tool is part of the TOE but not part of the TSF, and therefore not subject to most of the ADV documentation SARs. Additionally, there is no CC assurance family that addresses the assurances for generating the configuration vector and for establishing the correctness of the configuration vector. ADV_CTD_EXP.1 requires the following: a) The configuration tool must have the ability to generate both human-readable and machine-readable forms of configuration vectors, and to be able to convert between the two. b) The tool must be able to place a cryptographic seal on a generated configuration vector, c) The TOE developer must produce documentation for the tool that describes how to interpret the various forms in which the configuration vector is produced and provides instructions for the use of the tool to include placing a cryptographic seal on a generated configuration vector.
ADV_FSP_EXP.4	For high robustness, the evaluator requires a detailed understanding of the security relevance of each TSFI in terms of its intended use and behavior; an understanding of all the parameters associated with the use of a TSFI; and an understanding of all error/exception messages that would be

Explicit Component	Rationale
	observed at the TSFI. ADV_FSP_EXP.4, which is derived from ADV_FSP.4, was created to capture these requirements. ADV_FSP.4.2D clearly states that the developer is to provide a formal presentation of the TSFI functional specification. Such was implied in CC v2.3 ADV_FSP_EXP.4.1C is a restatement of ADV_FSP.4.4C. ADV_FSP_EXP.4.2C requires a semi-formal description of the TSFI. The semi-formal description is in the form of a consistent presentation structure using a set of defined and consistently used terms. The semi-formal description is independent of, but must be consistent with, the formal presentation of the TSFI required by ADV_FSP_EXP.4.2D and ADV_FSP_EXP.4.9C, whose purpose is to provide a mathematically provable correct and consistent statement of the TSFI. ADV_FSP_EXP.4.3C, .4C, .5C, .6C, .7C, .8C combine to require descriptive information about all parameters associated with each TSFI, the operations provided by each TSFI, and all exceptions, error messages, and effects associated with each TSFI (both those associated with the invocation of the TSFI and those that originate within the TSF and utilize a TSFI to provide error, exception or effects information to an authorized subject), such that the evaluator has the basis to completely understand all aspects of each TSFI. ADV_FSP_EXP.4.9C requires a formal presentation of the TSFI which provides mathematically provable assurance that the TSFI is complete and correct. This aids in establishing assurance that given a proper implementation of the security functions, the TSP will not be violated by use of the TSFI.
ADV_HLD_EXP.4	For high robustness, the evaluator requires sufficient information to acquire an understanding of the high-level design of the TSF, in the context of the TOE, such that the security-relevance of each subsystem could be ascertained, the support provided to the TSF by the IT environment is understood, and to understand how the behavior seen at the external TSFI maps into the subsystems that make up the TSF. ADV_HLD_EXP.4, which is derived from ADV_HLD.4, was created to capture these requirements. ADV_HLD_EXP.4.1C requires the semi-formal presentation of the high-level design of the TSF to be supported by informal explanatory text to enable a more complete comprehension of the design by the evaluator. ADV_HLD_EXP.4.2C requires an informal presentation of the high-level design of the runtime non-TSF components of the TOE. This requirement is necessary to establish a design context in which the TSF can be assessed. ADV_HLD_EXP.4.3C is no change from CC v2.3. ADV_HLD_EXP.4.4C, .4.5C combine to provide the subsystem structure of the TOE, and within that, identification of the TOE subsystem components that comprise the TSF and designation of the TSF subsystems that serve in TSP-enforcing and non-TSP enforcing capacities. ADV_HLD_EXP.4.6C, .4.7C combine to provide a description of each TSF subsystem and the interactions between the TSF subsystems. This allows the evaluator to understand the structure of the TSF in context of

Explicit Component	Rationale
	the entire TOE and to understand how TSF functionality is structured across the TSF. The documentation of subsystem interfaces, effects and exceptions (CC v2.3 ADV_HLD.4.6C, .4.7C, .4.8C) is captured in the documentation of the low-level design of the modules that comprise each subsystem. The justification of the means of obtaining separation, and the identification of hardware, firmware and software portions of the TSF is contained in ADV_ARC_EXP.1.2C and ADV_ARC_EXP.1.6C.
ADV_IMP_EXP.3	For high robustness, the evaluator requires access to all forms of software and firmware implementation representations, to include having detailed documentation that aids in understanding how to transform the implementation representation into the executable implementation. ADV_IMP_EXP.3, which is derived from ADV_IMP.3, was created to capture these requirements. ADV_IMP_EXP.3.1D, .3.2D require the implementation representation to be made available to the evaluator and with it, the necessary tools and instructions. These support ADV_IMP_EXP.3.2E and allow the evaluator to establish confidence in the transformation process by conducting an independent transformation of the implementation representation into the implementation, and verifying that the results of the transformation are identical to the implementation provided by the TOE developer. ADV_IMP_EXP.3.1C is no change from ADV_IMP.3.1C. ADV_IMP_EXP.3.2C requires that the implementation representation made available to the evaluator is no different in form and content used by the TOE developer. This makes it possible for the evaluator to establish equivalence between the implementation provided by the TOE developer and the implementation created by an evaluator-invoked transformation of the implementation representation.
ADV_INI_EXP.1	The CC SFRs and SARs assume that authorized administrators interacting with the TOE during an execution session contribute to establishing assurance that the TOE is properly initialized and is correctly configured to enforce the organizational security policies. The SKPP does not make this assumption. Therefore the TOE must be able to initialize and establish a secure state autonomously, without any intervention by authorized administrators. The initialization function is responsible for trusted initialization of the TOE which includes establishing the execution environment for the TSF and establishing the TSF in a secure state consistent with the configuration vector that defines the configuration data. Since the initialization function is part of the TOE but not part of the TSF, it is not subject to most of the ADV documentation SARs. Additionally, there is no CC assurance family that requires assurances for trusted initialization of the TOE when that initialization is accomplished without the aid of authorized administrators. ADV_INI_EXP.1.1D through 1.5D require the TOE developer to provide an initialization function that maintains the integrity of the TOE while establishing the TSF in a secure state consistent with the selected configuration vector, that is able to detect and respond to faults during initialization of the TOE, and once the TOE initialization completes, the initialization function will not arbitrarily interact with the operation of the

Explicit Component	Rationale
	TSF during the execution session. ADV_INI_EXP.1.6D requires the initialization function to establish the TSF security domain and to bring the TSF software and data into that domain. ADV_INI_EXP.1.7D and .1.8D require the TOE developer to design and implement the initialization function such that other components executing on the TOE can neither circumvent nor tamper with the initialization function. ADV_INI_EXP.1.9D through .1.11D require the TOE developer to apply modular decomposition to the design and implementation of the initialization function and to provide both a functional specification and a design document for the initialization function. ADV_INI_EXP.1.12D and 1.13D require the TOE developer to test the initialization function and to provide test documentation of the test results. ADV_INI_EXP.1.1C through 1.5C levy content requirements on the functional specification of the initialization function such that the evaluator is able to acquire an understanding of the intended behavior of the initialization function at its interfaces (to include behavior in the event of errors) ADV_INI_EXP.1.6C through .1.8C require an architectural description of the components that comprise the initialization function, to include identification of those components implemented by hardware, firmware or software means, and the designation of each component as either relevant or unrelated to the establishment of a secure state that is consistent with the selected configuration vector. This information provides the evaluator with insight into the structure of the initialization function. ADV_INI_EXP.1.9C requires the developer to describe how the internals of the initialization function work together to establish the TOE in a secure state. This is necessary because the TSF can only determine that it is in "a" secure state – the combination of the configuration tool, load function, initialization function, and the TSF determine that the TSF is in the "intended" secure state. ADV_INI_EXP.1.10C requires a description of the means and methods used by the initialization function to verify that the TSF code and TSF data have not been modified subsequent to being loaded. ADV_INI_EXP.1.11C requires a description of the fault management (detection of faults/errors, error/exception handling) capabilities of the initialization function, to include identification of all faults/errors that are addressed by the initialization function. ADV_INI_EXP.1.12C requires an argument for assurance that the initialization function will not arbitrarily interact with the TSF after TOE initialization completes. ADV_INI_EXP.13C and .1.14C require an analysis of the initialization design to demonstrate that no other component executing on the TOE can "spoof" or tamper with the initialization function. ADV_INI_EXP.1.15C and 1.16C levy modularity and minimization requirements on the internal structure of the initialization function. Any inclusion of components that do not support TOE initialization must be

Explicit Component	Rationale
	justified. ADV_INI_EXP.1.17C requires that both the initialization design and the functional specification of the initialization function are presented in informal style. ADV_INI_EXP.1.18C and .1.19C state the requirements for the scope of the testing to be performed and test documentation to be provided by the TOE developer. Since FPT_ESS_EXP.1 requires that the TSF not begin to enforce the partitioned information flow policy until after it has determined that it is in a secure state, it is necessary for the above assurance to be in place so that the evaluator is able to establish confidence in the integrity of the initialization process. ADV_INI_EXP.1.1E through 1.5E require the evaluator to determine that the initialization function does in fact achieve this objective.
ADV_INT_EXP.3	For high robustness, the evaluator is required to understand TSF module behavior and to understand how the TSF modules interact and couple with each other. To enable the evaluator to apply rigorous evaluation techniques in meeting their requirements, the TOE developer must minimize the size and complexity of TSF modules; must ensure TSF modules contain no unusable code that complicates the analysis or that poses a vulnerability with respect to the execution of the TSF; must apply accepted software engineering techniques to implement modularization, layering and coupling concepts at the level of TSF modules; and must ensure the TSF modules support the principle of least privilege when executing. ADV_INT_EXP.3, derived from ADV_INT.3, was created to express these requirements. The following elements state the requirements for the measures to be taken by the developer while designing and implementing the TSF at the module level, and the evidence developed to support the evaluator in performing their required verification and analysis: ADV_INT_EXP.3.1D, .1.2D, 3.1C, .3.2C, 3.3C requires the application of software engineering principles in achieving modular decomposition of the TSF, requires documentation of the process followed by the developer to determine how the TSF is to be decomposed into modules, requires the identification of each TSF module resulting from application of the decomposition process, and requires evidence that correlates the TSF as it was decomposed, back to the decomposition process that supposedly drove the decomposition decisions made by the developer. ADV_INT_EXP.3.3D, .3.6D, .3.9D, .3.4C, .3.13C, .3.14C requires that TSF modules be designed and implemented for good internal structure, minimization of complexity, to be simple enough to be analyzed; required a description of the design as it serves to minimize complexity, requires a justification for any TSF module that deviates from internal structure and complexity coding standards, and requires a justification for any unused code or redundant code that remains in the TSF. ADV_INT_EXP.3.4D, .3.5D, .3.5C, .3.6C, .3.7C requires the design of the TSF modules to exhibit specific coupling and cohesion properties, for the developer to conduct coupling and cohesion analysis on all TSF modules, and for the developer to provide justification any each TSF module that

Explicit Component	Rationale
	does not conform to the permitted types of module coupling and module cohesion. ADV_INT_EXP.3.7D, .3.8D, .3.8C-.3.12C requires a layered design and implementation of TSF modules where interactions between layers are minimized; requires description of the layering architecture and the methodology used to determine that architecture, requires a description of the modules allocated to each layer and a description of the services provided by each layer, and a description of the flow of interactions between layers to include a justification for any layering interactions that proceed from a lower layer to a higher layer in the architecture. ADV_INT_EXP.3.10D, .3.11D, .3.15C, .3.16C require that TSF modules be designed and implemented such that the TSF itself supports the principle of least privilege, that non-policy-enforcement or supporting functionality is included in the TSF modules, requires a justification for any non-policy enforcing or supporting functionality included in the TSF, and a description of how the TSF design and implementation supports the principle of least privilege. ADV_INT_EXP.3.2E requires the evaluator to perform their own coupling and cohesion analysis on a subset of TSF modules to substantiate claims made by the TOE developer. ADV_INT_EXP.3.3E requires the evaluator to examine a subset of TSF modules to determine if their design and implementation is consistent with the coding standards used for minimization of complexity. ADV_INT_EXP.3.4E requires the evaluator to determine if the TSF design and implementation sufficiently supports the principle of least privilege. ADV_INT_EXP.3.5E requires the evaluator to confirm that the design and implementation of the TSF modules is simple enough to support the various analysis required of the evaluator.
ADV_LLD_EXP.2	For high robustness, the evaluator requires detailed TSF module design evidence focused on the functionality provided by the TSF modules. This evidence differs from that required by ADV_INT_EXP, where its focus is on minimization of complexity, and the interactions between TSF modules. The TSF module design evidence includes discussion of the TSF module interfaces and the manner in which the TSF modules can be invoked, the data used by the TSF module and data coupling between TSF module, and an algorithmic description of the module. ADV_LLD_EXP.2, derived from ADV_LLD.2, was created to express these requirements. To provide the evaluator with the evidence to acquire a complete understanding of TSF module design and its use of data: ADV_LLD_EXP.2.3C requires each TSF module to be designated as SFR-enforcing, SFR-supporting, or non-security relevant. ADV_LLD_EXP.2.4C requires discussion of common data that are common to TSF modules. ADV_LLD_EXP2.5C requires a description of each TSF module's purpose, its interfaces, and methods for invoking the TSF module, values returned by the module, and calls made by the module to other TSF

Explicit Component	Rationale
	modules. ADV_LLD_EXP2.6C requires discussion of exceptions, errors and effects of each module. ADV_LLD_EXP2.7C requires a detailed algorithmic description of each TSF module. To ensure that the implementation representation is not provided as a substitute for the low-level design of TSF modules: The following are renumbering of elements contained in the base ADV_LLD.2 component: ADV_LLD_EXP2.1D is a restatement of CC v2.3 ADV_LLD.2.1D. ADV_LLD_EXP.2.1C is a restatement of CC v2.3 ADV_LLD.2.1C with the additional requirement of supporting informal text to explain the semi-formal presentation of the low-level design. ADV_LLD_EXP2.2C is a restatement of CC v2.3 ADV_LLD.2.2C. ADV_LLD_EXP2.1E, 2.2E are restatements of CC v2.3 ADV_LLD.2.1E, 2.2E.
ADV_LTD_EXP.1	The TOE requires integration with other IT components. That integration will include packaging of the TOE in various forms that are appropriate for the intended execution environment. It is the load function that provides the means to transfer the TOE into a form and onto media that allows its subsequent use in the execution environment. The load function includes the processes and mechanisms to convert the software portion of the TSF implementation and/or configuration vector set into a TOE-useable form. For high robustness, it is required to have at least the same level of detail in the load function design as that provided in the TOE high level design. The load function is part of the TOE but not part of the TSF, and therefore not subject to most of the ADV documentation SARs. Additionally, there is no CC assurance family that addresses the assurances that are appropriate for the load function. ADV_LTD_EXP.1 was created as a new ADV family to express these requirements. ADV_LTD_EXP.1.1D, 1.2D, 1.3D, 1.4D require a TOE loader design that loads the machine-readable software portion of the TSF implementation, including the configuration vector set, in a form that is accessible by the TOE initialization mechanism, and to do so while preserving the integrity of the implementation and configuration vectors. Additionally, this design is to be implemented as a capability available to the TOE user. ADV_LTD_EXP.1.1C requires the information to be provided at the same level of detail abstraction as the high-level design. ADV_LTD_EXP.1.2C requires a description to explain how the requirements of ADV_LTD_EXP.1.3D, 1.4D are met by the design and the implementation of the design. ADV_LTD_EXP.1.1E, 1.2E require the evaluator to determine that all requirements are met in terms of content and presentation, and that the TOE loader design and TOE loader capability each meet their specific requirements.
AGD_ADM_EXP.1	The creation of numerous explicit functional and assurance requirements has impact on the contents of the guidance provided for use by the administrators of the TOE. AGD_ADM_EXP.1 was created to describe

Explicit Component	Rationale
	the additional guidance required. Requirements AGD_ADM_EXP.1.9C and .1.11C require that the developer provide guidance on how to use the configuration vector generation tool to create configuration vectors that accurately reflects the operational configuration of the TOE as used by an organization. FDP_IFF and FDP_IFC require that access to resources be controlled by the TSF at the granularity to which those resources are made available (viz., exported) to subjects. Thus, the TSF provides the ability to enforce least privilege. Requirements AGD_ADM_EXP.1.10C and .1.11C require the developer provide guidance for creating TSF configuration that conforms to the principle of least privilege, and that the configuration data, in fact, enforces least privilege. Requirement AGD_ADM_EXP.1.12C requires the developer to describe the various considerations associated with the types of subjects that are to run in partitions provided by the TOE, and the considerations to be made, based on those subject types, to determine which of the Partitioned Information Flow Policy abstraction(s) are appropriate. Requirement AGD_ADM_EXP.1.13C requires the procedures for use of the load function be documented. Requirement AGD_ADM_EXP.1.14C requires the developer to document how to use the initialization function to bring the TSF to the initial secure state. Requirement AGD_ADM_EXP.1.15C requires the developer to describe the audit record structure in sufficient detail to allow the audit data to be properly interpreted. Requirement AGD_ADM_EXP.1.2E requires the evaluator to determine that the information provided in the administrator guidance is sufficient to meet the requirements specified by AGD_ADM_EXP.1.1C through 1.12C.
AMA_AMP_EXP.1	CC v2.3 provides no assurance class to address continuity of assurance for an evaluated TOE. For high robustness it is required that the TOE developer have a plan in place, at the time of evaluation, for the maintenance of the assurances established by the TOE evaluation. The explicit component AMA_AMP_EXP.1 was written to define the requirements for the assurance maintenance plan, and is captured in its entirety as defined by the CCIMB-2003-02-001 document "Supplement: AMA – Assurance Maintenance", dated February 2003 [8]. For AMA_AMP_EXP.1.8C, conceptual qualification refers to the security analyst's understanding of security concepts relevant to the TOM.
APT_PDF_EXP.1	See Appendix F.
APT_PSP_EXP.1	See Appendix F.
APT_PCT_EXP.1	See Appendix F.
APT_PST_EXP.1	See Appendix F.

Explicit Component	Rationale
APT_PVA_EXP.1	See Appendix F.
AVA_CCA_EXP.2	The scope of AVA_CCA was reduced in AVA_CCA_EXP.2.1D to reflect the semantics of the two rules of the *partitioned information flow* SFP. In that SFP, the *partition rule* enforces restrictions on flows between partitions independent of the subject-resource pair involved in the information flow; the *subject-exported resource rule* provides the ability to further restrict those flows to be specific to each subject-resource pair. Each partition defines an equivalence class of resources, such as would be used in the TSF's application domain to instantiate programs and files at a given DoD sensitivity level. Covert channel analysis is concerned with "leaks" (i.e., unintended flows) that might occur between the equivalence classes (e.g., sensitivity levels) separated by the partitioning capabilities of the TOE. However, since least privilege policies are orthogonal to flow policies, they are outside of the scope of covert channel analysis, and so AVA_CCA_EXP.2.1D was modified to only apply to the partition abstraction policy.
AVA_VLA_EXP.4	For high robustness, NSA policy requires that evaluator actions for independent vulnerability assessment and independent penetration testing be conducted by NSA personnel. The evaluator actions of AVA_VLA.4 were changed to require the NSA evaluator to perform independent vulnerability analysis and for the NSA evaluator to conduct independent penetration testing. The requirements for the Common Criteria Test Lab (CCTL) evaluator to perform and independent vulnerability assessment to and conduct independent penetration testing have been deleted.

7.7 Rationale for Strength of Function

135 The overall TOE minimum strength of function for SFRs contained in this PP and contained in a ST which claims conformance to this PP is SOF-HIGH. The evaluated TOE is intended to operate in DoD high robustness environments and the SOF-HIGH level is consistent with the level of the anticipated threat.

136 The minimum SOF does not apply to any cryptographic mechanisms with respect to a CC evaluation (to include those contained as part of the TOE but not part of the TSF). The assessment of strength of cryptographic algorithms is outside the scope of the CC. The strength of the cryptographic mechanisms will be determined by NIST FIPS 140-2 certified modules and requirements specified in this PP; the validation of these cryptographic mechanisms will be performed by the NSA.

137 Strength of function rationale for single SFRs contained in this PP is not provided as there are no strength of function claims made for any individual SFR contained in this PP.

7.8 Rationale for Non-Applicable Dependencies

138 This section provides rationale for all component dependency conditions specified by the CC but

not satisfied by the set of components contained in this PP.

Table 7.8. Rationale for Non-Applicable Component Dependencies

Component with Non-Applicable Dependency	CC-Required Dependency	Rationale
FAU_ARP.1	FAU_SAA.1	FAU_ARP.1 was refined to require specific action be taken by the TSF based solely upon specified failures of TSF self-tests. As a result of this refinement, there is no basis for requiring the TSF to satisfy FAU_SAA.1, i.e., to conduct analysis on audited events to identify potential TSP violation conditions. Furthermore, the TSF is not required to maintain audit information and thus it is not possible for the TSF to perform the audit analysis as required by FAU_SAA.1.

7.9 Rationale for Assurance Rating

139 This protection profile has been developed for U.S. Government high robustness environments. The TOE environment and the value of information processed within this environment (i.e., highly sensitive) establishes the basis for the set of CC-based and explicit security assurance requirements that are contained in this protection profile. As such, no EAL claim is made by this protection profile.

8. References

[1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model, CCIMB-2005-08-001, Version 2.3, August 2005.

[2] Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements, CCIMB-2005-08-002, Version 2.3, August 2005.

[3] Common Criteria for Information Technology Security Evaluation, Part 3: Security Assurance Requirements, CCIMB-2005-08-003, Version 2.3, August 2005.

[4] Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, CCIMB-2005-08-004, Version 2.3, August 2005.

[5] National Computer Security Center, Department of Defense Trusted Computer System Evaluation Criteria DoD 5200.28-STD, December 1985.

[6] National Security Telecommunications and Information Systems Security Committee, National Information Systems Security (INFOSEC) Glossary, NSTISSI No. 4009, September 2000.

[7] Harrison, M., Ruzzo, W. and Ullman, J., "On Protection in Operating Systems," Communications of the ACM, vol. 19, no. 8, August 1976, pp. 461-471.

[8] Common Criteria for Information Technology Security Evaluation, Supplement: AMA – Assurance Maintenance, CCIMB-2003-02-001, Version 0.9, February 2003, DRAFT.

[9] Denning, D. E., "A Lattice Model of Secure Information Flow," Communications of the ACM, vol. 19, no. 5, May 1976, pp. 236-243.

[10] Rushby, J., "Design and Verification of Secure Systems," ACM Operating Systems Review, vol.15, no.5, December 1981, p.12.

[11] UK IT Security Evaluation and Certification Scheme, UK CC Interpretation – UK/2.2/008, "Treatment of commercial hardware that is part of a TOE," 25 February 2005.

[12] Saltzer, J. H. and Schroeder, M. D., "The Protection of Information in Operating Systems," Proceedings of the IEEE. 63(9):1278-1308. 1975.

Appendix A - Acronyms

ANSI	American National Standards Institute
CC	Common Criteria for Information Technology Security Evaluation Version 2.3
COTS	Commercial-Off-The-Shelf
CVS	Configuration Vector Set
DoD	Department of Defense
EAL	Evaluation Assurance Level
FIPS	Federal Information Processing Standard
IA	Information Assurance
IT	Information Technology
NIST	National Institute of Standards and Technology
PA	Partition-to-partition Authorizations
PIFP	Partitioned Information Flow Policy
PoLP	Principle of Least Privilege
PP	Protection Profile
RIP	Residual Information Protection
RNG	Random Number Generator
SA	Subject-exported resource Authorizations
SF	Security Function
SFP	Security Function Policy
SFR	Security Function Requirement
SK	Separation Kernel
SKPP	U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness
ST	Security Target
TOE	Target of Evaluation
TOM	Target of Maintenance
TSC	TSF Scope of Control
TSF	TOE Security Functions
TSFI	TSF Interface
TSP	TOE Security Policy

Appendix B - Cryptographic Standards, Policies, and Other Publications

Standards

ANSI X9.31-1998	American National Standards Institute (ANSI) X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), [http://webstore.ansi.org/ansidocstore].
ANSI X9.42-2001	American National Standards Institute (ANSI) X9.42-2001 (2001), Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, (http://webstore.ansi.org/ansidocstore).
ANSI X9.62-1998	American National Standards Institute (ANSI) X9.62-1-1998 (10 Oct 1999), Public Key Cryptography for the Financial Services Industry: the Elliptic Curve Digital Signature Algorithm (ECDSA), (http://webstore.ansi.org/ansidocstore).
FIPS PUB 180-2	National Institute of Standards and Technology, Secure Hash Standard, Federal Information Processing Standard Publication (FIPS-PUB) 180-2, dated 1 August 2002, [http://cs-www.ncsl.nist.gov/publications/fips/fips180-2/fips180-2.pdf].
FIPS PUB 186-2	National Institute of Standards and Technology, Digital Signature Standard, Federal Information Processing Standard Publication (FIPS-PUB) 186-2, dated 2000 January 27, [http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf].
FIPS PUB 198	National Institute of Standards and Technology, The Keyed-Hash Message Authentication Code (HMAC), Federal Information Processing Standard Publication (FIPS-PUB) 198, dated March 6, 2002, [http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf].

Appendix C – Rationale for Two-Level Policy

This Appendix provides a discussion for the dual-level Least Privilege Abstraction of the Partitioned Information Flow Policy, in which FDP_IFC and FDP_IFF explicitly define both *partition* and *subject-exported resource* rules.

The Partition Policy Must Be Explicit

140 It is possible to formulate the subject-exported resource rules such that the partition rules could be derived by examination of the former, making the latter redundant. However, since the SKPP is intended to support partitioned information policy semantics at the product configuration interface, a partition policy is explicit.

141 Additionally, one of the primary intended uses of the SKPP is to evaluate products that provide separation of resource equivalence classes that may be mapped to, for example, MLS sensitivity levels. Covert channels between such equivalence classes are a significant concern. Therefore it is necessary for covert channel requirements to be stated at the level of partition rules. If FDP_IFC and FDP_IFF were written without concern for partitions, then it would be inconsistent to state covert channel requirements at the partition level.

The Subject-Exported Resource Policy Must Be Explicit

142 If a system cannot restrict individual subjects to have only the access authorizations that they require to complete their functions, the accountability mechanism (viz., audit) will be less able to accurately capture the source of various actions, e.g., individual modifications within a file. Thus, the ability of a secure system to realize the goals of accountability, as well as the confinement of damage, is limited by the level of granularity with which the system is able to invoke the principle of least privilege. To provide high assurance, SKPP requirements for least privilege apply at the same granularity as the resources that are exported, i.e., at both the partition and subject-exported resource levels of abstraction.

143 Even if there is only one subject in a partition, the TSF must still ensure that the subject can only access those resources in its address space: it is the premise of separation that the TSF controls what resources the subject accesses. It would be circular to say that, since separation is provided, the TSF need not be able to control a subject's access to individual resources. Similarly, least privilege regarding exported resources should not be difficult for an implementation of a separation kernel, since the kernel will need to control the composition of each subject's address space to achieve the required separation properties.

144 For example, suppose there is one subject in a given partition, and the partition includes several exported resources. Then to avoid all-or-nothing security, the subject should be given only the modes of access to each of those resources that it requires, as opposed to a blanket (e.g., maximal) access to all of the resources in the partition. This is why "super-user" privileges are not allowed in secure systems. However, this restriction may be difficult to express or

understand if FDP_IFC and FDP_IFF do not articulate requirements at the subject-exported resource level.

Appendix D – Rationale for Secure State

145 The definition of "secure state" for this TOE is based on two separate properties: (*A*) that the TSF is capable of enforcing the security policy (i.e., its own data and mechanisms are intact); and (*B*) that exported resources are correctly separated (e.g., application data, and related descendants and copies, are associated with the correct partition according to the configuration data).

146 For example, if a TOE security function "breaks," so that the TOE can no longer enforce the security policy, then the TOE is not in a secure state. Conceivably, upon detection of the problem the TOE could enter maintenance mode and repair the function, and then return to normal mode in a secure state;[12] or the TOE could be shut down, repaired offline, and then started again to return to a secure state. In either case, item *1* in the definition of "secure state" is achieved, corresponding to property *A*. However, if before entering maintenance mode (or shutting down) this service failure resulted in the "pollution" of a partition with data from another partition that is not allowed to flow there by the security policy,[13] then repair of the broken function is (necessary but) not sufficient to return the TOE to a secure state.

147 Item *2* in the definition of "secure state" addresses property *B*. Continuing with the above example, item *2b* reflects the possibility that the TOE is designed so that the individual effects of operations that violate the policy – ones performed while the TOE is in an insecure state – can be "undone," in a transactional sense. Systems that do not have this *rollback* capability can have a problem much like a single drop of dye in a glass of water: undoing the effects of "pollution" is difficult to achieve. In such a case, the TOE could achieve the same rollback result required by *2b* by disabling the polluted partition or through re-initialization, per *2c*. The latter bears some discussion.

148 It is axiomatic to the SKPP concept of security, and to systems that enforce policies based on equivalence classes of resources in general, that the data that determines the initial assignment of resources to equivalence classes (viz., SKPP partitions) is *semantically correct*. In other words, the association of resources and partitions (Section 5.3.2 User-Subject Binding (FIA_USB)) as well as the related rules regarding flows (Sections 5.2.1 and 5.2.2) according to the configuration data reflect the intention of the data owner(s) (e.g., see O.CORRECT_CONFIG in Section 4.1). These requirements are levied on the TSF. Whether or not the TOE initialization mechanism has the capability to assess this semantic correctness, including determining if any partitions have been "polluted" in a previous execution session, is beyond the scope of this PP. However, if the TSF is able to detect that a partition has been polluted, the SKPP requires that the association of resources with partitions is once again made consistent with the configuration data (by definition of secure state). An acceptable, although not required, recovery action is to make all resources within the polluted partition(s) unavailable.

---

[12] If maintenance mode is entered when the TOE is in a secure state, the operations will either be consistent with *2a*, or will be outside of the scope of the model (e.g., TSF may be required to take recovery actions that override the security policy).

[13] For example, using information sensitivity labels mapped to partitions, high confidentiality or low integrity resources have been put into a low confidentiality or high integrity partition.

Appendix E – TSF Data Description

149 There are various types of TSF Data, for example: internal data structures, configuration data, and TSF-generated data. Configuration data includes flow policy and non-flow policy data. Some or all configuration data may be imported from the IT environment during system initialization. The TSF generates some data, such as audit records and digital signatures. The TSF may export certain TSF Data, including generated data, configuration data, and other implementation-dependent TSF Data.

150 Examples of TSF data are, Internal TSF Structures, Configuration Data and TSF-Generated Data:

A. Internal TSF Structures
    1. Hardware registers
    2. Software data structures

B. Configuration Data
    1. Flow Policy Configuration Data
        a. Least Privilege Flow Configuration Data
        b. Partition Flow Configuration Data
    2. Non-Flow Policy Configuration Data
        c. Audit Configuration Parameters
        d. General Configuration Parameters
            i. Clock Settings
            ii. Self-Test Periods

C. TSF-Generated Data
    1. Subject and resource policy-enforcement attributes
    2. Audit Output (e.g., audit records)
    3. Clock Output

Appendix F – Example TOE Scenario

151 For the configuration data and TOE implementation components, Figure F-1 provides a notional illustration of an acceptable scenario for their generation, movement and use, as well as the allocation of components to the TSF and TOE.

152 For simplicity and generality, the entity that develops or modifies a TOE component is called a "TOE developer," even if some of the development is performed by an entity that is an integrator or customer in some other scenario. If a component (e.g., a new hardware dependent module) is to be integrated into the TOE, then the component, as well as the combination of that component with the rest of the TOE will need to have been evaluated. Also, implementation components will always need to be accepted by the TOE trusted delivery mechanism. For example, if an "integrator," receives a TOE from the original developer and then modifies certain TOE components as part of the integration of the SK into a larger component/system – conceptually creating a new TOE – evaluation or re-evaluation of the new TOE must occur. It is beyond the scope of this protection profile to specify requirements that are specific to a partial re-evaluation of the new TOE.

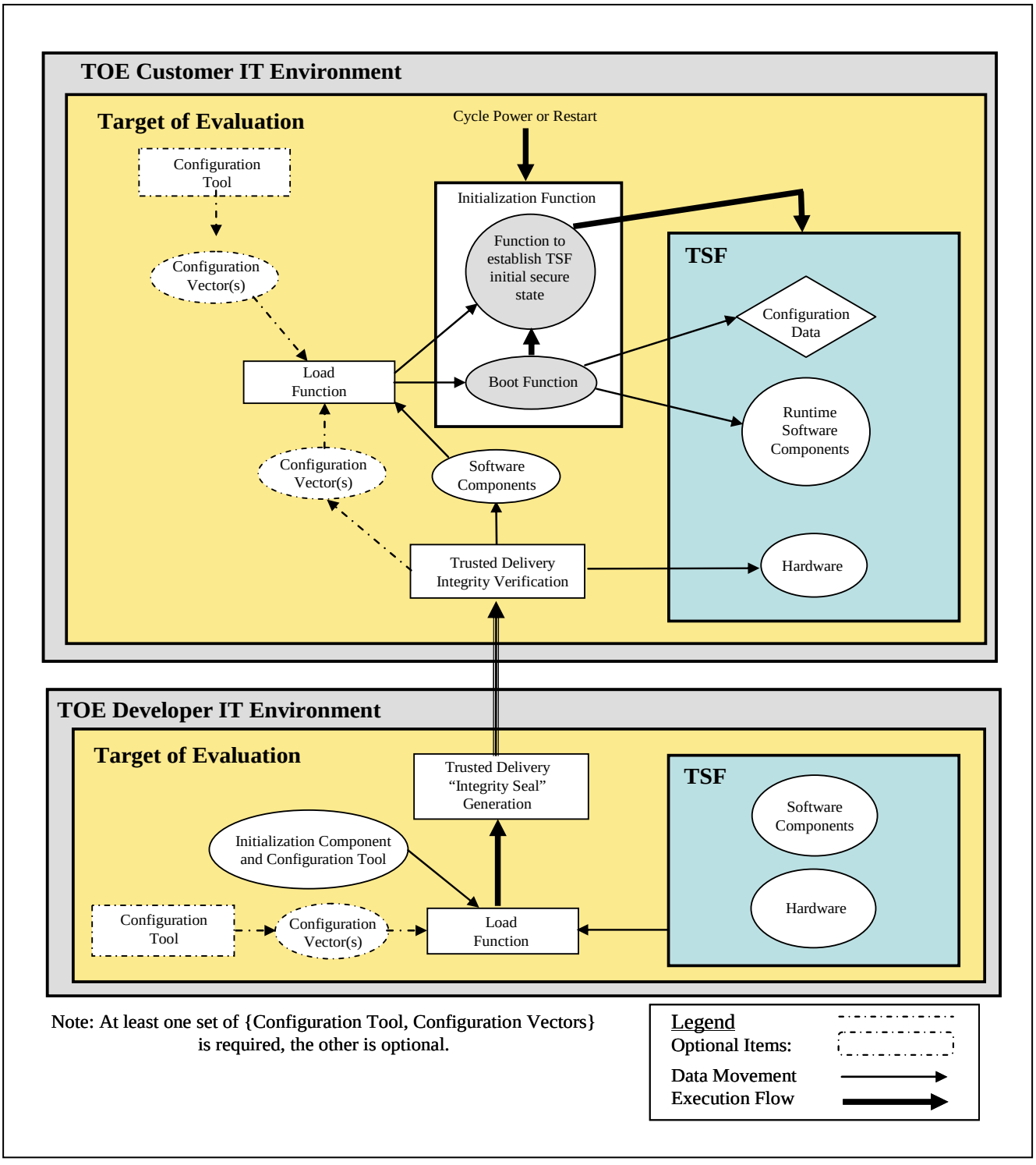


Figure F-1. Example TOE Scenario

Appendix G – Rationale for Class APT Platform Assurance

153 This Appendix provides explanatory material for the explicit Class APT requirements.

154 It is up to the TOE vendor to define the platform components, as well as the degree of openness of the platform definition.

1. Rationale for Class APT

Objective

155 The families in the "Platform Assurance" class specify assurance requirements that provide confidence that commercial off-the-shelf (COTS) platform components used to implement the operation of a TOE are capable of effectively supporting the TOE's security functions.

156 For purposes of this class, platform components are defined *by specification*, as supplied in the developer's Platform Definition Document, allowing an end-user to identify and procure platform components that will act correctly as components of the TOE.

Application notes

157 These families are intended for use with TOEs that rely on mass-produced, non-specialized platform components. These families are a substitute for ADV, ATE, and other assurance requirements for such platform components, allowing acceptable platform components to be included in the TOE.

158 To the extent that specialized platform components are required to implement the TOE's security functions, or that specialized mechanisms within those components are required to satisfy physical protection requirements such as FPT_PHP or anti-tampering requirements, the hardware mechanisms used to satisfy those requirements cannot be considered part of the platform and must be evaluated in accordance with the non-platform assurance requirements (e.g., ADV).

2. APT_PDF_EXP — Platform Definition

Objective

159 This family states requirements for how the platform is defined, in terms of component types and component properties. This definition is contained in a specific document called the *platform definition*, and the platform definition is required to be available to potential product end-users in the same manner as the security target.

Component levelling

160 The components in this family are levelled based on the degree of detail required in the platform definition. Although not specified in this PP, the basic component is intended to be satisfied by existing standard commercial practices in terms of defining "compatible" platform components and the next level component requires explicit security analysis of the definition, an activity specific to a particular evaluation. This PP mandates the highest level which requires that detailed specifications for all components be available.

Application notes

161 The APT_PDF_EXP family follows the model of UK interpretation #008 [11] in requiring a definition of platform components (by type) and an analysis of each component type against the SFRs and architectural properties that it upholds.

3. APT_PSP_EXP — Platform Specification

Objective

162 The specification of platform component interfaces allows analysis of the TSF for functional correctness and supports analysis for vulnerabilities exercised through those interfaces.

Component levelling

163 Although not specified in this PP, the basic component simply requires that interface specifications be identified for external platform interfaces only and the next level component requires that the specification be well-defined and complete. This PP requires the highest level component which adds internal interfaces to the required specifications.

Application note

164 The expectation is that the interface specifications called for by this family are those provided by the manufacturer/supplier of each platform component, or, where a single platform component (such as a computer) comprises multiple elements (e.g., a CPU, a network interface), the manufacturer of the individual platform element.

4. APT_PCT_EXP — Platform Conformance Testing

Objective

165 Platform conformance testing is the process by which platform components are determined to be acceptable as part of a valid platform for the TOE.

Component levelling

166 The components in this family are levelled on the basis of the level of effort devoted to ensuring that components are acceptable in the TOE.

Application notes

167 For the lowest (casual) level, the intent is that it be satisfied by any platform component that allows the product to work well enough to run some level of exposure testing. The assumption is that otherwise-compatible platform components are unlikely to introduce subtle security problems while apparently functioning well overall—and in any case, if they do, APT_PST_EXP.1 should pick them up.

168 The other two levels require more rigorous testing. In both cases, testing is expected to be performed through the TSF interface, with an argument made about how those tests exercise the platform features—as opposed to APT_PST_EXP, which requires testing specific platform interfaces at the platform component interface level. This PP only requires testing of all security features identified in the platform component security analysis. The highest level requires testing of all platform interfaces.

5. APT_PST_EXP — Platform Security Testing

Objective

169 Platform security testing verifies that all external platform interfaces, and those internal platform interfaces used by the TOE function correctly and are resistant to attack.

Component levelling

170 This family has two components: one addressing only the external platform interfaces, and one addressing all external platform interfaces, and those internal platform interfaces used by the TOE. This PP requires the second level.

Application notes

171 This family may require tests that run directly on the platform, rather than under control of the TSF. The intent of this class is to make deterministic tests of the platform mechanisms rather than relying on test coverage arguments at the TSFI level.

6. APT_PVA_EXP — Platform Vulnerability Assessment

Objective

172 This component provides explicit requirements for considering platform component interfaces in satisfying the applicable AVA_VLA requirements.

Component levelling

173 There are two levels for the APT_PVA_EXP family: one that addresses only external platform interfaces, and one that addresses all external platform interfaces, and those internal platform interfaces used by the TOE. This PP requires the second level.

Application notes

174 There are no specific AVA requirements for hardware and firmware. This is just a requirement that hardware and firmware be considered.