Green Hills Software

INTEGRITY-178B Separation Kernel

Security Target

Version 4.2

Prepared for:

Green Hills Software, Inc.

34125 US Hwy 19 North
Suite 100
Palm Harbor, FL 34684
USA

Prepared By:

Science Applications International Corporation

Common Criteria Testing Laboratory

7125 6841 Benjamin Franklin Dr.
Columbia, MD 21046

Document Control Information	
Number:	IN-ICR750-0402-GH01ST
Date:	May 31, 2010

LIST OF TABLES

1. Security Target Introduction

This section identifies the Security Target (ST) and Target of Evaluation (TOE) identification, ST conventions, ST conformance claims, and the ST organization. The TOE is Green Hills Software INTEGRITY-178B Separation Kernel provided by Green Hills Software, Inc.

The TOE is a separation kernel designed to instantiate and separate partitions that serve to host custom applications. The TOE manages access to memory, devices, communications and processor resources to ensure that partitions can be entirely separated and can interact only in well defined ways configured by System Architects.

The TOE is an embedded real time operating system, in that it does not include operating system constructs such as a file system, shell prompt, or user logins. It does schedule partitions to execute on the actual hardware and provides granular scheduling capability to entities (i.e., tasks) operating within a given partition.

The Security Target contains the following additional sections:

- TOE Description (Section 2)
- Security Environment (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8).

1.1 Security Target, TOE and CC Identification

ST Title – Green Hills Software INTEGRITY-178B Separation Kernel Security Target

ST Version – Version 4.2

ST Date – 31 May 2010

TOE Identification – INTEGRITY-178B Separation Kernel, comprising:

- INTEGRITY-178B Real Time Operating System (RTOS), version IN-ICR750-0402-GH01_Rel
- Compact PCI card, version CPN 944-2021-021 w/PowerPC, version 750CXe

TOE Developer – Green Hills Software, Inc.

Evaluation Sponsor – Green Hills Software, Inc.

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 2.3, August 2005.

1.2 Conformance Claims

This TOE is conformant to the following CC specifications:

- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Requirements, Version 2.3, August 2005, CCMB-2005-08-002
    - Part 2 Extended

- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Requirements, Version 2.3, August 2005, CCMB-2005-08-003.
    - Part 3 Extended, including the following CC Part 3 requirements: ACM_AUT.2; ACM_CAP.5; ACM_SCP.3; ADO_IGS.1; ADV_RCR.3; ADV_SPM.3; AGD_USR.1; ALC_DVS.2; ALC_FLR.3; ALC_LCD.2; ALC_TAT.3; ATE_COV.3; ATE_DPT.3; ATE_FUN.2; ATE_IND.3; AVA_MSU.3; AVA_SOF.1.
- US Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03, 29 June 2007.

1.3 Conventions and Terminology

1.3.1 Conventions

This Security Target reproduces the security requirements specified in the Separation Kernels Protection Profile (Separation Kernels PP), including the formatting conventions used in the Separation Kernels PP. These conventions are described in Section 1.4 of the Separation Kernels PP. Where the Security Target completes assignment and selection operations left incomplete in the Separation Kernels PP, or performs tailoring in the form of refinements, the following conventions are used:

- Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that in cases where a selection operation is combined with an assignment operation and the assignment is null, the assignment operation is simply deleted leaving on the completed selection to identify the combination of operations.
- Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
- Refinements are indicated by bold for additions and strike-through for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").

1.3.2 Terminology

Section 1.5 of the Separation Kernels PP provides a detailed glossary of terms related to the Separation Kernels PP and conformant TOEs. The following additional terminology is used within this ST.

jitter — This refers to the delay in scheduling a partition for execution that can arise if an API call that traps into the kernel is made just prior to the partition switch. The amount of delay is dependent on the API that is called and how soon after the call the current scheduling time window expires.

System Architect — Specific term for an authorized administrator who creates the static configuration file defining the partitions of the system, the subjects and resources allocated to each partition, and the rules for sharing information between partitions. See also *Administrator, authorized* in Section 1.5 of the Separation Kernels PP.

2. TOE Description

The Target of Evaluation (TOE) is Green Hills Software INTEGRITY-178B Separation Kernel.

2.1 TOE Overview

The TOE is a separation kernel designed to instantiate and separate partitions that serve to host custom applications. The TOE manages access to memory, devices, communication resources, and processor resources to ensure that partitions are entirely separated and can interact only in well defined manners configured by a System Architect.

The System Architect creates a static configuration file that defines the partitions of the system, the subjects (i.e., sets of tasks) and resources (such as memory objects, links, connections and clocks) allocated to each partition, and the rules for sharing of information between partitions, at the granularity of subjects and resources. The configuration file also defines the mechanism by which the TSF schedules partitions and their corresponding tasks to execute.

Each partition provides an environment for a multi-tasking application. Applications communicate with the kernel and with applications in other partitions via a well-defined kernel API. The TOE is an object-based operating system. In order to communicate with the kernel or (via the kernel) an application in another partition, an application invokes an API specific to the target object type. The application uses the API to pass an object reference to the kernel. The kernel operates on the referenced object.

The TOE includes capabilities for task creation within a partition at run-time. With this capability, a wider range of multi-tasking applications can be developed, including applications that require full Ada, full C++, and/or SCA POSIX run-time support. In addition, the TOE includes capabilities required by support libraries, such as file systems and network stacks, such that these support libraries can be provided as middleware (i.e., resident in a virtual address space rather than included in the kernel) available for application use.

2.2 TOE Architecture

The TOE comprises the INTEGRITY-178B real time operating system (RTOS) running on an embedded PowerPC processor on a Compact PCI card. The card plugs into its IT environment via the PCI bus, but other than drawing power from that bus it has no security dependency on the bus or other devices connected to it. Devices on the bus, or devices that can be installed on the embedded card directly, can be made available to partitions, although the TOE itself does not include any device drivers. Access to such devices can be provided to partitions by mapping their control and data registers to memory regions in a given partition and device drivers can be implemented outside the TOE in the partitions as necessary. Alternately, development of restricted device drivers that partially run in privileged mode is included in the scope of ratings maintenance changes. Procedures for ensuring changes are compliant within the scope of ratings maintenance are described in the rating maintenance plans discussed in Section 6.2.6. For the evaluated configuration, device drivers that run in privileged mode were not included.

The INTEGRITY-178B RTOS comprises the following architectural components:

- Common Kernel
- Hardware Dependent Components, comprising:
    - Architecture Support Package (ASP), which provides a processor-independent interface between the Common Kernel and the underlying processor
    - Board Support Package (BSP), which provides a board-independent interface between the Common Kernel and any peripheral hardware (which may include devices in the processor) . The BSP can be provided by either the user or by Green Hills Software. A Green Hills Software-supplied BSP is supported in the evaluated configuration.
- Kernel API, which provides the interface between applications running in a partition and the Common Kernel. The Kernel API is linked in with the application.

INTEGRITY-178B is an object-based operating system. The object types supported by INTEGRITY-178B are as follows:

- AddressSpace, which defines a partition and supports task management
- Task, which supports task management
- MemoryRegion, which supports memory management
- Link, which supports access management
- IODevice, which supports I/O management
- Connection, which supports synchronous and asynchronous communications
- Activity, which supports asynchronous communications and task management
- Semaphore, which supports task synchronization
- Clock, which supports time management.

The AddressSpace, MemoryRegion, Link, IODevice and Connection objects are strictly static objects. All instances of these objects that are required by the applications executing on the TOE are defined and allocated in configuration data that is defined off-line and loaded onto the processor with the operating system. The Task, Activity, Semaphore and Clock objects can be static or dynamic, with the number of dynamic objects that can be created restricted by the amount of memory allocated to the partition. Static objects can be shared between partitions by the use of the Link object. Dynamic objects are not shareable.

The configuration comprises the following main sections:

- Target—provides information about the target machine for the application, including: minimum and maximum memory addresses; minimum and maximum priority any task can have; and default values for task priority, memory region size, and task stack size
- Kernel—defines any tasks to be run in physical memory (i.e., kernel space). The System Architect can specify maximum priorities for kernel tasks, default sizes for MemoryRegions declared in kernel space, the objects to be created in kernel space, and partition scheduling directives for kernel space
- AddressSpace—one or more sections, each of which defines one of the application's AddressSpaces. The System Architect can specify configuration data for each AddressSpace and for the following objects within an AddressSpace[1]:
    - AddressSpace configuration—maximum priority for each task in the AddressSpace; partition scheduling directives for the AddressSpace
    - Task configuration—size of task's stack section; task's beginning and maximum weight;
    - MemoryRegion configuration—MemoryRegion's start address and length
    - Semaphore configuration—initial value and priority
    - Activity configuration—priority
- Schedule—contains information used to implement partition scheduling. Each physical or virtual AddressSpace with a Schedule section is considered a partition. The System Architect can configure the length (in seconds) of the cycle for all partitions, called the major frame period, and each partition's offset and running time within the major frame period, as well as the maximum priority of tasks to be executed during idle time

The TOE uses the hardware protection mechanisms of the PowerPC, i.e., two-state architecture and memory protections, to ensure that partitions are effectively separated and as such must request access to controlled resources of the TOE so that appropriate decisions can be made. The TOE also uses other hardware protection mechanisms of

[1] This list is not complete, but only addresses configuration options related to resource utilization. A full description of all the configuration data options is provided in the *Integrate User's Guide*.

the PowerPC to ensure least-privilege (e.g., read/write/no execute) can be specified for a partition's memory allocation.

The TOE software is loaded into the embedded card using a dedicated interface where the software is flashed (along with its configuration data) into bootable memory on the card. This interface is assumed to be physically protected against modification or other inappropriate tampering.

2.2.1 Physical Boundaries

Physically, the TOE consists of the INTEGRITY-178B Separation Kernel and its hosting embedded PowerPC processor and Compact PCI card. The card interfaces with its IT environment via a Compact PCI bus where only power is necessary to support the operation of the TOE. Otherwise, the TOE supports partitions that interact on a logical basis.

2.2.2 Logical Boundaries

Logically, the TOE supports partitions for the execution of custom applications and controls access to memory, devices, communication, other control constructs and process resources to ensure that partitions are appropriately separated.

2.2.2.1 Security audit

The TOE is capable of auditing security relevant events. The audit trail is maintained in a circular kernel memory buffer and can be accessed via an application in a partition through an IODevice object designed specifically for this purpose. Each audit event identifies at least the responsible entity, affected object, operation, current timestamp, and success or failure. Note that when the TOE detects a failure within itself, it is designed to shut down.

2.2.2.2 User data protection

The TOE instantiates partitions and allows them to interact only through specifically configured mechanisms (e.g., shared memory regions or communication objects). The separation extends even to processor resources where partitions can be given fixed blocks of guaranteed processing time, or can be combined into groups that can share blocks of time, to effectively manage the affect that one partition can have on another in this regard. Note that all shareable resources are statically configured and cannot be reallocated once the TOE is in an operational state.

2.2.2.3 Identification and authentication

The TOE maintains unique identification of partitions and defined resources so that they can be unambiguously associated.

2.2.2.4 Security management

The TOE is intended for use as an embedded component with no capability for direct interaction between authorized individuals and the TSF during run-time. Therefore, it does not provide for security management roles, identification and authentication of individuals or association of authenticated users with security management roles. All security management functionality is achieved through the allocation of appropriate authorizations to partitions and subjects in the TSF configuration data.

2.2.2.5 Protection of the TSF

The TOE includes self tests that can run at boot-time and as scheduled by the System Architect to ensure the underlying hardware and aspects of the TOE (e.g., integrity of code and read-only data using SHA-1) are working correctly. A System Architect deploying the TOE can use built-in hooks to perform specific functions during boot up and in the event of identified problems. Note that most failures would be catastrophic in nature and the system would reset and would subsequently come back up in a secure state based on the previously "flashed" TOE configuration.

2.2.2.6 Resource utilization

Both memory and processor resources are among the resources that can be assigned to partitions. Specific memory regions are assigned and the partition cannot acquire more memory without the TOE being reconfigured while off-line. As for processor resources, a given partition can be given a specific block of guaranteed processor time or can be pooled with other partitions to share blocks of processor time, at the discretion of the System Architect when configuring the TOE while off-line.

2.3 TOE Documentation

There are a number of guides that help users utilize the TOE effectively. Section 6.2 of the Security Target identifies documents that are applicable to the evaluation.

3. Security Environment

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 Organizational Security Policies

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.

3.2 Threats

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.

3.3 Assumptions

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.[2]
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.

[2] The TOE is allowed to be configured with multiple partitions representing a single policy equivalence class, and 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 Separation Kernel PP 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 of the Separation Kernel PP for further discussion of rationale for this assumption.

4. Security Objectives

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 Security Objectives for the TOE

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 Security Objectives for the Environment

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[3]. 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.

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

5. IT Security Requirements

5.1 TOE Security Functional Requirements

This section contains the requirements for the TOE trusted security functions (TSF). 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.

The following table identifies the SFRs that are satisfied by the INTEGRITY-178B Separation Kernel.

Requirement Class	Requirement Component
FAU: Security audit	FAU_ARP.1: Security alarms
	FAU_GEN.1: Audit data generation
	FAU_SAR_EXP.1: Explicit: Audit Review
	FAU_SEL_EXP.1: Explicit: Selective Audit
FDP: User data protection	FDP_IFC.2: Complete information flow control
	FDP_IFF.1: Simple security attributes
	FDP_IFF.3: Limited illicit information flows
	FDP_RIP.2: Full Residual Information Protection
FIA: Identification and authentication	FIA_ATD_EXP.1(1): Explicit: User Attribute Definition (for partition attributes)
	FIA_ATD_EXP.1(2): Explicit: User Attribute Definition (for subject attributes)
	FIA_ATD_EXP.1(3): Explicit: User Attribute Definition (for non-subject exported resource attributes)
	FIA_USB_EXP.1(1): Explicit: User-Subject Binding (for partition attribute bindings)
	FIA_USB_EXP.1(2): Explicit: User-Subject Binding (for subject attribute bindings)
	FIA_USB_EXP.1(3): Explicit: User-Subject Binding (for non-subject exported resource attribute bindings)

Requirement Class	Requirement Component
FMT: Security management	FMT_MCD_EXP.1: Explicit: Management of Configuration Data
	FMT_MOF.1(1): Management of Security Functions Behavior (to change the TOE configuration)
	FMT_MOF.1(2): Management of Security Functions Behavior (to restart the TOE)
	FMT_MOF.1(3): Management of Security Functions Behavior (to halt the TOE)
	FMT_MOF.1(4): Management of Security Functions Behavior (to initiate TOE self-tests)
	FMT_MOF.1(5): Management of Security Functions Behavior (to transition the TOE to maintenance mode)
	FMT_MOF.1(6): Management of Security Functions Behavior (to manage the audit function)
	FMT_MSA_EXP.1: Explicit: Management of Security Attributes
	FMT_MSA_EXP.3: Explicit: Static Policy Attribute Initialization
	FMT_MTD.1(1): Management of TSF Data (for obtaining TSF self-test results)
	FMT_MTD.1(2): Management of TSF Data (for obtaining audit information)
	FMT_MTD.3: Secure TSF data
	FMT_SMF.1: Specification of Management Functions
FPT: Protection of the TSF	FPT_AMT.1: Abstract machine testing
	FPT_CFG_EXP.1: Explicit: Configuration Change
	FPT_ESS_EXP.1: Explicit: Establishment of Secure State
	FPT_FLS.1: Failure with preservation of secure state
	FPT_HLT_EXP.1: Explicit: TOE Halt
	FPT_MTN_EXP.1: Explicit: TOE Maintenance
	FPT_MTN_EXP.2: Explicit: TOE Maintenance Secure
	FPT_PLP_EXP.1: Explicit: TSF Least Privilege
	FPT_RCV_EXP.2: Explicit: Automated recovery
	FPT_RCV.4: Function recovery
	FPT_RST_EXP.1: Explicit: TOE Restart
	FPT_RVM.1: Non-bypassability of the TSP
	FPT_SEP.3: Complete reference monitor
	FPT_STM.1: Reliable time stamps
	FPT_TST_EXP.1: Explicit: TSF Testing
FRU: Resource utilization	FRU_RSA.2: Minimum and Maximum Quotas
	FRU_PRU_EXP.1: Explicit: TSF Predictable Resource Utilization

Table 1: TOE Security Functional Components

5.1.1 Security audit (FAU)

5.1.1.1 Security Alarms (FAU_ARP.1)

FAU_ARP.1.1 **Refinement:** The TSF shall take [**action to enter maintenance mode**] upon detection of **any failure of the tests defined in FPT_AMT.1 and FPT_TST.1**[1].

5.1.1.2 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1-NIAP-0407 Refinement: 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 2; and**

d) [***no additional 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
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
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_CFG_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 FPT_TST.1 tests • Other TSF failures specified in the assignment statement of FPT_FLS.1.1b
Explicit: TOE Halt (FPT_HLT_EXP.1)	(None)

Security Functional Requirement	Audit events prompted by requirement
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 TSP (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
Minimum and Maximum Quotas (FRU_RSA.2)	• Attempts to exceed memory quota • Attempts to exceed processing time quota
Explicit: TSF Predictable Resource Utilization (FRU_PRU_EXP.1)	(None)

Table 2: Auditable Events

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.**

5.1.1.3 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.

5.1.1.4 Explicit: 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) [***no additional attributes***].

5.1.2 User data protection (FDP)

5.1.2.1 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].

5.1.2.2 Simple Security Attributes (FDP_IFF.1)

FDP_IFF.1.1-NIAP-0407 Refinement: The TSF shall enforce the **Partitioned Information Flow SFP as a [*Partition 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.~~

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.

5.1.2.3 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 **[a maximum capacity that is controlled by the manner in which the System Architect configures the scheduling algorithms and permitted caching policies, controls partition jitter, and controls exceptions]**.

5.1.2.4 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 [***deallocation of the resource***][6].

5.1.3 Identification and authentication (FIA)

5.1.3.1 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
- [***no other partition security attributes***].

5.1.3.2 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
- [***no other subject security attributes***].

5.1.3.3 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
- [***no other non-subject exported resource security attributes***].

5.1.3.4 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
- [***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) **[attributes are associated as defined in the static configuration file]**.

FIA_USB_EXP.1.3(1) The TSF shall enforce the following rules governing changes to the configuration data security attributes with partitions: **[no changes can be made to the attributes while the TOE is operating]**.

5.1.3.5 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
- [***no other subject security attributes***].

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) **[attributes are associated as defined in the static configuration file]**.

FIA_USB_EXP.1.3(2) The TSF shall enforce the following rules governing changes to the configuration data security attributes associated with subjects: **[no changes can be made to the attributes while the TOE is operating]**.

5.1.3.6 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
- [***no other non-subject exported resource security attributes***].

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) **[attributes are associated as defined in the static configuration file]**.

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: **[no changes can be made to the attributes while the TOE is operating]**.

5.1.4 Security management (FMT)

5.1.4.1 Explicit: Management of Configuration Data (FMT_MCD_EXP.1)

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

5.1.4.2 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 of the TOE** to **authorized subjects**.[7]

5.1.4.3 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]

5.1.4.4 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]

5.1.4.5 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]

5.1.4.6 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]

5.1.4.7 Management of Security Functions Behavior (to manage the audit function) (FMT_MOF.1(6))

FMT_MOF.1.1(6) The TSF shall restrict the ability to **[*disable, enable*]** the functions **[audit function]** to **[authorized subjects].**

5.1.4.8 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,
- **[no other authorizations]**.

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

5.1.4.9 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.

5.1.4.10 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**.

5.1.4.11 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**.

5.1.4.12 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]

5.1.4.13 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,**
- **[enable and disable the audit function]**

5.1.5 Protection of the TSF (FPT)

5.1.5.1 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 [at no other times]** to demonstrate the correct operation of the security assumptions provided by the abstract machine that underlies the **software portions of the** TSF.[13]

5.1.5.2 Configuration Change (FPT_CFG_EXP.1)

FPT_CFG_EXP.1.1 The TSF shall provide **[*no configuration change capability*]**.

~~**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;~~
~~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;~~

~~**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.

5.1.5.3 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).

5.1.5.4 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) **[failures of the initial or continual AMT tests, failures of the initial or continual self tests]**

b) **[power failures, partition initialization error]**.

5.1.5.5 Explicit: TOE Halt (FPT_HLT_EXP.1)

FPT_HLT_EXP.1.1 When requested by **[*an authorized subject executing on the TOE*]**, the TSF shall halt the TOE.

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

5.1.5.6 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.

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.

5.1.5.7 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.

5.1.5.8 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.

5.1.5.9 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: **[**

- **Partition initialization error:** ***halt the TOE without initiating any recovery action***
- **Initial AMT failures, continuous AMT failures, initial self-test failures, continuous self-test failures[4]:** ***transition the TOE to maintenance mode and initiate recovery action while in maintenance mode***
- **Power failure:** ***initiate recovery action that results in a restart of the TOE without transitioning to maintenance mode*****]**

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 **[*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 **[*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.

[4] See section Abstract Machine and TSF Testing (FPT_AMT.1, FPT_TST_EXP.1) for a better understanding of these terms.

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.

5.1.5.10 Function Recovery (FPT_RCV.4)

FPT_RCV.4.1 The TSF shall ensure that **[in the event of a**

- **Continuous AMT failure,**
- **Continuous self-test failure, or**
- **Power failure**

the TSF will] have the property that the SF either completes successfully, or for the indicated failure scenarios, recovers to a consistent and secure state.

5.1.5.11 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.

5.1.5.12 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.

5.1.5.13 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]

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]

5.1.5.14 Reliable Time Stamps (FPT_STM.1)

FPT_STM.1.1 **Refinement:** The TSF shall be able to provide reliable time stamps for its own use **that meet [a frequency of 60 Hz and are monotonically increasing from the time the TOE initializes]**.[16]

5.1.5.15 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 **[[*at no other times*]]** to demonstrate the correct operation of the software portion of the TSF implementation.

FPT_TST_EXP.1.2 The TSF suite of self tests shall verify the integrity of TSF configuration data and **[no other data]**.

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.

5.1.6 Resource utilization (FRU)

5.1.6.1 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: [the TSF supports a virtual memory model, so the maximum memory that can be allocated to a partition is governed by the maximum memory address supported by the underlying processor],**
- **Processing time: [the TSF ensures each partition is restricted to the time window within the partition schedule specified for it in the configuration data].**[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: [the amount required to support the objects defined for the partition in the configuration data],**
- **Processing time: [the TSF ensures each partition is allocated the time window within the partition schedule specified for it in the configuration data].**[18]

5.1.6.2 Explicit: TSF Predictable Resource Utilization (FRU_PRU_EXP.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.

5.2 TOE Security Assurance Requirements

This section contains the security assurance requirements for the TOE. The requirements contained in this section are either selected from Part 3 of the CC or are explicitly stated in accordance with the CC rules for explicitly stated requirements.

The Separation Kernels PP claims that the combination of assurance components is equivalent to an Evaluation Assurance Level 6 with augmentation (EAL6+). This ST does not claim conformance to EAL6+, because of the large number of explicitly stated assurance requirements specified in the Separation Kernels PP. The ST author leaves it to the Separation Kernels PP to justify any claims for EAL conformance. This ST claims conformance to the Separation Kernels PP.

The following table identifies the SARs that are satisfied for the INTEGRITY-178B Separation Kernel.

Requirement Class	Requirement Component
ACM: Configuration Management	ACM_AUT.2: Complete CM Automation
	ACM_CAP.5: Advanced Support
	ACM_SCP.3: Development Tools CM Coverage
ADO: Delivery and Operation	ADO_DEL_EXP.2: Explicit: Detection of Modification
	ADO_IGS.1: Installation, Generation, and Start-Up Procedures
ADV: Development	ADV_ARC_EXP.1: Explicit: Architectural design
	ADV_CTD_EXP.1: Explicit: Configuration Tool Design
	ADV_FSP_EXP.4: Explicit: Formal Functional Specification
	ADV_HLD_EXP.4: Explicit: Semiformal High-Level Explanation
	ADV_IMP_EXP.3: Explicit: Structured Implementation of the TSF
	ADV_INI_EXP.1: Explicit: Trusted initialization
	ADV_INT_EXP.3: Explicit: Minimization of Complexity
	ADV_LLD_EXP.2: Explicit: Semiformal Low-Level Design
	ADV_LTD_EXP.1: Explicit: Load Tool Design

Requirement Class	Requirement Component
	ADV_RCR.3: Formal Correspondence Demonstration
	ADV_SPM.3: Formal TOE Security Policy Model
AGD: Guidance Documents	AGD_ADM_EXP.1: Explicit: Administrator Guidance
	AGD_USR.1: User Guidance
ALC: Life Cycle Support	ALC_DVS.2: Sufficiency of Security Measures
	ALC_FLR.3: Systematic Flaw Remediation
	ALC_LCD.2: Standardized Life-Cycle Model
	ALC_TAT.3: Compliance with Implementation Standards – All Parts
AMA: Maintenance of Assurance	AMA_AMP_EXP.1: Explicit: Assurance Maintenance Plan
APT: Platform Assurance	APT_PDF_EXP.1: Explicit: Specified Platform Definition
	APT_PSP_EXP.1: Explicit: Complete Platform Specification
	APT_PCT_EXP.1: Explicit: Tested Platform Conformance
	APT_PST_EXP.1: Explicit: Comprehensive Platform Security Testing
	APT_PVA_EXP.1: Explicit: Comprehensive Platform Vulnerability Assessment
ATE: Tests	ATE_COV.3: Rigorous Analysis of Coverage
	ATE_DPT.3: Testing: Implementation Representation
	ATE_FUN.2: Ordered Functional Testing
	ATE_IND.3: Independent Testing - Complete
AVA: Vulnerability Assessment	AVA_CCA_EXP.2: Explicit: Systematic Covert Channel Analysis
	AVA_MSU.3: Analysis and Testing for Insecure States
	AVA_SOF.1: Strength of TOE Security Function Evaluation
	AVA_VLA_EXP.4: Explicit: Highly Resistant

Table 3 TOE Security Assurance Components

5.2.1 Configuration Management (ACM)

5.2.1.1 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 meets all requirements for content and presentation of evidence.

5.2.1.2 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.5.4.C 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.

5.2.1.3 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.

5.2.2 Delivery and Operation (ADO)

5.2.2.1 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.
ADO_DEL_EXP.2.2D The developer shall use the delivery procedures.
ADO_DEL_EXP.2.3D The developer shall use **[*cryptographic signature*]** 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.
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 **[RSA *Digital Signature Algorithm (rDSA with odd e) with a key size (modulus) of 2048 bits or greater*]** that meets the following:

~~a) Case: Digital Signature Algorithm~~

~~FIPS PUB 186-2[5], 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[6];~~

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

ANSI X9.31-1998 (May 1998), Digital Signatures Using Reversible Public Key Cryptography For The Financial Services Industry (rDSA)[7];

~~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 (ECDSA)~~

~~**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.

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.

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.

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

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

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

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

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

[5] 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.

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

[7] See previous footnote.

5.2.3 Development (ADV)

5.2.3.1 Explicit: Architectural Design (ADV_ARC_EXP.1)

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.

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.

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.

5.2.3.2 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.

5.2.3.3 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.
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.

5.2.3.4 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.
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.
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.

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

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.

5.2.3.6 Explicit: Trusted Initialization (ADV_INI_EXP.1)

ADV_INI_EXP.1.1D	The developer shall provide a 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.
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.
ADV_INI_EXP.1.5D	The TOE initialization function shall not be able to arbitrarily interact with the TSF after TOE initialization completes.
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.

5.2.3.7 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.

5.2.3.8 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.

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.

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.

5.2.3.9 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.

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.1.3C 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.

5.2.3.10 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.

5.2.3.11 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.[19]
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.[20]
ADV_SPM.3.6C	**Refinement:** The proof of correspondence between the TSP model and the **formal presentation of the** functional specification shall be formal.[21]
ADV_SPM.3.1E	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.4 Guidance Documents (AGD)

5.2.4.1 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.
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.
AGD_ADM_EXP.1.12C	The administrator guidance shall document procedures necessary to securely load the TSF code and configuration vectors.
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.

5.2.4.2 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.

5.2.5 Life Cycle Support (ALC)

5.2.5.1 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.

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

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

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

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

5.2.5.2 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 documentation 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.

5.2.5.3 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.

5.2.5.4 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.

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.

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.

5.2.6 Maintenance of Assurance (AMA)

5.2.6.1 Explicit: Assurance Maintenance Plan (AMA_AMP_EXP.1)

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.

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.

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.

5.2.7 Platform Assurance (APT)

5.2.7.1 Explicit: Specified Platform Definition (APT_PDF_EXP.1)

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

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-off-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.

5.2.7.2 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.

5.2.7.3 Explicit: Tested Platform Conformance (APT_PCT_EXP.1)

APT_PCT_EXP.1.1D For each type of commercial-off-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.

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.

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.

5.2.7.4 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.

5.2.7.5 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.

5.2.8 Tests (ATE)

5.2.8.1 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.

5.2.8.2 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.

5.2.8.3 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.

5.2.8.4 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.

5.2.9 Vulnerability assessment (AVA)

5.2.9.1 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.

5.2.9.2 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 a manner that is insecure.

5.2.9.3 Strength of TOE Security Function Evaluation (AVA_SOF.1)

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

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

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

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

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

5.2.9.4 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.

6. TOE Summary Specification

This chapter describes the TOE security functions and associated assurance measures.

6.1 TOE Security Functions

6.1.1 Security Audit

By default, the TOE does not record audit events. The System Architect enables the auditing capability by defining an optional configuration function as part of the system configuration. Auditing is enabled when the function is defined and returns a non-zero value.

The TOE provides access to the capabilities of the Security Audit security function through various IODevice objects. The System Architect grants authorization to subjects to access the security function capabilities by granting access to the appropriate IODevice objects in the static configuration file.

6.1.1.1 Audit Data Generation (FAU_GEN.1)

The TSF can generate audit records of the following auditable events:

- Audit logging enabled
- Audit logging disabled
- TOE startup
- Initial Abstract Machine Tests bypassed
- Initial Abstract Machine Tests failed
- Initial Abstract Machine Tests succeeded
- Continuous Abstract Machine Tests started
- Continuous Abstract Machine Tests failed
- Initial TSF self tests bypassed
- Initial TSF self tests failed
- Initial TSF self tests succeeded
- Continuous TSF self tests started
- Continuous TSF self tests failed
- Attempted memory access violation
- Instruction violation
- Message transfer error
- Connection object information flow established
- IODevice object information flow established
- Link object information flow established
- Memory Region object information flow established
- Mode change requested
- Mode change completed

- AddressSpace restart requested
- AddressSpace restart completed
- Secure state commanded
- TSF data value rejected
- TOE configuration error
- TOE operational error
- TOE initialization completed
- Dynamic object creation requested (rejected and successful).

The TOE auditable events satisfy the requirements for generation of identified auditable events as identified in the following table, which maps the TOE auditable events to the SFRs and event descriptions as specified in FAU_GEN.1.

SFR	Audit event prompted by SFR	TOE auditable event
FAU_ARP.1	Actions taken due to failure of TSF self tests and tests defined in FPT_AMT.1.1	Secure state commanded
FDP_IFF.1	Denial of requested operation	• Attempted memory access violation • Instruction violation • Message transfer error • Dynamic object creation requested (rejected and successful)
FDP_IFF.3	The use of identified illicit information flow channels	• Message transfer error • TOE initialization completed (indicates if partition scheduler is not being used, which would allow for the exploitation of certain covert timing channels)
FIA_USB_EXP.1	Unsuccessful binding of security attributes to individual partitions, subjects, non-subject exported resources	• Initial TSF self tests failed • TSF data value rejected
FMT_MSA_EXP.3	Any TSF assignment of a restrictive default value	Not audited – restrictive default values are not explicitly assigned
FMT_MTD.3	Rejection of specified values for TSF data	TSF data value rejected
FPT_AMT.1	Failures detected by tests of the underlying abstract machine and the results of the tests	• Initial Abstract Machine Tests failed • Continuous Abstract Machine tests failed
FPT_CFG_EXP.1	All requests for a configuration change	Not audited – the TOE does not provide a capability for dynamic configuration changes

SFR	Audit event prompted by SFR	TOE auditable event
FPT_ESS_EXP.1	Startup of the TOE, i.e., successful and unsuccessful establishment of secure state	• TOE Initialization • TOE startup
FPT_FLS.1	Failures detected by the FPT_AMT.1 and FPT_TST.1 tests	• Initial Abstract Machine Tests failed • Continuous Abstract Machine tests failed • Initial TSF self tests failed • Continuous TSF self tests failed
	Other TSF failures specified in the assignment statement of FPT_FLS.1.1b	TOE configuration error
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	Secure state commanded
FPT_RCV_EXP.2	TOE condition that causes the TSF to be in an insecure state	• TOE configuration error • Initial Abstract Machine Tests failed • Continuous Abstract Machine tests failed • Initial TSF self tests failed • Continuous TSF self tests failed
	Action taken to attempt to recover the TOE to a secure state	Secure state commanded
FPT_RCV.4	The inability of the TOE to return to a secure state after failure of a security function	Secure state commanded
	The detection of a failure of a security function	Secure state commanded
FPT_STM.1	Changes to the TSF-internal time source	Not audited – the TOE does not provide a capability to change the time of the high-resolution timer used for time stamps
FPT_TST_EXP.1	Failures of TSF self tests and the results of the tests	• Initial TSF self tests failed • Continuous TSF self tests failed
FRU_RSA.2	Attempts to exceed memory quota	Not audited – the TOE does not provide a capability to allocate additional memory during run-time

SFR	Audit event prompted by SFR	TOE auditable event
	Attempts to exceed processing time quota	Not audited – the TOE does not provide a capability to allocate additional processing time during run-time

The security audit function records the following information in each audit record:

- Timestamp
- Type of event
- Subject identity
- Outcome of the event (success or failure)
- Resource identity.

6.1.1.2 Audit Review (FAU_SAR_EXP.1)

The TSF exports audit records by providing authorized subjects the capability to read an audit record from the audit event log. The audit event log comprises a circular buffer in kernel memory. The TSF provides an IODevice object that allows authorized subjects to read audit records from the audit event log. Audit records have a fixed format whose structure is available to applications executing on the TOE. In this way, the exported audit records are provided to authorized subjects in a manner suitable for the authorized subject to interpret the information contained in the audit record. Once read, the audit record is removed from the audit event log. If the audit event log fills up, the TSF's default action is to overwrite the oldest record.

6.1.1.3 Selective Audit (FAU_SEL.1)

The TSF provides the capability to select which auditable events will be audited, based on the following attributes:

- Resource identity
- Subject identity
- Event type
- Success of auditable security event
- Failure of auditable security event.

The System Architect can define in a static configuration file an Audit Log Exclusion Table that defines the auditable events that are to be excluded from being audited by the TSF. The TSF uses the entries in the Audit Log Exclusion Table to determine if an audit record for an auditable event is to be generated.

6.1.1.4 Security Alarms (FAU_ARP.1)

Any time the TSF detects a failure that requires halting system operation (including failure of the Abstract Machine Tests and TSF Self Tests), it enters maintenance mode. Task scheduling is not performed within this state.

6.1.1.5 Security Audit Summary

The Security Audit function is designed to satisfy the following security functional requirements:

- FAU_ARP.1: If the TSF detects any failure that requires halting the system (including failure of the Abstract Machine Tests and the TSF self-tests), it will halt system scheduling and enter maintenance mode.
- FAU_GEN.1: The TSF is able to generate audit records that include start-up and shutdown of the audit function, start-up and shutdown of the TOE, and all the auditable events identified in Table 2 of the ST.

The generated audit records contain a timestamp for the audited event, the type of event, the subject identity, the resource identity and the outcome of the event.

- FAU_SAR_EXP.1: The TSF provides the capability to export audit records to authorized subjects in a form that the authorized subject is able to interpret.
- FAU_SEL_EXP.1: The TSF provides the ability to select which events are to be audited, based on resource identity, subject identity, event type and outcome (success or failure) of the event.

6.1.2 User Data Protection

The TOE is an object-based operating system. Every resource in the TOE is represented by an Object. The object types supported by the TOE are as follows:

- AddressSpace, which defines a partition and supports task management
- Task, which is an active entity (subject) that supports task management
- MemoryRegion, which supports memory management
- Link, which supports access management
- IODevice, which supports I/O management
- Connection, which supports synchronous and asynchronous communications
- Activity, which supports asynchronous communications and task management
- Semaphore, which supports task synchronization
- Clock, which supports time management.

The AddressSpace, MemoryRegion, Link, IODevice and Connection objects are strictly static objects. All instances of these objects that are required by the applications executing on the TOE are defined and allocated in configuration data that is defined off-line and loaded onto the processor with the operating system. The Task, Activity, Semaphore and Clock objects can be static or dynamic, with the number of dynamic objects that can be created restricted by the amount of memory allocated to the partition. Static objects can be shared between partitions by the use of the Link object. Dynamic objects are not shareable.

An AddressSpace represents an individual space of memory addresses for application execution. The operating system kernel runs in a physical AddressSpace and the TOE utilizes the capabilities of the PowerPC memory management unit to manage virtual AddressSpaces. An AddressSpace is allocated processor time for the execution of its tasks in a partition time window schedule, which is defined in the configuration data. The schedule defines the offset and duration within the schedule sequence that the AddressSpace's tasks can execute.

A Task is an individual execution thread within an AddressSpace. Multiple Tasks can be defined to run within each AddressSpace. A Task has full access to the memory and objects within its corresponding AddressSpace.

6.1.2.1 Partitioned Information Flow SFP (FDP_IFC.2, FDP_IFF.1)

The TOE implements the Partitioned Information Flow SFP to control information flows between subjects and exported resources. The TOE enforces the Partitioned Information Flow SFP as a Partition Abstraction, as defined in Section 2.3.2 of the Separation Kernels PP. In this abstraction, the subjects in a partition have homogenous requirements for access, on a per-partition basis, to exported resources.

A virtual AddressSpace is assigned resources (e.g., memory, data) and authorized information flows (e.g., shared memory, Connections, Links to IODevice resources). An AddressSpace has a single identifier that represents both the partition and bounded subject. Thus, the TOE enforces seperation at the virtual AddressSpace boundary. Within an AddressSpace, the active elements are the Tasks (and corresponding Activity objects). The set of Tasks and corresponding Activity objects associated with an AddressSpace are collectively the "Subject". All Tasks associated with an AddressSpace have identical access to the AddressSpace's assigned resources.

The TOE implements a partition time window scheduler, which is used to assign execution time windows to one or more AddressSpaces. Each TOE "partition" is an execution time window within a major frame. In this context, the "partition" is a time resource with all assigned AddressSpaces having the same access to the time window. Access to the time resource is explicitly controlled through the configuration data (assigned on a per AddressSpace basis).

All of the static objects defined in the configuration data, plus any dynamic objects that might be created by executing tasks, constitute the exported resources of the TOE.

The TOE configuration data defines:

- The AddressSpaces
- The objects belonging to each AddressSpace that must be statically defined, comprising the AddressSpace's MemoryRegion, Link, IODevice, and Connection Objects
- The objects belonging to each AddressSpace that may be statically defined, comprising the AddressSpace's static Task, Activity, Semaphore, and Clock Objects
- Each AddressSpace's time allocations in the partition time window schedule
- Each AddressSpace's access to IODevices and physical memory
- Authorized information flows between AddressSpaces using Connection Objects and MemoryRegion Objects
- Authorized access to Objects defined in other AddressSpaces using Link Objects.

The TOE defines an API that specifies all the operations that can be performed on Objects. Operations are specific to Object types. Each function in the API results in a call into the kernel to perform the requested operation. The kernel determines that the caller is authorized to perform the requested operation.

The configuration data specifies the partitions that will exist and be supported by the TOE, and for each defined partition its static objects. Each partition and each subject and resource within each partition is identified, along with the rules for sharing resources between partitions. These rules represent the authorized information flows. The TOE enforces the Partitioned Information Flow SFP by ensuring each attempt to pass information between partitions is allowed by the defined resource sharing rules. If a controlled operation would result in an information flow not explicitly authorized by the configured information flow rules, then the operation is denied.

6.1.2.2 Limitation of Covert Channels (FDP_IFF.3)

The implementation of the Partitioned Information Flow SFP ensures that no covert storage channels can exist between partitions. The only storage resources that can be shared between partitions are those that are statically defined and specifically authorized for sharing in the configuration data. Although an application within a partition can create certain objects (Task, Activity, Semaphore and Clock objects) dynamically, these dynamic objects cannot be shared between partitions. Therefore, since only static objects can be shared, and all sharing of static objects between partitions is explicitly authorized in the configuration data, there are no covert storage channels between partitions in the TOE.

The TSF limits the capacity of covert timing channels between partitions to a maximum that is controlled by the manner in which the System Architect configures the scheduling algorithm and permitted caching policies, and controls partition jitter and exceptions.

6.1.2.3 Residual Information Protection (FDP_RIP.2)

When a resource is deallocated, the TSF sanitizes the contents of the resource to ensure the information content of the resource is not available when the resource is re-allocated.

6.1.2.4 User Data Protection Summary

The User Data Protection function is designed to satisfy the following security functional requirements:

- FDP_IFC.2: The TSF enforces the Partitioned Information Flow SFP as a Partition Abstraction on all configured partitions, subjects and exported resources and all operations that cause information to flow between subjects and exported resources.
- FDP_IFF.1: The TSF explicitly authorizes information flows between partitions, subjects and exported resources based on the identity of the involved partitions, subjects and exported resources and the authorized flows explicitly defined for them in the configuration data. If a controlled operation would result in a flow not explicitly authorized in the configuration data, the operation is denied. In addition, an attempted information flow is denied if the target resource has been disconnected.
- FDP_IFF.3: The TSF prevents covert storage channels between partitions and limits the capacity of covert timing channels.
- FDP_RIP.2: The TSF sanitizes the contents of a resource when the resource is deallocated.

6.1.3 Identification and Authentication

The TSF configuration data defines the following security attributes for each partition, as represented by an AddressSpace and its allocated objects:

- Identity—each AddressSpace has its own unique identity, defined by the static configuration data
- Minimum and maximum memory quota—the memory allocated to an AddressSpace is static, so the minimum and maximum quota is the same
- Minimum and maximum quotas for processing time—when using Partition Scheduling, the minimum and maximum processing time quota is the same – the System Architect allocates the execution time windows for each AddressSpace, and when the AddressSpace is active its CPU resource is guaranteed
- Information flow authorizations—the static configuration data defines the subjects (i.e., tasks) and objects allocated to each AddressSpace, and the allowed operations between the subjects and objects, which define the authorized information flows (i.e., any information flow that is not specifically defined by Object allocations in the static configuration data is., by definition, unauthorized).

The TSF configuration data defines the following security attributes for each subject (i.e., one or more Tasks and corresponding Activity objects allocated to an AddressSpace):

- Identity of the subject—each Task object has its own unique identity, defined in the static configuration data or defined by the TOE at run-time
- Identity of the partition to which the subject is bound—each Task object is allocated to a single AddressSpace, as defined in the static configuration data
- Subject authorizations, Information flow authorizations—the static configuration data defines the Objects allocated to each AddressSpace and the allowed operations that define the authorized information flows.

The TSF configuration data defines the following security attributes for each non-subject exported resource:

- Identity of the exported resource—each Object has its own unique identity, defined in the static configuration data
- Identity of the partition to which the exported resource is bound—each Object is allocated to a single AddressSpace, as defined in the static configuration data
- Information flow authorizations—the static configuration data defines the Objects allocated to each AddressSpace and the allowed operations on Objects that define the authorized information flows.

All non-subject exported resources (indeed, all resources) in the TOE are represented by Objects. Objects are owned on a per-AddressSpace basis. Each AddressSpace has its own Object Table containing the Objects that belong to it. An Object can be directly accessed only from that AddressSpace. The TOE provides the Link Object to allow Objects to be shared across AddressSpaces (on a per AddressSpace basis).

The TSF configuration data is defined in a file called the Integrate Input Configuration File or Intex File. This file is used as input to the Green Hills Software Integrate tool, which produces the TOE image to be loaded onto the target processor. The configuration data is used by the Integrate tool to produce a number of Boot Tables. The Boot Tables are processed during initialization and are used to create the AddressSpaces, Page Tables, memory segments and initial objects assigned to each partition. Initialization processing also ensures the appropriate authorizations, as defined in the configuration data, are associated with each partition, subject and resource.

The Identification and Authentication function is designed to satisfy the following security functional requirements:

- FIA_ATD_EXP.1(1)-(3): As indicated above, the TSF associates the appropriate security attributes with all partitions, subjects and resources defined in the TSF configuration data.
- FIA_USB_EXP.1(1)-(3): As indicated above, during TOE initialization the Boot Tables are processed and used to create the appropriate bindings of security attributes with partitions, subjects and resources as defined in the TSF configuration data.

6.1.4 Security Management

6.1.4.1 Security Management Functions (FMT_SMF.1, FMT_MOF.1)

The TSF is able to perform the following security management functions:

- Restart the TOE – the TSF provides an authorized subject the ability to restart the TOE
- Halt the TOE – the TSF provides an authorized subject the ability to halt the TOE
- Conduct TSF self-tests – the TSF provides Abstract Machine Tests and TSF self tests that are executed automatically during initialization and can also be executed continuously by authorized subjects
- Transition the TOE to maintenance mode – the TSF provides an authorized subject the capability to transition the TOE to maintenance mode
- Enable and disable the TOE audit function – the TSF provides an authorized subject the capability to enable and disable the audit function.

As the TOE is intended for use as an embedded component with no capability for direct interaction between authorized individuals and the TSF during run-time, it does not provide for security management roles, identification and authentication of individuals, or association of authenticated users with security management roles. All security management functionality is achieved through the allocation of appropriate authorizations to partitions and subjects in the TSF configuration data.

6.1.4.2 Security Management Authorizations and Restrictions (FMT_MSA_EXP.1, FMT_MOF.1, FMT_MTD.1)

The System Architect can allocate the following authorizations to subjects in the configuration data:

- Invoke a TOE halt, Invoke a TOE restart, Enter maintenance mode – authorize the subject to access the Commanded Shutdown IODevice object, which allows the authorized subject to transition the TOE to the secure maintenance state while specifying the reason for the transition (halt, reset, maintenance, or secure state)
- Invoke TSF self-tests – authorize the subject to execute the Abstract Machine and TSF self-tests
- Obtain results of TSF self-tests – the results of TSF self-tests are written as audit records to the audit event log, so a subject authorized to read audit records is authorized to obtain results of TSF self-tests
- Access audit records – authorize the subject to read audit records by granting the subject read access to an IODevice object provided for this purpose
- Enable and disable auditing during run-time – authorize the subject to access the audit log enable/disable IODevice Object provided for this purpose.

The TSF assigns these authorizations to subjects as specified in the configuration data and restricts the ability to perform these actions to appropriately authorized subjects. Since the TOE does not provide a capability for dynamic configuration changes, there is no authorization to invoke a configuration change that can be allocated to a subject.

6.1.4.3 Static Policy Attribute Initialization (FMT_MSA_EXP.3)

The System Architect defines the allowed information flows in the configuration data. If an information flow is not explicitly defined in the configuration data, then any attempt by a subject to invoke such an information flow will be denied by the TSF enforcing the Partitioned Information Flow SFP. Therefore, by definition the default values for security attributes are the most restrictive possible, since by default no information flow will be possible. All information flows must be explicitly defined in the configuration data, which provides the only means to override the default restrictive values of security attributes.

Even in the case of dynamically assigned resources, such resources must be assigned explicitly and then can only be by dynamically used within the ranges explicitly configured by the System Architect. No resources can be assigned without being fully specified (and no default values are offered), otherwise the assignment will fail. As such, strictly speaking there are no attributes applicable to FMT_MSA_EXP.3.

6.1.4.4 Management of TSF Data (FMT_MCD_EXP.1, FMT_MTD.3)

Once the configuration data has been integrated with the TOE executable using the Green Hills Software Integrate tool, the TSF prevents its modification. There is no capability to modify the configuration data integrated with the TOE executable and loaded onto the target hardware. The configuration information is converted into data structures known as the Boot Tables, which are protected by the kernel. The TSF also restricts the ability to read audit records (including the results of TSF self-tests) from the audit event log to authorized subjects.

The TSF performs a boot time check of the Boot Tables to ensure valid values are accepted for TSF data.

6.1.4.5 Security Management Summary

The Security Management function is designed to satisfy the following security functional requirements:

- FMT_MCD_EXP.1: The TSF prevents unauthorized modification of the configuration data by protecting it within the kernel data structures known as Boot Tables.
- FMT_MOF.1(1)-(6): The TSF restricts the ability to perform authorized functions to subjects that have been appropriately authorized in the configuration data. FMT_MOF.1.1(1) is vacuously satisfied because the TSF does not provide any capability to invoke a configuration change of the TOE (see FPT_CFG_EXP.1).
- FMT_MSA_EXP.1: The TSF assigns authorizations to perform security management functions to subjects as specified in the configuration data. Note that the TSF does not assign an authorization to allow the ability to invoke a TOE configuration change, as this capability is not provided by the TSF.
- FMT_MSA_EXP.3: The TSF enforces restrictive default security attributes that ensure no information flows can occur unless specifically overridden by explicit specification of security attributes in the configuration data.
- FMT_MTD.1(1)-(2): The TSF maintains the TOE configuration data and prevents its modification.
- FMT_MTD.3: The TSF ensures only valid values are accepted for TSF data.
- FMT_SMF.1: The TSF provides security management functions to: restart the TOE; halt the TOE; conduct self-tests; and transition to maintenance mode.

6.1.5 TSF Protection

The TOE implements various mechanisms to protect the integrity of the TSF and to detect conditions that could otherwise compromise its security.

6.1.5.1 Abstract Machine and TSF Testing (FPT_AMT.1, FPT_TST_EXP.1)

The TOE implements a suite of self -tests to verify the correct operation of its supporting hardware and the integrity (using SHA-1) of its own executables and configuration data. The TOE executes the self -tests during system startup, during the process of automated recovery from a temporary hardware failure, and periodically on a preconfigured schedule defined in the configuration data by the System Architect.

Available tests for the supporting hardware or abstract machine and the TOE are categorized in two ways:

- abstract machine tests (or AMT) or self-tests (or TSF) and
- initial (or start-up/power-up) or continuous.

The AMTs address the correct operation of the supporting hardware while the (TSF) self-tests address the integrity of the TSF. Initial tests are run during start-up or during automatic recovery, while continuous tests are available to be invoked from partitions as configured by TOE users.

- The continuous AMT tests include Processor Privileged Instruction Tests, RAM Bit Errors Tests, Unmapped Memory Protection Test, and Mapped Memory Read-Only Protection Test.
- The initial AMT tests include Arithmetic Logic Unit (ALU) Tests, High Resolution Clock Tests, L1 Cache Tests, RAM Memory Reservation Tests, Memory Management Unit (MMU) Tests, RAM Bit Errors Tests, RAM Register Tests, and ROM Tests.
- The initial and continuous (TSF) self-tests perform SHA-1 integrity checks of the TSF executable and configuration files.

The User and Adminsitrator guidance should be consulted for more specific information about these tests, examining their results, and exercising the tests.

6.1.5.2 Establishment of Secure State (FPT_ESS_EXP.1)

The TOE ensures the TSF is established in a secure state as defined by the configuration data. During TOE initialization, the Boot Tables are processed. The Boot Tables define the partitions, subjects, resources, and authorized information flows. During initialization, the data in the Boot Tables is used to create AddressSpaces, the Page Table, memory segments and initial objects assigned to each partition. The kernel also assures that no applications or partitions can execute until after initialization has completed. If the TSF encounters any errors during initialization, the kernel will enter the secure halt state. At the end of initialization, the initial secure state of the TOE is established and the TSF will enforce the Partitioned Information Flow Policy as specified by the configuration data.

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

If the TSF identifies a failure in its initial start-up tests or its periodically executed self tests, or encounters any other failure that requires halting system operation, it enters the secure halt state. This state represents the secure state of the system following a failure condition. Task scheduling is not performed within this state. Within this state, a processor reset is required to exit, which assures all tests and initialization must succeed before re-entering the secure state.

6.1.5.4 TOE Halting, TOE Maintenance, and TOE Restart (FPT_HLT_EXP.1, FPT_MTN_EXP.1, FPT_MTN_EXP.2, FPT_RST_EXP.1)

The TOE provides authorized subjects with the capability to request the following changes in the TOE state:

- Halting the TOE
- Transitioning to Maintenance Mode
- Restarting the TOE.

These capabilities are supported through the Commanded Shutdown IODevice. A subject is authorized to access this IODevice by a Link Object to the IODevice being included for the subject's AddressSpace in the configuration data.

When a valid Commanded Shutdown service call is invoked by the authorized subject, the maintenance state is entered. When within the maintenance state, the secure state is preserved (i.e., scheduling is halted). There are no controlled operations that can be invoked by the user. From within the maintenance state a user handler, if defined, is invoked. The user intent, conveyed by the data passed into the service call, is available to the user handler in order to determine the project specific appropriate action. If the user handler is not defined, or does not perform a project specific action (TOE restart, enter maintenance mode, etc), the TOE is halted.

6.1.5.5 TOE Recovery (FPT_RCV_EXP.2, FPT_RCV.4)

The TSF uses automated procedures to return the TOE to an operational secure state following TOE restarts in response to a power failure. When automated recovery is not possible, the TSF enters secure state.

The TOE regards all failures within the kernel, other than power failure, as catastrophic. In the event of a catastrophic failure, the TOE enters the secure state. Therefore, all security functions of the TOE either complete successfully or, in the event of a catastrophic failure, recover to a consistent and secure state.

6.1.5.6 Reference Monitor (FPT_PLP_EXP.1, FPT_RVM.1, FPT_SEP.3)

The TOE kernel operates in its own domain and protects itself from interference and tampering by untrusted subjects. Untrusted subjects operate in the partitions that are created and maintained by the kernel. Each partition's memory is protected from access by another partition. The TOE uses the underlying hardware Memory Management Unit (MMU) to enforce execute/read/write permissions on memory segments. The kernel halts any task that attempts to violate the memory protection. Access into the kernel from a partition is via a kernel API. The APIs are specific to particular object types defined by the TOE. They provide the only means to manipulate the TOE objects. Therefore, the TSF is able to ensure that all TSP enforcement functions are invoked and succeed before any other function in the TSF scope of control is allowed to proceed.

The software architecture of the TSF is such that the TSF software subsystems responsible for handling the operations on TOE objects (and therefore enforcing the Partition Information Flow Policy) are separated architecturally from the other software subsystems of the TSF. This separation is described in the TSF architectural design and demonstrated in the design documentation. Since the underlying processor used by the TOE (PowerPC) provides only two modes of operation (privileged and user), it is not possible to use hardware mechanisms to separate the part of the TSF that enforces the Partition Information Flow Policy from the rest of the TSF, and also separate the TSF from the other domains within the TSC. Therefore, separation is achieved by architectural means within software, as permitted by the Separation Kernel PP.

The TSF is implemented in accordance with the principle of least privilege. No internal functions of the TSF have more access to TSF data and other TSF resources than is required for their assigned functionality. The TOE makes use of the following mechanisms to achieve least privilege:

- Processor supervisor-mode – the TSF executes in the context of an AddressSpace that has supervisor-mode privileges. Only those components necessary to support kernel operation are included in supervisor-mode
- User application virtualization – each application executes in the context of a virtual AddressSpace, where the application is limited only to the resources provided to it and is constantly monitored by the processor hardware for violations. Within a virtual AddressSpace, an application has read/write access only to a limited subset of the processor's registers (the user-mode registers)
- Static configuration data – configuration data is fixed at build time and cannot be modified during run-time. There are no dynamic means for an application to allocate resources or information flows during execution. Least privilege is enforced by providing to each virtual AddressSpace only the resources and information flows defined in the static configuration data.

The TOE design evidence evaluated against the ADV_INT_EXP.3 requirements specified in Section 5.2 substantiates this behavior.

6.1.5.7 Reliable Time Stamps (FPT_STM.1)

The TSF is able to provide reliable time stamps for its own use, based on the hardware clock provided by the underlying PowerPC processor. The time stamp comprises a value that monotonically increases from the time the TOE starts up. The high resolution clock used for time stamps provides a resolution of better than 10 microseconds.

6.1.5.8 TSF Protection Summary

The TSF Protection function is designed to satisfy the following security functional requirements:

- FPT_AMT.1: The TSF implements self tests that execute during initial start-up, automated recovery and periodically during normal operation.
- FPT_CFG_EXP.1: The TSF does not provide a configuration change capability. In order to change the configuration of the TOE, the TOE must be halted and a new image, based on a different static configuration file, must be downloaded onto the PowerPC card.
- FPT_ESS_EXP.1: The TSF establishes its initial secure state by processing the Boot Tables, which represent the secure state determined by the System Architect in the configuration data. The configuration data defines the Partitioned Information Flow Policy, which is enforced by the TSF.
- FPT_FLS.1: The TSF preserves a secure state in the event of a failure of the initial start up tests or the periodic self tests, or the occurrence of a failure requiring the halting of system operation. In these circumstances, the TSF enters the defined secure state.
- FPT_HLT_EXP.1: An authorized subject can call a kernel function to request a halt of the TOE. The TSF will preserve the secure state when halting the TOE.
- FPT_MTN_EXP.1: An authorized subject can call a kernel function to transition the TOE to maintenance mode. The TSF will preserve the secure state when maintenance mode is entered.
- FPT_MTN_EXP.2: When in maintenance mode, the TSF does not respond to any requests for operations on TOE objects, thereby ensuring that no operations that could violate the TSP can be performed.
- FPT_PLP_EXP.1: The implementation of the TSF enforces the principle of least privilege, using techniques such as processor supervisor-mode, user application virtualization, and static configuration data, as demonstrated by the design evidence evaluated against the ADV_INT_EXP.3 requirements specified in the SKPP.
- FPT_RCV_EXP.2: The TSF uses automated recovery procedures to return the TOE to an operational secure state following failures requiring halting of system operation.
- FPT_RCV.4: All security functions of the TOE either complete successfully, or in the event of power failures, temporary hardware failures or catastrophic failures, recover to a consistent secure state.
- FPT_RST_EXP.1: An authorized subject can request a restart of the TOE by calling a kernel function. The TSF preserves the secure state of the TOE following a restart.
- FPT_RVM.1: Entry to the kernel is via object-specific kernel APIs that are processed by the kernel and that ensure the TSP enforcement functions cannot be bypassed.
- FPT_SEP.3: The TSF uses the underlying MMU to support the creation of partitions that maintain separation between the security domains of subjects in the TSF scope of control, and between untrusted subjects and the TSF. The software design and implementation of the TSF provides separation of the unisolated portion of the TSF from the portion responsible for enforcing the Partition Information Flow Policy.
- FPT_STM.1: The TSF generates its own time stamps, which are monotonically increasing from the time of TOE initialization, with a frequency of better than 100KHz.
- FPT_TST_EXP.1: The TSF implements self tests that execute during initial start-up, automated recovery and periodically during normal operation. In addition, the TSF uses SHA-1 to verify the integrity of TSF

configuration data and stored TSF executable code. The results of the self tests (success or failure) are written to the audit event log, which can be read by authorized subjects.

6.1.6 Resource Utilization

The System Architect specifies in the configuration data the memory resources that are allocated to each partition. Each partition has its own memory quota that cannot be changed by the partition during TOE operation. All dynamic objects required by the application running in a partition are created from the allocated memory quota. The partition cannot grow its memory allocation, nor does it release memory back to the kernel. A partition cannot exhaust or affect the memory availability of another partition.

The System Architect also specifies in the configuration data the processing time to be allocated to each partition, by defining execution time windows for each partition. The TOE implements partition scheduling, in which each partition is allocated blocks of time to execute its applications. When a partition is active, its CPU resource is absolutely guaranteed.

The Resource Utilization function is designed to satisfy the following security functional requirements:

- FRU_RSA.2: As indicated above, the TSF enforces minimum and maximum quotas of both system memory and processing time for each partition by allocating fixed, invariable memory and time quotas for each partition as defined in the configuration data.
- FRU_PRU_EXP.1: Since the memory and processing time quotas allocated to each partition are fixed, the TSF is able to exhibit predictable and worst-case bounded usage of execution time and memory.

6.2 TOE Security Assurance Measures

6.2.1 Configuration Management

The Green Hills Software configuration management (CM) system uses a version control system to provide an automated means of ensuring only authorized changes are made to configuration items, including the TOE implementation representation. Build scripts, maintained under the version control system, are run against the source code to support generation of the TOE. The version control system can also be used in conjunction with scripts to ascertain all changes between the TOE and its preceding version and to conduct impact analysis, whereby it is possible to identify all configuration items affected by the modification of a given configuration item. Green Hills uses a problem tracking software tool for tracking and maintaining security flaws. The CM documentation describes the automated tools and their use within the CM system.

The TOE reference is unique to each version of the TOE and the TOE is labeled with this reference. The CM documentation comprises the configuration list, CM plan, acceptance plan, and integration procedures. The configuration list describes the configuration items comprising the TOE and clearly identifies the configuration items comprising the TSF. The list of configuration items includes: the implementation representation; security; development tools and related information; and the evaluation evidence required by the assurance components in the ST. Configuration items are uniquely identified by the combination of directory path name, file name, date, and version control system version number. This is described in the CM documentation.

The CM plan describes how the CM system is used. As identified above, the CM system ensures only authorized changes can be made to the configuration items and supports generation of the TOE. The CM system provides the capability to audit all modifications to the TOE, including the originator, date and time. The CM system is able to identify the master copy of all material used to generate the TOE. The acceptance plan describes the procedures for accepting modified or new configuration items as part of the TOE. Green Hills Software uses a Configuration Control Management board (CCMB) to oversee the configuration management of the TOE and to accept and approve changes to the TOE. The integration procedures describe how the CM system is applied to the TOE manufacturing process.

The Configuration Management assurance measure satisfies the following security assurance requirements claimed for the TOE:

- ACM_AUT.2
- ACM_CAP.5
- ACM_SCP.3

6.2.2 Delivery and Operation

Green Hills Software provides delivery documentation and procedures to identify the TOE, allow detection of unauthorized modifications of the TOE or attempts to masquerade as the developer, and installation and generation instructions at start-up. Green Hills Software uses a NIST-approved digital signature algorithm and secure hash algorithm to sign the TOE code. Green Hills Software delivers the TOE code and cryptographic keying material via independent channels. Green Hills Software provides the TOE media either via CD or FTP server and then initiates a call with the recipient in order to deliver the applicable key. They provide a signature with the delivery that can be verified using Green Hills Software's public PGP key and then used to verify the delivery prior to decryption. The signature service uses the RSA digital signature algorithm with 2048 bits with an odd exponent.

The delivery procedures describe all applicable procedures, including: use of independent delivery channels for code and cryptographic keying material; use of cryptographic mechanisms to detect modification of the code during initial delivery and subsequent updates; use of cryptographic mechanisms to verify the integrity of TOE code; use of cryptographic mechanisms to verify delivery from the intended source (i.e. Green Hills Software).

Green Hills Software also provides documentation that describes the steps necessary to install the INTEGRITY-178B Separation Kernel in accordance with the evaluated configuration:

- High Assurance Security Products Installation, Generation, and Start-up Document

The Delivery and Operation assurance measure satisfies the following security assurance requirements claimed for the TOE:

- ADO_DEL_EXP.2
- ADO_IGS.1

6.2.3 Development

Green Hills Software provides comprehensive design documentation describing the architecture, functional specification, high-level and low-level design, and implementation of the TOE. The design evidence also includes the design of the TOE configuration and loader tools, a representational correspondence, and a TSP model. In addition, the TOE provides a mechanism for trusted initialization.

The informal TSF architectural design describes the design of the TSF self-protection mechanisms and justifies that the design of the TSF protects itself from bypass and tampering and achieves the principle of least privilege.

Green Hills Software provides a number of tools to support the generation and validation of static configuration data. The static configuration data is initially defined in a human-readable, structured, well-defined ASCII file. The System Architect can use the Integrate tool, which includes a graphical user interface for defining the partitions, subjects and resources to be defined in the configuration data, to create this file Alternatively, the static configuration data can be defined directly in an ASCII file, the structure of which is described in the Integrate User's Guide. The System Architect then uses the AdaMULTI/MULTI integrated development environment (IDE) to convert the ASCII file into a machine-readable form, called the boot table representation. Green Hills Software also provides a utility (called gdump) that can convert the boot table representation back into a human-readable form. In order to ensure the integrity of generated configurations, the Security Architect can include SHA-1 hashes and can configure integrity tests for a given project so that the hashes are checked.

The TOE loader combines the TOE, boot table representation, user applications, and SHA-1 hashes into a single image that is loaded and then checked with its own SHA-1 hash, which is used to verify the transferred image after loading.

The TOE is designed so that its components load in a specific order designed to ensure secure initialization. Those components include the ability to check the integrity of the entire configuration and to halt if any errors are detected.

These functions are invoked and run alone as part of a system reset and are part of the kernel, which remains inaccessible even after the system has started. In this manner, the initialization functions are protected from tampering and access at all times.

The functional specification completely represents the TSF and describes the external TSF interfaces (TSFI) in a formal style, supported by informal explanatory text as appropriate. The presentation of the TSFI in the functional specification: designates each external TSFI as security enforcing or security supporting; describes the purpose and method of use of each external TSFI; identifies and describes all parameters associated with each external TSFI; describes all effects and exceptions associated with each external TSFI; describes all error messages resulting from the effects and exceptions associated with each external TSFI; and indicates the TSFI associated with each indirect error message.

The high-level design is semi-formal, supported by informal explanatory text as appropriate, and describes the structure of the TOE in terms of subsystems. It identifies all subsystems in the TSF and designates them as security-enforcing or security-supporting. It describes the structure of all TSF subsystems, the design of the behavior of all TSF subsystems, and the interactions between TSF subsystems.

The low-level design is semi-formal, supported by informal explanatory text as appropriate, and describes the TSF in terms of modules. It describes each module in terms of its purpose, interfaces, return values from those interfaces, called interfaces to other modules, and global variables. It also identifies and describes data shared by security-enforcing modules. The low-level design includes an algorithmic description of each module sufficiently detailed to represent the TSF implementation.

The software architectural description identifies the TSF modules, as described in the low-level design, and describes the process used for modular decomposition of the TSF. It describes how the entire TSF has been structured to minimize complexity and achieve least privilege. The software architectural description addresses: coding standards and any deviation from coding standards, on a per module basis; inter-module coupling and cohesion; the layering architecture, including the services provided by each layer and the interactions between layers; any mutual dependencies; any unused or redundant code in the TSF; and any non-TSP enforcing modules included in the TSF.

The TOE is implemented primarily in C, with some assembly code in the low-level modules. The additional information supplied with the implementation representation describes the use of compiler pre-processor constructs that select which parts of the implementation representation will be included in the implementation itself. The implementation information describes the format of the external representation of the implementation, maps the implementation representation to the external representation of the implementation, and provides a detailed description of how the external representation of the implementation is loaded and executed. Green Hills Software also has debugging tools and documentation that can be used to investigate the behavior of the TSF.

The representational correspondence provides, for each adjacent level of abstraction of the TOE design, a correspondence between the more abstract and the less abstract TOE representation.

Green Hills Software, in collaboration with Rockwell-Collins, has developed a formal TOE Security Policy (TSP) model and a correspondence between the functional specification and the TSP model. The TSP model provides a formal description of the rules and characteristics of all policies in the TSP that can be formally modeled, and a semi-formal description of the rules and characteristics of all other TSP policies. The TSP model includes rationale that it is consistent and complete with respect to all the policies of the TSP. The correspondence demonstrates with appropriate rigor (formally or semi-formally) that all TSFI in the functional specification are consistent and compete with respect to the TSP model.

The Development assurance measure satisfies the following security assurance requirements claimed for the TOE:

- ADV_ARC_EXP.1
- ADV_CTD_EXP.1
- ADV_FSP_EXP.4
- ADV_HLD_EXP.4
- ADV_IMP_EXP.3

- ADV_INI_EXP.1
- ADV_INT_EXP.3
- ADV_LLD_EXP.2
- ADV_LTD_EXP.1
- ADV_RCR.3
- ADV_SPM.3

6.2.4 Guidance Documents

Green Hills Software provides administrator and user guidance on how to utilize the TOE security functions and warnings to administrators and users about actions that can compromise the security of the TOE. The administrator guidance includes procedures necessary for the correct generation of the TSF configuration data and guidance on how to grant the most restrictive set of authorizations and information flows to subjects needed to perform authorized tasks.

Guidance explicit to the INTEGRITY-178B Separation Kernel is documented in:

- High Assurance Security Products User and Administrator Guidance
- High Assurance Security Products Installation, Generation, and Start-up Document.

Green Hills Software documents that support the use of the INTEGRITY-178B Separation Kernel include:

- Safety Critical Products DO-178B Level A Product Specification
- INTEGRITY Reference Manual
- Integrate User's Guide
- INTEGRITY Development Guide
- INTEGRITY BSP User's Guide
- AdaMULTI: Building Applications for Embedded PowerPC.

The Guidance Documents assurance measure satisfies the following security assurance requirements claimed for the TOE:

- AGD_ADM_EXP.1
- AGD_USR.1

6.2.5 Life Cycle Support

The Green Hills Software development security procedures describe all the physical, procedural, personnel and other security measures it has established to protect the confidentiality and integrity of the TOE design and implementation.

Green Hills Software uses a problem tracking software tool to track reported product issues. A list of registered customers receives flaw notifications when a flaw affects security or safety obtaining a fix from Green Hills Software. There is also a service center to deal with problems, including reporting of flaws. The Green Hills Software software verification processes are used to ensure that corrections do not introduce new problems.

Green Hills Software uses a standard development life cycle model, based on RTCA/DO-178B with multiple objectives. The Software Development Plan document describes how objectives are intended to be satisfied and a matrix is built to document correspondence. The life cycle includes software conformity reviews.

Green Hills Software has identified all of the tools used in developing the TOE. The development tools used with the TOE implementation are well defined. The development tools documentation unambiguously defines the meaning of all statements used in the implementation and all implementation-dependent options.

The Life Cycle Support assurance measure satisfies the following security assurance requirements claimed for the TOE:

- ALC_DVS.2
- ALC_FLR.3
- ALC_LCD.2
- ALC_TAT.3.

6.2.6 Ratings Maintenance

The Green Hills Software Assurance Maintenance (AM) Plan identifies the assurance baseline and references the ST for a description of the TOE and its security functionality. The AM Plan describes the planned release cycle of the Target of Maintenance (TOM), the planned schedule of AM audits, and the conditions for the end of maintenance. The AM Plan defines the role and necessary qualifications of the Security Analyst. The AM Plan describes the procedures for identifying changes to the assurance baseline, the procedures necessary for maintaining the assurance established in the certified TOE, and the controls and mechanisms that ensure the procedures described in the AM Plan are followed.

Ratings maintenance activities are documented in:

- High Assurance Security Products GHS Assurance Maintenance Plan
- Green Hills Software INTEGRITY-178B Separation Kernel Assurance Maintenance Requirements

The Ratings Maintenance assurance measure satisfies the following security assurance requirements claimed for the TOE:

- AMA_AMP_EXP.1.

6.2.7 Platform Assurance

The Green Hills Software Platform Assurance Documentation identifies the commercial commodity hardware components, based on the PowerPC architecture, that provide the platform for the TOE. The Platform Assurance Documentation includes specifications of the platform's interfaces, the acceptance test procedures, the platform security tests, and the vulnerability assessment of the TOE platform.

The Platform Assurance assurance measure satisfies the following security assurance requirements claimed for the TOE:

- APT_PDF_EXP.1
- APT_PSP_EXP.1
- APT_PCT_EXP.1
- APT_PST_EXP.1
- APT_PVA_EXP.1.

6.2.8 Tests

The Green Hills Software test documentation comprises a test plan, test procedure descriptions, expected test results, actual test results, and analyses of test coverage and test depth. The test plan identifies the security functions to be tested and the goal of tests to be performed. The test procedure descriptions identify the tests to be performed and describe the scenarios for testing each security function. The tests are automated and the tester cannot jump to a specific test or otherwise modify the test execution sequence. Therefore, test ordering dependencies are automatically dealt with. The expected test results show the anticipated outputs from a successful execution of the tests, while the actual test results demonstrate that each tested security function behaved as specified.

The test coverage analysis demonstrates the complete correspondence between the tests identified in the test documentation and the TSF as described in the functional specification. It rigorously demonstrates that all external TSFI have been completely tested.

The test depth analysis demonstrates the tests identified in the test documentation are sufficient to demonstrate that the TSF operates in accordance with the high-level and low-level design. Test depth also involves coverage of software structure. Green Hills Software uses G-cover, an object code based tool, to ensure that code decision points and code structures are fully covered when exercising the tests. The test philosophy is to cover all code paths.

The Tests assurance measure satisfies the following security assurance requirements claimed for the TOE:

- ATE_COV.3
- ATE_DPT.2
- ATE_FUN.2
- ATE_IND.3.

6.2.9 Vulnerability Assessment

Green Hills Software has conducted a systematic search for inter-partition covert channels. The covert channel analysis identifies all covert channels found in the search and estimates their capacity. The covert channel analysis describes the procedures used to identify covert channels and the information used to conduct the analysis. It describes all assumptions made during the analysis, the method used for estimating channel capacity (based on worst case scenarios), and the worst case exploitation scenario for each identified channel.

The guidance documentation identified in Section 6.2.4 identifies all possible modes of operation of the TOE, including operation following failure or operational error, their consequences, and their implications for maintaining secure operation. The guidance documentation has been written to be complete, clear, consistent and reasonable. It lists all assumptions about the intended environment in which the TOE will operate and all requirements for external security measures, including external procedural, physical and personnel controls. Green Hills Software has conducted an analysis of all guidance documentation supplied with the TOE, demonstrating that the guidance documentation is complete.

There is no strength of function claim associated with the ST or with any of the security mechanisms implemented in the TOE. Although AVA_SOF.1 is claimed as a security assurance requirement, this has been retained simply for Separation Kernels PP conformance. The Separation Kernels PP also makes no strength of function claim for the security functional requirements claimed for the TOE.

Green Hills Software has conducted a systematic search for vulnerabilities, based on traditional fault-tree analysis methods. All TOE deliverables were analyzed. The analysis documents the disposition of all identified vulnerabilities, showing that none of the identified vulnerabilities can be exploited in the intended environment for the TOE. The analysis justifies that the TOE is resistant to obvious penetration attacks.

The Vulnerability Assessment assurance measure satisfies the following security assurance requirements claimed for the TOE:

- AVA_CCA_EXP.2
- AVA_MSU.3
- AVA_SOF.1
- AVA_VLA_EXP.4.

7. Protection Profile Claims

As documented in this Security Target (ST), Green Hills Software INTEGRITY-178B Separation Kernel complies with U.S. Government Protection Profile for Separation Kernels in Environments requiring High Robustness, Version 1.03, dated 29 June 2007.

The Security Environment, Objectives, and Requirements in this ST have been reproduced from the Separation Kernels PP, as indicated below:

- Except as noted below, all threats, organizational security policies and assumptions have been included and no new threats, organizational security policies or assumptions have been introduced.
- Except as noted below, all of the Separation Kernels PP security objectives have been included and no new objectives have been introduced.
- All operations have been completed on the requirements in compliance with the Separation Kernels PP as indicated using bold and bold-italic text in Section 5.1 and 5.2.
- The Separation Kernels PP makes extensive use of end notes to explain and justify refinements made to security functional requirements drawn from CC Part 2. This ST includes an end note to identify each such requirement, but the end note text simply refers to the Separation Kernels PP end note for the full text.
- References to tables and section headings within the requirement statements have been changed as the tables and sections in the ST do not have the same numbers as in the Separation Kernels PP.
- Table 2 (Auditable Events) has been refined to identify the security functional requirements actually included in the Separation Kernels PP and ST.

The following additional tailoring of specific security requirements has been performed:

- FAU_GEN.1: In accordance with the conventions stated in Table 1.1 of the Separation Kernels PP, the ST author has added the text "**Refinement:**" to the start of FAU_GEN.1.1-NIAP-0407, since the Separation Kernels PP has clearly refined this requirement.
- FPT_CFG_EXP.1: In accordance with the Application Note to FPT_CFG_EXP.1.1, the evaluator has struck out FPT_CFG_EXP.1.2 and FPT_CFG_EXP.1.3, as they are not applicable to the TOE.
- ADO_DEL_EXP.2.5D: Cases (a) (Digital Signature Algorithm) and (c) (Elliptic Curve Digital Signature Algorithm) are rendered non-applicable by the selection of RSA Digital Signature Algorithm and have been removed.
- ADO_DEL_EXP.2.6D is rendered non-applicable by the selection in ADO_DEL_EXP.2.3D of "cryptographic signature" and so has been removed.

The ST specifies an additional security management requirement in FMT_SMF.1, to provide the capability to enable and disable the audit function. An additional iteration of FMT_MOF.1 (i.e., FMT_MOF.1(6)) restricts the ability to enable and disable the audit function to authorized subjects.

8. Rationale

This section provides the rationale for completeness and consistency of the Security Target. The rationale addresses the following areas:

- Security Objectives;
- Security Functional Requirements;
- Security Assurance Requirements;
- Strength of Functions;
- Requirement Dependencies;
- TOE Summary Specification; and,
- PP Claims.

The Separation Kernels PP successfully completed evaluation on July 17, 2007.The TOE described in this Security Target conforms with the evaluated Separation Kernels PP.

8.1 Security Objectives Rationale

The Separation Kernels PP provides rationale for the security objectives demonstrating that security objectives are suitable to cover the intended environment. The rationale (provided in Sections 7.1 through 7.3 of the Separation Kernels PP) is valid for this ST as no new security objectives or environmental claims were added.

8.2 Security Requirements Rationale

The Separation Kernels PP provides rationale for the security requirements, demonstrating that the security requirements are suitable to address the IT security objectives. The rationale for the Separation Kernel PP requirements is included here by reference (see Section 7.4 of the Separation Kernels PP).

This ST specifies an additional security management function in FMT_SMF.1, the ability to enable and disable the audit security function. To support this capability, this ST also includes an iteration of FMT_MOF.1 not specified in the Separation Kernels PP: FMT_MOF.1(6), specifying restrictions on the ability to enable and disable the audit function. The specification of FMT_MOF.1(6) contributes to the satisfaction of the following IT security objectives of the Separation Kernels PP:

- O.AUTHORIZED_SUBJECT: FMT_MOF.1(6) specifies that only authorized subjects have the capability to enable and disable the audit security function, thus contributing to the objective that only authorized subjects are allowed to access restricted services
- O.MANAGE: The additional operation in FMT_SMF.1 specifies the capability to enable and disable the audit function, which is a capability to support management of the TOE security functions. FMT_MOF.1(6) specifies that only authorized subjects can enable and disable the audit function, thus ensuring that unauthorized subjects are restricted from this security management function.

8.3 Explicitly Stated Requirements Rationale

The Separation Kernels PP provides rationale for the explicitly stated security requirements, demonstrating that the explicitly stated security requirements are necessary, because the Common Criteria requirements were found to be insufficient as stated. The rationale (provided in Section 7.6 of the Separation Kernels PP) is valid for this ST as no new explicitly stated security requirements were added.

8.4 Strength of Functions Rationale

The Separation Kernels PP provides rationale for the minimum strength of function claim made for the TOE security functional requirements. The rationale (provided in Section 7.7 of the Separation Kernels PP) is valid for this ST as no new security requirements were added.

8.5 Requirement Dependency Rationale

The Separation Kernels PP requirements have been evaluated and it has been determined that all dependencies have been satisfactorily addressed in this ST. Since this ST does not introduce any new requirements, no additional rationale is necessary.

8.6 TOE Summary Specification Rationale

Each subsection in Section 6, the TOE Summary Specification, describes a security function of the TOE. Each description is followed with rationale that indicates which requirements are satisfied by aspects of the corresponding security function. The set of security functions work together to satisfy all of the security functions and assurance requirements. Furthermore, all of the security functions are necessary in order for the TSF to provide the required security functionality.

This Section in conjunction with Section 6, the TOE Summary Specification, provides evidence that the security functions are suitable to meet the TOE security requirements. The collection of security functions work together to provide all of the security requirements. The security functions described in the TOE summary specification are all necessary for the required security functionality in the TSF. **Table 4 Security Functions vs. Requirements Mapping** demonstrates the relationship between security requirements and security functions.

	Security audit	**User data protection**	**Identification and authentication**	**Security management**	**TSF Protection**	**Resource utilization**
FAU_ARP.1	X					
FAU_GEN.1	X					
FAU_SAR_EXP.1	X					
FAU_SEL_EXP.1	X					
FDP_IFC.2		X				
FDP_IFF.1		X				
FDP_IFF.3		X				
FDP_RIP.2		X				
FIA_ATD_EXP.1(1)-(3)			X			
FIA_USB_EXP.1(1)-(3)			X			
FMT_MCD_EXP.1				X		
FMT_MOF.1(1)-(6)				X		
FMT_MSA_EXP.1				X		
FMT_MSA_EXP.3				X		
FMT_MTD.1(1)-(2)				X		
FMT_MTD.3				X		

FMT_SMF.1				X		
FPT_AMT.1					X	
FPT_CFG_EXP.1					X	
FPT_ESS_EXP.1					X	
FPT_FLS.1					X	
FPT_HLT_EXP.1					X	
FPT_MTN_EXP.1					X	
FPT_MTN_EXP.2					X	
FPT_PLP_EXP.1					X	
FPT_RCV_EXP.2					X	
FPT_RCV.4					X	
FPT_RST_EXP.1					X	
FPT_RVM.1					X	
FPT_SEP.3					X	
FPT_STM.1					X	
FPT_TST_EXP.1					X	
FRU_RSA.2						X
FRU_PRU_EXP.1						X

Table 4 Security Functions vs. Requirements Mapping

8.7 PP Claims Rationale

See Section 7, Protection Profile Claims.

[1] See End Note 1 of Section 5 of the Separation Kernels PP.
[2] See End Note 2 of Section 5 of the Separation Kernels PP.
[3] See End Note 3 of Section 5 of the Separation Kernels PP.
[4] See End Note 4 of Section 5 of the Separation Kernels PP.
[5] See End Note 5 of Section 5 of the Separation Kernels PP.
[6] See End Note 6 of Section 5 of the Separation Kernels PP.
[7] See End Note 7 of Section 5 of the Separation Kernels PP.
[8] See End Note 8 of Section 5 of the Separation Kernels PP.
[9] See End Note 9 of Section 5 of the Separation Kernels PP.
[10] See End Note 10 of Section 5 of the Separation Kernels PP.
[11] See End Note 11 of Section 5 of the Separation Kernels PP.
[12] See End Note 12 of Section 5 of the Separation Kernels PP.
[13] See End Note 13 of Section 5 of the Separation Kernels PP.
[14] See End Note 14 of Section 5 of the Separation Kernels PP.
[15] See End Note 15 of Section 5 of the Separation Kernels PP.
[16] See End Note 16 of Section 5 of the Separation Kernels PP.
[17] See End Note 17 of Section 5 of the Separation Kernels PP.
[18] See End Note 18 of Section 5 of the Separation Kernels PP.
[19] See End Note 1 of Section 6 of the Separation Kernels PP.
[20] See End Note 2 of Section 6 of the Separation Kernels PP.
[21] See End Note 3 of Section 6 of the Separation Kernels PP.