Bundle Protocol Security (BPSec) Library (BSL) Software Requirements Document (SRD)

Prepared By

Johns Hopkins University Applied Physics Laboratory (JHU/APL)

CHANGE LOG

Revision	Submission Date	Affected Sections or Pages	Change Summary
Initial	1 April 2024	All	Initial document release

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

1.1 IDENTIFICATION

Table 1: Software Identification

Property	Value
Configuration ID (CI)	681.4
Element	Mission Control System (MCS)
Program Set	Bundle Protocol Security Library (BSL)
Version	1.0

1.2 PURPOSE

The purpose of this Software Requirements Document (SRD) is to outline the set of features and interfaces necessary for the proper operation of the Bundle Security Protocol (BPSec) Library (BSL). To that end, this specification provides normative guidance on how the resulting BSL is to be developed and establishes the system-level acceptance criteria for BSL software.

This specification is considered a "living document" and will be refined over the course of the project to both adjust to current events and to refine requirements in the context of incremental development.

This specification is written with multiple stakeholders in mind as listed here:

- **Software architects and engineers**, who use this specification to inform the design, implementation, and unit tests of the BSL software in the context of development spirals
- **Software testers**, who use this specification to develop acceptance tests used to demonstrate the expected functioning of BSL.
- **Project management**, who uses this specification to ensure that the BSL features will provide the expected value to the AMMOS program and its expected customer set.
- **Adopting missions**, particularly those developing software interfacing with the BSL, who use this specification to determine the mapping of BSL features to their own software capabilities and needs.
- **Network architects and operators**, who review this specification to ensure that the BSL can support needed operational use cases within the physical resources and capabilities of the networks in which they may be deployed.

This specification provides the context for the interpretation of requirements, a description of the capabilities of the system, and a set of traceable requirements statements. The contributions of these informational elements to the overall purpose of this document are as follows.

- **System goals and objectives** describe the capabilities and elements of the BSL. This information also includes system-wide design constraints. With the BSL CONOPs document, this information provides the consolidated view of external stakeholder expectations.
- **User stories** describe the higher-layer functioning of the BSL system in the context of system goals and objectives.
- **Requirements** capture the key properties of BSL with appropriate levels of specificity and measurability.

The remainder of this document is organized as follows.

- The remainder of **Section 1** identifies requirement levels and any custom terminology or notation.

- **Section 2** provides an overview of BSL and a decomposition of elements as well as user stories.
- **Section 3** provides the assumptions and constraints of the BSL system.
- **Section 4** provides the set of library requirements for each subsystem identified by the logical decomposition.
- **Sections 5 - 9** identify other non-functional requirements of the BSL as a whole, including performance, reliability, safety, adaptability, and security requirements.

1.3 REQUIREMENTS LEVELS AND SCOPE

BSL requirements exist within a larger requirements ecosystem provided by the AMMOS. Specifically, the requirements identified in this specification exist as Level 5 requirements that trace up to one Level 4 requirement in the Application Security (ASEC) subsystem, ASEC-86.

Lower-level requirements (e.g., Level 6) are provided, where appropriate and as necessary, as part of the design of the BSL and are otherwise outside of the scope of this document.

1.4 TERMINOLOGY AND NOTATION

1.4.1 TERMINOLOGY

The following words have specific meaning in the requirements:

- Shall—required testable functionality.
- Must—used in constraints to denote BSL functionality that is needed but is not directly testable.
- Will—used in assumptions to denote a capability that is external to the BSL but needed for the BSL to function.

Table 2: BSL Acronyms

Term	Definition
AAD	Additional Authenticated Data
ADU	Application Data Unit
AEAD	Authenticated Encryption with Associated Data
BCB	Block Confidentiality Block
BIB	Block Integrity Block
BP	Bundle Protocol
BPA	Bundle Protocol Agent
BPSec	Bundle Protocol Security
BPv7	Bundle Protocol version 7
CLA	Convergence Layer Adapter
EID	Endpoint Identifier
DTN	Delay-Tolerant Networking
LTP	Licklider Transmission Protocol

Term	Definition
PDU	Protocol Data Unit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

1.4.2 Requirements Notation

Requirements are identified throughout this document in accordance with the following naming convention:

BSL-<mnemonic>-<Grouping>-<Group Enumerator>

Where:

- **<Mnemonic>** refers to the type of requirement (e.g., “GEN” for general functional requirements, “SEC” for security service requirements, etc.).
- **<Grouping>** represents clusters of related requirements within the requirements type. The concept of grouping helps to add context where multiple requirements may be considered as working towards describing a common purpose of capability and not otherwise independent.
- **<Group Enumerator>** represents the enumeration of requirements as part of that grouping, starting at 0. Unless otherwise specified there is no expected meaning implied in the ordering of requirements in a grouping.

Table 3: Mnemonics

Term	Definition
ASM	Assumptions
CFG	Configuration Requirements
CON	Constraints
ERR	Error and Safety Requirements
GEN	General Functional Requirements
PFR	Performance and Reliability Requirements
SSF	Security Service Functional Requirements
SVC	Service Requirements
?IN	Interface Requirements where ? is one of BPA, Crypto, Policy, Telemetry, or Logging

1.5 References

Table 4: Applicable MGSS Documents

Title	Doc. Number	Revision
AMMOS Technical Standards Profile	DOC-001101	A
MGSS Implementation and Maintenance Task Requirements (External)	DOC-001819	B
Application Security (ASEC) Functional Requirements Document (FRD)	DOC-001253	D*

Title	Doc. Number	Revision
Bundle Protocol Security Library (BSL) Concept of Operations	DOC-005727	-

*** Note: Rev. D is the latest release of the ASEC FRD, which must be revised to add Level 4 requirements associated with the Level 5 requirements in this document.**

Table 5: Applicable External Documents

Title	Reference
Bundle Protocol Version 7	https://www.rfc-editor.org/rfc/rfc9171.html
Bundle Protocol Security	https://www.rfc-editor.org/rfc/rfc9172.html
Default Security Contexts for Bundle Protocol Security (BPSec)	https://www.rfc-editor.org/rfc/rfc9173.html
Security Requirements for Cryptographic Modules	https://csrc.nist.gov/pubs/fips/140-3/final
PKCS #11 Cryptographic Token Interface Base Specification	https://docs.oasis-open.org/pkcs11/pkcs11-base/v3.0/pkcs11-base-v3.0.pdf

2 OVERALL DESCRIPTION

This section provides an overview of BSL and the BSL context and interfaces.

2.1 OVERVIEW

The BSL Concept of Operations provides the following overview of BSL:

The Bundle Protocol (BPv7) was developed as part of an overall Delay-Tolerant Networking (DTN) architecture for data exchange in challenging communications environments. BPv7 has been baselined for use in a variety of NASA and other space agency missions such as NASA's LunaNet and ESA's Moonlight[1] *projects, and recommended for a variety of other IOAG and CCSDS space-networked architectures and functions.*

The BPv7 protocol data unit is the Bundle and bundles are comprised of multiple Blocks[2] *of information. An application that produces, processes, and/or delivers bundles in compliance with the BPv7 specification is called a Bundle Protocol Agent (BPA).*

The Bundle Protocol Security (BPSec) protocol is the standardized mechanism for BPv7 bundle-layer security[3]*. BPSec defines special extension blocks that carry cryptographic information related to other blocks in the same bundle. All standards-compliant BPAs must be able to process BPSec blocks in a received bundle if required by the security policy of the BPA.*

[1] https://www.esa.int/Applications/Connectivity_and_Secure_Communications/Moonlight

[2] A *Primary Block*, a *Payload Block*, and zero or more *Extension Blocks*.

[3] Separate from application-layer or link-layer security, which should also be present in a secure system.

The processing of BPSec extension blocks is both standard and deterministic. Any BPSec security block processor applies cryptographic functions from one or more cipher suites to the contents of one or more bundle blocks. The BSL implements a general-purpose BPSec security block processor.

2.2 BSL SYSTEM CONTEXT

A system context for the BSL is provide in in Figure 1. This figure identifies common logical elements that are expected to exist on any hosting platform, the recommended location of the BSL within a hosted system, and the interfaces that describe how the BSL exchanges information and processing with host capabilities.

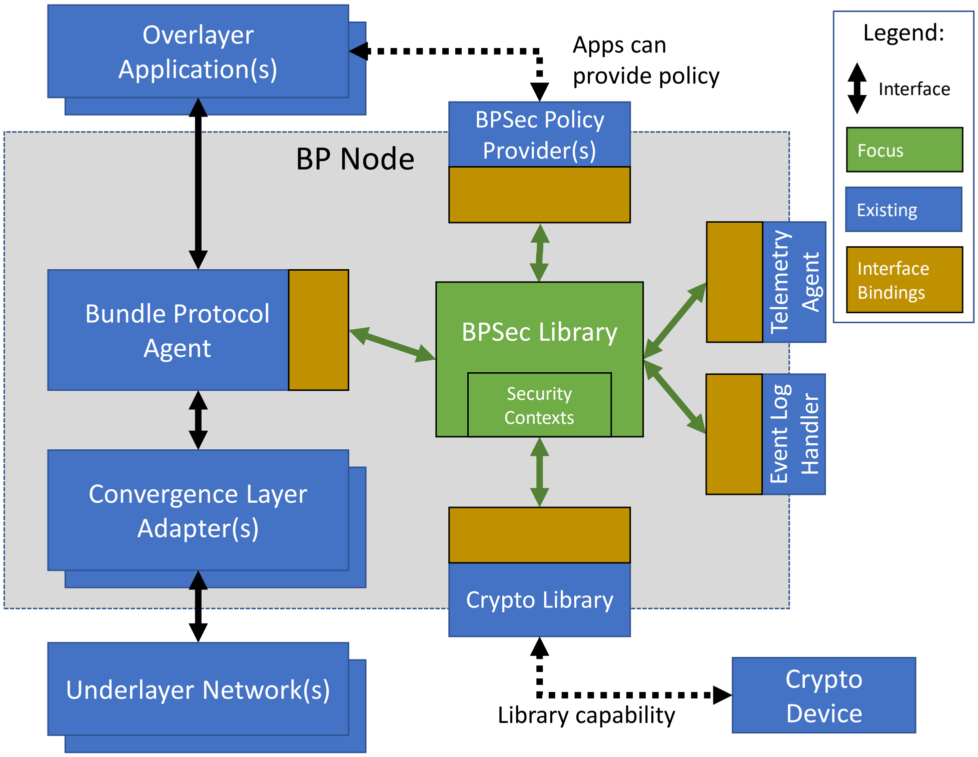

Figure 1: BSL System Context

Similar to the identification of logical interfaces, the definition of logical elements in this system context is meant to explain data and control flows with a representative (not mandated) host implementation. The elements described in this section are not intended to levy any requirements on the construction of a host system beyond the requirements on host interfaces and bindings.

The BSL expects that the hosting system provides nine different logical functions. These functions can be described based on how they do (or do not) interact with the BSL.

2.2.1 INDIRECT HOST FUNCTIONS

Indirect functions are those that do not directly interface with the BSL, but whose existence and implementation provide data, configuration, or other information that is needed or processed by the BSL. These functions include overlayer applications, underlayer networks, and convergence layer adapters.

- **Overlayer Applications (OAs)**: These applications do not directly interact with the BSL, but their existence is presumed as the ultimate source and destination of application data units (ADUs). Additionally, these applications might also be the source of some or all of the security policy on the host – particularly to the extent that OAs need to ensure that their specific traffic is secured in desired ways.

NOTE: OAs are not required for the proper function of a host or the BSL. For example, networks might deploy Bundle Protocol Nodes that exist only as forwarding engines and host no overlayer applications and, thus, perform no ADU delivery.

- **Convergence Layer Adaptors (CLAs):** CLAs performs all processing necessary to deliver bundles, as atomic units to and from the BPA, across a specific underlayer network transport. Different convergence layer types and instances will have different security properties, for which BPSec policy can be tailored to ensure specific application-defined levels of security.
- **Underlayer Networks (UNs)**: The UNs represent the physical networks, through which CLA PDUs are transported between BP nodes. They could represent true internetworks (such as IP networks) or simpler point-to-point links or datagram tunnels. UNs will have different security properties, which may cause their associated CLAs to define or tailor different BPSec policies.
- **Crypto Device (CD)**: Certain host implementations may require that cryptographic processing occur as part of an external hardware device or other trusted subsystem independent of the host BP Node. The existence (or absence) of the CD is expected to be hidden from the BSL. Therefore, the BSL should not require any additional implementation, interface, or configuration to be used with these devices for the implementation of trusted execution and storage.

2.2.2 Direct Host Functions

Direct host functions are those that directly interface with the BSL. These functions include the Bundle Protocol Agent, Policy Providers, Crypto Libraries, and Telemetry and Logging mechanisms.

- **Bundle Protocol Agent (BPA)**: The BPA performs the necessary bundle handling functions defined in RFC 9171. This includes sourcing from and delivering to applications as well as routing along a path to a destination. The PDUs exchanged between BPAs of different nodes are encoded bundles. There is no standardized BPA interface for generic bundle or block handling, so the interface between BPA and extension block handlers (of which the BSL is one) is defined on the block handler side and used by the BPA.
- **BPSec Policy Providers (BPPs):** BPPs maintain logic about what security operations are required for specific outgoing, transiting, or incoming traffic. This includes information about when to source, verify, or accept security operations. These policy providers collect relevant configurations from other host components and contain the information necessary to inform all BPSec-related processing.
- **Crypto Library (CL)**: CLs implement cryptographic primitives such as random number generation, message authentication code (MAC) and key derivation function (KDF) processing. Similarly, they perform cryptographic processing.

  NOTE: These libraries might exist solely within the user-space of the BP Node, as external software subsystems, or as hardware devices. The implementation of crypto libraries is a function of the security requirements and capabilities of the host itself. For example, crypto libraries might communicate with a hardware device via PKCS #11 or similar interface.

  NOTE: Poorly implemented crypto libraries may introduce security vulnerabilities not associated with BPSec processing or the BSL.
- **Telemetry Agent (TA)**: The telemetry agent collects various state information related to the proper operation of the BP Node (of which the BSL is a part). Telemetry includes information such as the state of the system as well as any feedback for errors or successful operation needed by the host for its own health and status reporting.
- **Event Logging (EL)**: Separate from telemetry agents, many hosts maintain logging mechanisms for the generation of persisted events, warning, errors, alarms, and other important notifications. The EL function is used by the BSL to log similar events that are, otherwise, not part of normally produced health and status telemetry.

2.2.3 BSL Functions

BSL functions are the set of elements included in the BSL itself.

- **BPSec Library**: The principal artifact of this project is a BPA-independent software library implementing the functions defined in RFC 9172. This library has external software interfaces (using in-process APIs) defined for each aspect of: BPA interface, policy interface, and security function interface, and telemetry interface. This library has internal software interfaces for security context implementations and will be provided with implementations of "default contexts" of RFC 9173.

2.3 BSL External Interface Bindings

Since the BSL may be hosted in a variety of environments, any functionality that is not directly related to general BPSec processing is delegated to host functions. In this way, the BSL may provide BPSec logic without requiring that specific hosts incur any greater processing overhead or inefficiencies than necessary.

This delegation is accomplished in the BSL through the definition of various external interfaces between the generic BSL functional processing and host-provided functionality. These interfaces are *external* in the sense that they are external to the BSL library itself. These interfaces are *logical* in the sense that different interfaces are defined for different types of functionalities, but the host may choose to implement these interfaces in any way they see fit. The definition of these interfaces does not imply a *physical* design on the host beyond simple adherence to the interface.

2.3.1 Service Interface

The service interface of the BSL is used by the BPA at specific points in its bundle processing flows. To the extent that all BPAs conform to processing rules and procedures imposed by RFC9171, all BPAs perform approximately the same types of actions in approximately the same order. To that end, an exemplar information flow of bundles through a generic BPA is illustrated in Figure 2. In that figure, the green boxes indicate interactions of the BPA with either overlayer applications or underlayer CLAs, and the annotation "[*]" indicates a multiplicity of many while no annotation indicates a multiplicity of one.

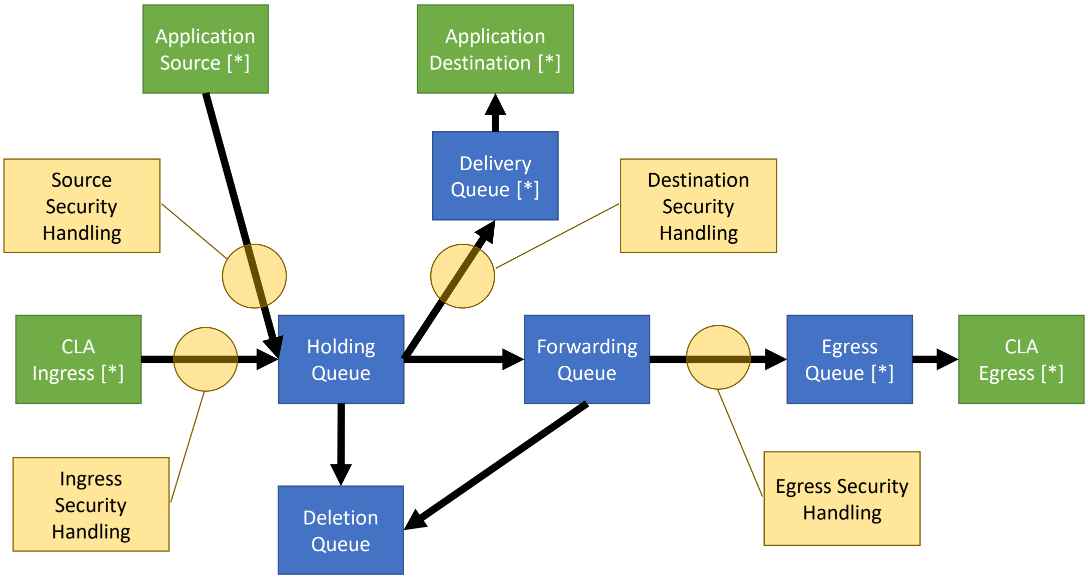

Figure 2: Assumed BPA Information Flows and BSL Interaction Points

This sample flow implies certain interaction points where a BPA may request security services as provided by the BSL.

- **Ingress Security Handling**: This occurs during bundle ingress, after receiving a well-formed bundle from a CLA and before further processing by the BPA (to choose what to do with the bundle: deliver it, forward it, or delete it). This includes security operations with the current node as the security acceptor or verifier. Some of the policy matching can be related to the labeling provided by the CLA used for ingress.
- **Source Security Handling**: This occurs after a bundle is created from source endpoint application data (which includes ADU data as well as any application-supplied metadata to inform extensions and internal labeling). This includes security operations with the current node as the security source as well as the bundle source.
- **Destination Security Handling**: This occurs before a bundle is delivered to an destination endpoint application as a combination of ADU data and metadata. This includes security operations with the current node as the security acceptor as well as the bundle destination.
- **Egress Security Handling**: This occurs during egress, after determining that the bundle is to be forwarded using a specific CLA. This includes security operations with the current node as the security source. Some of the policy matching can be related to the labeling provided to the CLA used for egress.

2.3.2 BPA Binding

Different BPAs may implement different internal representations of bundles, blocks, and associated information. The BSL does not presupposed a specific BPA design and does not levy requirements on how the host BPA chooses to implement RFC9171. Therefore, the BSL requests that the BPA perform block processing on its behalf where such processing relies on the internal implementation of a specific BPA on a specific host.

This type of block processing includes receiving bundle characteristics, as requested, from the BPA without requiring how the BPA stores bundle information. The BSL might, for example, request bundle source and destination information for a bundle, or request the block-type-specific-data field of a particular bundle block, or request listings of blocks present in a bundle.

Block processing also includes requests by the BSL to insert, modify, or remove blocks from a bundle as part of adding, removing, or reacting to security processing. For example, the BSL will provide new security block contents to the BPA when a security operation is required by policy to be added to the bundle. These contents will include security results, security context flags and parameters, and other data necessary to populate the Abstract Security Block (ASB) structure as defined in RFC 9172 Section 3.6. Similarly, when applying a BCB-confidentiality security service to a bundle, the BSL will provide block-type-specific data to the BPA. This block-type-specific data, either plain-text or cipher-text depending on if the security target is being encrypted or decrypted, is expected to be accepted by the BPA to replace the existing block-type-specific data of the associated security target block.

2.3.3 Policy Binding

Many of the functions performed by the BSL in the application of BPSec security rely on one or more policy statements that determine configurations, behaviors, and expectations with varying levels of fidelity. Policy statements are therefore fundamental to, and frequently queried by, the BSL.

Examples of policy information defined by BPSec and needed by the BSL include:

- Identification of security operations required to be represented in a bundle and the role(s) the BPA must assume when processing these security operations.
- Determination of the security operation(s) to be added, verified, or otherwise processed for a bundle.

- Information needed to generate security results, such as security context identifiers and associated security context parameters[4].
- Identification of optional processing action(s) the BSL is expected to execute in the event of a security processing failure. For example, security policy may be used to request the generation of a bundle status report if the verification of a security operation fails.

The variety of policy information that is queried by the BSL may be implemented in many different ways on various types of host systems. For example, hosts with abundant computation resources may implement policy lookups with databases, whereas resource-constrained hosts might implement policy as a series of key-value pairs, wildcard matching, or simple hard-coded rulesets.

For this reason, the policy binding does not imply any implementation of policy information on the host system.

2.3.4 Crypto Binding

The BSL is used to prepare bundle information and other inputs to cryptographic processors, and to accept cryptographic materials from these processors for inclusion in bundle blocks. The BSL does not implement any cryptographic primitives and does not perform any cryptographic processing. All cryptographic processing performed by the BSL occurs through the implementation of cryptographic libraries provided by the host.

The BSL crypto bindings make no presumptions on the cryptographic libraries provided by the host, or whether a single cryptographic library or multiple such libraries are used.

2.3.5 Telemetry and Event Logging Bindings

As a part of the host system, the BSL will make use of host-provided capabilities for reporting its health, status, and statistics telemetry as well as persisting event logs. The BSL does not otherwise implement its own telemetry service or logging as doing so would make unnecessary assumptions on the nature of those interfaces as they are implemented on a host.

In particular, the BSL makes no assumptions that the hosts Telemetry Agent or Event Logger does anything with the information provided to it. The collection, analysis, reporting, and other handling of such information, after it has been reported by the BSL, is solely a function of the design and implementation of the host.

2.4 BSL Internal Interface Bindings

Separate from the bindings between the BSL and elements of the BP Node and other host elements, there are certain internal interfaces within the BSL used to increase the overall extensibility of the library.

2.4.1 Security Context Interface

The application of security services occurs within the context of a security context – which includes the set of cipher suites, parameters, data aggregation, and other elements needed to map bundle information, configuration, and cryptographic processing. RFC9172 presumes the creation of multiple security contexts to specify various ways in which bundles can be secured. This includes default security contexts defined by RFC9173, and emerging security contexts such as being defined by the IETF and CCSDS.

As new security contexts may be implemented over time, the BSL needs to be extended to support these contexts with minimum difficulty. This extensibility is achieved by defining internal security context application programming interfaces to ease the burden of integrating new security context implementations.

[4] RFC 9173 provides examples of the security context parameters supported by the Default Security Contexts for BPSec in sections 3.3 and 4.3.

2.4.2 Security Block Interface

The two security blocks defined by RFC9172 are the Block Confidentiality Block and the Block Integrity Block. However, RFC9172 also allows for the definition of other security blocks (OSBs). While the BSL only implements BCB and BIB blocks (as no OSBs have been standardized), the BSL will define a standard security block processing interface. Such as interface would allow for the possible extensibility of the BSL to include OSB processing if such blocks were to be standardized over time.

2.5 BSL User Stories

User stories defining the expected operation of security services for a BPA, and thus the security processing expectations of the BSL, are documented in the AMMOS BPSec Security Library Use Case Document.

3 BSL Constraints and Assumptions

The following assumptions and constraints are derived both from the above descriptions and BSL system context and are delineated for traceability.

NOTE: These assumptions and constraints are not intended to be testable and are requirements. They serve as context for the implementation and development of the BSL.

3.1 Constraints

The following constraints reflect that the BSL is designed to operate in a spaceflight flight software environment.

Table 6: BSL Constraints

Constraint ID	Constraint	Rationale
BSL-CON-1-0	The BSL must be written in C and compliant with ISO C99.	Compliance with the ISO C99 version of the C language is considered to be the most widely supporting language variant, especially for embedded systems. Restricting to this language variant increases the likelihood of portability and support to the largest number of embedded platforms.
BSL-CON-2-0	The BSL must use an OS Abstraction Layer Library.	The BSL may be used on a variety of host platforms, with different instruction set architectures and operating systems. Using an OS abstraction layer allows the BSL code to have greater portability to a larger number of platforms than hard-coding the library to a specific operating system.
BSL-CON-3-0	The BSL must be independent of the underlying hardware platform.	These include attributes such as address size, register length, endianness.
BSL-CON-4-0	The BSL must allow for parallel processing of bundles and security operations within bundles.	While there may be multiple ways to satisfy performance requirements through the use of powerful hardware, the BSL must also operate on resource-constrained systems that may otherwise be unable to operate quickly in a serialized manner. Therefore, the design of the BSL must accommodate parallel processing techniques.
BSL-CON-5-0	The BSL APIs and documentation must be compatible with terminology and data structures used in Bundle Protocol Version 7 and related specifications [RFC 9171, RFC 9172, RFC 9173].	The layout of data structures in various specifications are intentional to manage the complexity and multiple use cases of various deployments. The BSL design will use these as-is without optimization, renaming, or other modifications that may place restrictions on hosts seeking to use the BSL.
BSL-CON-6-0	The BSL software must conform to the JHU/APL Quality Management System	The document provides best practices and patterns/behaviors to avoid when implementing embedded software and conformance to this is required by any flight software produced by APL.

	process SD-QP-650 as tailored for BSL development.[5]	
BSL-CON-7-0	The BSL behaviors must not conflict with the requirements of NIST FIPS 140-3.	Nothing the BSL does should cause a system to lose compliance.
BSL-CON-8-0	The BSL must not rely on implementation-specific data structures or logical constructs for its BPA interface API.	The BPA interface uses C99 primitive types or data structures defined by the BSL itself. This avoids coupling the interface to a specific BPA implementation.
BSL-CON-9-0	The BSL must not rely on implementation-specific data structures or logical constructs for its Telemetry interface API.	The BPA interface uses C99 primitive types or data structures defined by the BSL itself. This avoids coupling the interface to a specific telemetry sink.

[5] SD-QP-650 is entitled *Space Exploration Sector (SES) Quality Procedure: Software Development Process* and it defines the process by which flight software is developed at JHU/APL.

3.2 ASSUMPTIONS

A BPA is required in order for the BSL to operate. The following detail the assumptions that will be provided by the BPA. The BSL compiles and runs as a library service for a BPA. For the purposes of development, verification, and testing BSL will compile and run on a desktop environment such as Red Hat Enterprise Linux. Ultimately, the system will operate in both ground and flight software environments using an operating abstraction layer (OSAL).

Table 7: BSL Assumptions

Assumptions ID	Assumption	Rationale
BSL-ASM-1-0	The local BPA and BSL will not have duplicate or redundant processing.	The integration of the BSL into the host system will be the only mechanism for implementing bundle security on the host, and the BPA will not ask for the same security services to be applied multiple times to the same bundle.
BSL-ASM-2-0	The external configuration to BSL will be complete and consistent.	The BSL relies on multiple host interfaces for processing and information gathering. These various host-specific implementations are assumed to be coherent with regard to each other. For example, if a policy provider provides the name of a key to use, that the crypto library is able to parse and find keys with that name.
BSL-ASM-3-0	The BSL will only need to process RFC9172 security blocks (BCB and BIB).	While RFC9172 presumes the potential of OSBs, none have been defined and, therefore, only BCB and BIB security blocks need to be processed by the BSL.
BSL-ASM-4-0	Crypto libraries and devices will implement appropriate security for the host device.	The BSL delegates security processing to crypto libraries and devices on the host system. If these libraries and device themselves have vulnerabilities, then the security processing of the BSL will be compromised.
BSL-ASM-5-0	All health, status, and logging functions will occur through the host.	Host systems must produce their own health, status, and statistics telemetry as well as logging various types of events. The BSL should not re-implement these features and can, instead, provide information to existing host mechanisms for these purposes.
BSL-ASM-6-0	The host system will have sufficient resources for security processing within the configured thresholds of the library.	The BSL may be incorporated into hosts of various compute, memory, storage, and other resources. It is incumbent on the host adopting the BSL to properly configure the limitations and thresholds of the library within the capabilities of the host.
BSL-ASM-7-0	The BPA will perform security processing at distinct points along a bundle’s flow within a node.	The BPA is free to use various forms of internal and external (long-term) storage for bundles, especially in its holding queue of bundles awaiting forwarding or delivery (possibly because of time-varying routes or applications respectively). Neither RFC 9171 nor RFC 9172 address a need for internal-to-a-node security, so this assumption addresses the desire to apply security to

		bundles with a local endpoint application which might be held in the node for a significant period of time.
BSL-ASM-8-0	The BSL will operate over an OSAL that will have constrained memory handling, such as memory pool implementations, that can be configured with the number and size of memory pools and blocks within those pools.	Embedded FSW is memory constrained and well-defined memory usage. As such, coding in that environment requires pre-allocated memory pools to avoid memory leaks and other issues.
BSL-ASM-9-0	The BSL will operate over an OSAL that will have threading libraries that allow for the construction of multiple lightweight processes instantiated within the BSL itself.	BSL will be multi-threaded and therefore requires this feature from the OSAL.
BSL-ASM-10-0	The BSL will operate over an OSAL that will have mutual exclusion mechanisms for protecting access to shared memory resources.	Isolation of memory resources enables deterministic memory usage.
BSL-ASM-11-0	The BSL will operate over an OSAL that will have support for common operational operating systems, to include Red Hat Enterprise Linux (RHEL), VxWorks, and RTEMS.	Development of the BSL will be done in RHEL to meet the AMMOS baseline OS and in a flight-like environment using the UT699 platform and RTEMS. VxWorks is included here as it also a predominate RTOS for FSW, however, the BSL will not be verified on that platform.
BSL-ASM-12-0	The BSL will operate over an OS that will have POSIX support.	This leaves open the option to code to POSIX in lieu of an OSAL.

4 BSL REQUIREMENTS

The primary driver of implementation of BSL is RFC 9172. As such, much of the functional requirements for BSL are derived from that document and not replicated here. Requirements that are left to the implementer of the RFC are the primary focus of this document.

This section is segmented as follows:

- **Functional Requirements**: This section details requirements internal to the BSL that relate to manipulation and processing of blocks and bundles.

- **Interface Requirements**: This section primarily describes how a BPA will interface with the BSL.
- **Policy Requirements**: This section describes how the BSL uses and processes blocks and bundles based on policy.
- **Error and Safety Requirements**: This section focuses on requirements for error handling and other off-nominal conditions.
- **Configuration Management Requirements**: This sections focuses on configuration and operations in a given BP node environment.
- **Operating Environment Requirements**: This section details requirements imposed on the BSL based on the expected operating environment.

4.1 BSL Functional Requirements

4.1.1 BSL General Functional Requirements

The general functional requirements of the BSL are identified by the mnemonic “GEN”. These requirements outline the basic processing functions of the BSL as they relate to compliance with the BPSec specification and the exchange of information therein.

Table 8: BSL Security Service Utilities Functional Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-GEN-1-0	RFC 9172 Compliance	The BSL shall be compliant with RFC 9172.	RFC 9172 defines the purposes and function of an implementation of BPSec to include normative statements of required structures and behaviors that do not need to be recreated in this document.	Test
BSL-GEN-1-1	Deterministic Processing Order for Security Blocks	The BSL shall impose a deterministic processing order for all security blocks.	RFC9172 requires security block processing order in certain circumstances (Section 5.1). The BSL will implement a deterministic processing order in all cases as part of the BSL design.	Test
BSL-GEN-2-0	Addition of a Security Block	The BSL shall construct security blocks for inclusion in a bundle.	To add a security operation to a bundle, the BSL may construct a new security block. Alternatively, some security operations can be added to existing security blocks in the bundle (see BSL-GEN-2-2). The actual addition of a security block to a bundle is performed by the BPA.	Test
BSL-GEN-2-1	Addition of Security Operations	The BSL shall add security operations to a security block.	To fulfill the Security Source role, the BSL adds security operations to a bundle.	Test
BSL-GEN-2-2	Determination of Addition of Security Block	The BSL shall determine whether a new security block can be added to the bundle when adding a security operation to a bundle.	The BPA defers to the BSL when security needs to be added.	Test
BSL-GEN-2-3	Validate SOP Uniqueness	The BSL shall ensure that security operations in a bundle are unique.	The BSL will maintain compliance with RFC 9172. This requirement captures that it is the responsibility of the BSL to check/enforce this uniqueness constraint.	Test
BSL-GEN-3-0	Removal of Security Operations	The BSL shall remove security operations from a bundle.	To fulfill the Security Acceptor role (and to handle some security processing failure cases), the BSL removes security operations from a bundle.	Test

BSL-GEN-3-1	Determination of Removal of a Security Block	The BSL shall determine when a security block should be removed from a bundle.	The BSL may request the removal of a security block from the bundle for several reasons, including: a) All of the security operations associated with that security block have been removed; b) Removal of the security block is required by policy; or c) An error has occurred during security processing. The actual removal of information from a bundle is performed by the BPA.	Test
BSL-GEN-3-2	Discarding a Security Block Upon Security Operation Removal	The BSL shall inform the BPA to discard a security block when all security operations for that block have been removed.	The BPA will act upon information regarding successful processing.	Test
BSL-GEN-4-0	Read Block Contents	The BSL shall read non-security block contents as provided by the BPA.	The BSL needs block data, such as block type and the BTSD of security target blocks, in order to apply security services to the blocks in a bundle.	Test
BSL-GEN-5-0	Update Block Contents	The BSL shall provide updated block contents to the BPA.	The BSL may need to change the contents of a security target block. For instance, the BSL must replace the security target block’s BTSD with ciphertext when it is the target of a BCB-confidentiality security operation.	Test
BSL-GEN-6-0	Encode BTSD	The BSL shall encode the BTSD produced for a security block in compliance with RFC9172 encodings.	Security blocks are wire-encoded when provided to the BPA for inclusion. This avoids issues with differing internal representations of bundle structures for different BPA implementations. The information necessary to represent the security block in the bundle is provided, in part, by the BPA (described in Assumptions). The BSL produces security-specific configuration information for a new security block is provided by security policy.	Test
BSL-GEN-7-0	Decode BTSD	The BSL shall decode the BTSD of a RFC9172 encoded security block.	Security blocks are wire-encoded when received by the BPA for processing. This avoids issues with differing internal representations of bundle structures for different BPA implementations. Blocks provided by the BPA must be decoded so that the BSL can use the block characteristics and data to generate and process security results.	Test

BSL-GEN-8-0	SOP Role Determination	The BSL shall determine what security role (if any) the local node shall have for a given security operation.	The BPA defers to the BSL when security needs to be processed. Some of this processing is adding security operations/blocks and some of the processing is removing security operations/blocks. RFC9172 defines three security roles: 1) Security Acceptor 2) Security Verifier 3) Security Source	Test
BSL-GEN-9-0	Perform Processing Action(s)	The BSL shall perform processing action(s) in response to security operation lifecycle events when required by policy.	To apply security policy to a bundle, the BSL must execute the configured processing actions associated with a security operation lifecycle event if it occurs. Processing actions may be performed using host interface calls as opposed to directly implementing actions in the BSL itself.	Test
BSL-GEN-9-1	Delete Security Block	The BSL shall request that a BPA remove a security block when required by policy.	Local security policy may require the BSL to delete a security block in response to the occurrence of a particular security operation lifecycle event.	Test
BSL-GEN-9-2	Delete Security Target Block	The BSL shall request that a BPA delete a security target block when required by policy.	Local security policy may require the BSL to delete a security target block in response to the occurrence of a particular security operation lifecycle event.	Test
BSL-GEN-9-3	Delete Security Operations	The BSL shall request that the BPA delete all security operations represented by a security block when required by policy.	Local security policy may require the BSL to delete all security operations for a specific security block in response to the occurrence of a particular security operation lifecycle event.	Test
BSL-GEN-9-4	Delete Bundle	The BSL shall request that the BPA delete a bundle when required by policy.	Local security policy may require the BSL to delete a bundle in response to the occurrence of a particular security operation lifecycle event.	Test
BSL-GEN-9-5	Generate Status Report	The BSL shall generate a bundle status report when required by policy.	Local security policy may require the BSL to generate a bundle status report in response to the occurrence of a particular security operation lifecycle event.	Test

4.1.2 BSL Security Service Functional Requirements

The security service requirements of the BSL are identified by the mnemonic “SSF”. These requirements outline how the BSL produces, handles, and otherwise communicates cryptographic materials.

Table 9: BSL Security Service Functional Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-SSF-1-0	Generation of Cryptographic Material	The BSL shall generate cryptographic materials based on bundle information and local policy.	The BSL produces cryptographic results when adding, verifying, or otherwise processing bundle security services. Cryptographic materials may be generated with the assistance of various host interfaces.	Test
BSL-SSF-1-1	Cryptographic Function Status	The BSL shall determine the success or failure of any attempted cryptographic function.	The BSL identifies if a particular cryptographic function has succeeded or not in order to determine if the overall application of security policy has been successful.	Test
BSL-SSF-2-0	Block Augmentation with Cryptographic Material	The BSL shall alter the contents of non-security blocks to incorporate cryptographic outputs in accordance with RFC 9173.	To apply security services to a bundle, the BSL may produce cryptographic results that must be represented in security and/or other blocks in that bundle. The altered contents will be provided back into the bundle by the BPA. The actual alteration of the block is dependent upon the RFC9173 security context being applied.	Test
BSL-SSF-2-1	Encoding Cryptographic Material	The BSL shall place cryptographic material in security block security result fields in accordance with RFC 9172 and RFC 9173.	Cryptographic results are encoded in order to be represented in a bundle.	Test
BSL-SSF-3-0	Assembly of Security Context Inputs	The BSL shall extract the set of bundle and block data needed to assemble security context inputs.	The BSL assembles the data needed by a security context to generate security results.	Test
BSL-SSF-3-1	Assembly of Key Material as Security Context Input	The BSL shall retrieve key-related parameters required by key-based security contexts.	The BSL assembles the data needed by a security context to generate security results. Key related parameters may include the key itself, or descriptions/names of keys that are provided to a host interface.	Test

BSL-SSF-4-0	Security Context Support.	The BSL shall support the security contexts identified in RFC 9173.	The BSL uses security contexts to generate security results.	Test
BSL-SSF-4-1	Support of BCB-AES-GCM Default Security Context	The BSL shall support the use of the BCB-AES-GCM default security context [RFC 9173] for BCB-confidentiality security operations.	The BSL needs a BCB context in order to be testable.	Test
BSL-SSF-4-2	Support of BIB-HMAC-SHA Default Security Context	The BSL shall support the use of the BIB-HMAC-SHA default security context [RFC 9173] for bib-integrity security operations.	The BSL needs a BIB context in order to be testable.	Test

4.2 Error and Safety Requirements

The error and safety requirements of the BSL are identified by the mnemonic "ERR". These requirements outline how the BSL responds to errors encountered during other processing.

Table 10: BSL Error and Safety Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-ERR-1-0	Security Operation Result	The BSL shall indicate to the BPA the result (e.g. success, failure, or error) of each attempted security operation.	The BPA may keep its own statistics and processing separate from the processing actions provided by the BSL.	Test
BSL-ERR-1-1	Verification of Security Operations	The BSL shall verify the security operations of a security block for which the BPA is the Security Verifier as defined in RFC 9172.	Define verification in terms of a security operation.	Test
BSL-ERR-1-2	Verification Security Blocks Are Present	The BSL shall verify that every security block required by security policy at the current node is present in the bundle.	BSL will request specific data about the bundle from BPA.	Test
BSL-ERR-1-3	Designation of Block of Unintelligible	The BSL shall have the ability to inform the BPA that a block is unintelligible	The BPA will act upon information about bad block processing.	Test

		using Reason Code 8 as defined in RFC 9171.		
BSL-ERR-2-0	Reporting Metrics and Telemetry	The BSL shall collect metrics indicating health and performance.	It is important to get internal performance metrics, such as error counts, running averages, as well as others to characterize health and performance bottlenecks. The BSL’s telemetry delivery mechanism should be library agnostic. These metrics can be used to inform both real-time and post-hoc remediation and troubleshooting.	Test
BSL-ERR-2-1	Logging Runtime Behavior	The BSL shall write diagnostic information to a configurable logging system.	Important resource for diagnosing problems. The BSL should use a wrapper for its logging calls, which can be set to a specific logger in the interface code. The BSL should log in a manner that is library-agnostic	Test
BSL-ERR-3-0	Error and Failure Handling	The BSL shall establish abort procedures to recover from security operation failures.	Safely cleaning up resources when security processing fails. This is separate from processing actions relating to handling security operation failures. This requirement is meant to ensure that host-provided applications and the library itself maintain sane configuration and status when errors occur.	Test
BSL-ERR-3-1	Interface Errors	The BSL shall report on the failure of any interface to perform a requested operation.	For example, if the BSL requests that a BPA delete a block from a bundle, and the BPA fails to perform that request.	Test
BSL-ERR-3-2	Process Abort	The BSL shall cease processing related security operations when there is a processing error associated with those operations.	Upon failure of a part of a security processing chain, the remainder of the security processing chain should not be attempted.	Test
BSL-ERR-4-0	Test Framework	The BSL shall implement fault-injection interfaces.	The BSL should exercise error/failure handling as part of routine builds and deliveries. Fault injection interfaces are typically conditionally compiled into the library and not enabled for operational use.	Test

4.3 Interface Requirements

Interface requirements of the BSL are identified by the mnemonic "?IN", where ? is one **B**PA, **C**rypto, **P**olicy, **T**elemetry, or **L**ogging interface requirements. These requirements outline how the BSL interfaces with capabilities on the host platform.

Table 11: BSL BPA Interface Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-BIN-1-0	Obtaining BPA Configuration	The BSL shall use a BPA interface to query node-specific BPA configuration items.	There are some pieces of information which are already part of a BPA configuration or state that the BSL needs for its normal operations.	Test
BSL-BIN-2-0	BPA Side Processing	The BSL shall use a BPA interface to query specific processing activities which are executed as part of processing a security operation.	During the processing of a single security operation, there are steps before, during, and after the BSL processing at which the BPA and the policy provider needs to be able to influence the operation. Rather than attempting to handle all possible current and future needs, the BSL delegates these behaviors to the BPA and/or policy providers which have more complete information about what needs to be done.	Test
BSL-BIN-3-0	Deleting a Bundle	The BSL shall use a BPA interface to request the BPA to remove a bundle.	The BPA has control of bundle operations for specific failure conditions. Bundle deletion has side effects in the BPA, including external status reporting and bookkeeping. Dropping a bundle for security purposes removes retention constraints from a BPA without performing any of the normal deletion side effects.	Test
BSL-BIN-4-0	Searching Block Types	The BSL shall use a BPA interface to query what block types exist in a bundle.	The BSL needs to be able to introspect a bundle for its contents to be able to provide this information to policy providers and to implement security context needs.	Test
BSL-BIN-4-1	Searching Block Numbers	The BSL shall use a BPA interface to query what block numbers are present in a bundle.	The BSL needs to be able to introspect a bundle for its contents to be able to provide this information to policy providers and to implement security context needs.	Test
BSL-BIN-5-0	Obtaining Block	The BSL shall use a BPA interface to request, from the BPA, block contents associated with a specific block.	Target block BTSD is necessary for any security context, but some security contexts provide binding of the target to other blocks in	Test

	Metadata and Data		the same bundle and the BSL needs to access all of that other data.	
BSL-BIN-5-1	Block-Type-Specific Data Access	The BSL shall use a BPA interface to query block-type-specific data in a piecewise, sequential manner.	Because the BTSD is of arbitrary size, the interface to read and write BTSD needs to allow time- and resource-bounded access to the BTSD. It is not expected that random access into BTSD is needed.	Test
BSL-BIN-6-0	Adding Blocks	The BSL shall use a BPA interface to have the BPA add new blocks to a bundle.	This is part of the role of Security Source to add new security blocks. The actual interface to add blocks will not be atomic, as some information is needed from the BPA such as assigning unique block numbers.	Test
BSL-BIN-7-0	Removing Blocks	The BSL shall use a BPA interface to have the BPA remove existing blocks from a bundle.	This is part of the role of Security Acceptor to remove security blocks after they are no longer needed.	Test
BSL-BIN-8-0	Modification of Block-Type-Specific Data	The BSL shall use a BPA interface to modify the block-type-specific data of non-security, non-primary blocks.	The confidentially requires replacing target block BTSD between plain text and cipher text.	Test
BSL-BIN-9-0	Send Status Report	The BSL shall use a BPA interface to have a provided bundle status report transmitted by the BPA.	The BSL may need to construct status reports as part of required processing actions to signal reasons as defined in RFC9172. This requires that the status report be communicated to the BPA and transmitted.	Test
BSL-BIN-10-0	Delegated Structure Encoding	The BSL shall use a BPA interface for encoding complex structures (such as Endpoint IDs).	Each BPA will have its own internal representation of EIDs and EID Patterns that are opaque to the BSL. Part of the BSL—BPA binding will be BPA-provided functions for these activities.	Test
BSL-BIN-10-1	Delegated Structure Decoding	The BSL shall use a BPA interface for decoding complex structures (such as Endpoint IDs).	Each BPA will have its own internal representation of EIDs and EID Patterns that are opaque to the BSL. Part of the BSL—BPA binding will be BPA-provided functions for these activities.	Test

Table 12: BSL Crypto Interface Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-CIN-1-0	Key Material Use	The BSL crypto interface shall identify private key material indirectly.	The BSL avoids handling private key materials so that it does not disrupt any security qualifications of the crypto provider.	Test

			Key material may include symmetric keys, private asymmetric keys, and associated material.	
BSL-CIN-1-2	Certificate Access	The BSL crypto interface shall retrieve certificates.	Some security contexts will need access to known public key material or to have the ability to provide new public key material for cryptographic functions.	Test
BSL-CIN-1-3	Certificate Storage	The BSL crypto interface shall store certificates.	Some security contexts will need access to known public key material or to have the ability to provide new public key material for cryptographic functions.	Test
BSL-CIN-2-0	Key Usage	The BSL crypto interface shall update statistics associated with keys.	Statistics may include the number of times the key was used in an operation by the BSL, and the number of bytes processed by that key.	Test
BSL-CIN-3-0	Cryptographic Functions	The BSL crypto interface shall process all cryptographic primitives (such as symmetric and asymmetric cipher operations, key agreement and key derivation, and random number generation).	The BSL avoids crypto processing so that it does not disrupt any security qualifications of the crypto provider.	Test

Table 13: BSL Policy Interface Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-PIN-1-0	Security Operation Determination	The BSL policy interface shall determine the security operations to be performed by the local BPA for a given set of blocks in a bundle.	The main entry point for the BSL on a bundle is to figure out what, if anything, needs to be processed in the bundle.	Test
BSL-PIN-1-1	Security Role Determination	The BSL policy interface shall determine what security roles are performed by the local BPA for a given security operation.	The main entry point for the BSL on a bundle is to figure out what, if anything, needs to be processed in the bundle.	Test
BSL-PIN-1-2	Security Operation Comparison	The BSL policy interface shall determine what security operations are expected to exist in a given bundle.	Policy at nodes identifies expected security operations in a bundle, and the absence of a required security operation is an error that may be handled through error handling and processing action.	Test
BSL-PIN-2-0	Security Context Parameters	The BSL policy interface shall query security context information for a given security operation. This information includes the security context identifier and parameters.	The options of a security policy are a superset of the “parameters” which are present in the encoded security block. Options could also include choices about needed key strengths and an envelope of choices to be satisfied by crypto functions of a security context.	Test

BSL-PIN-2-1	Parameter Overloading	The BSL policy interface shall query what policy-provided parameters should override parameters present in security blocks.	Security parameters may be present in a received security block but overridden as a matter of local policy.	Test
BSL-PIN-3-0	Policy Side Processing	The BSL policy interface shall provide specific processing activities which are executed as part of processing a security operation.	During the processing of a single security operation, there are steps before, during, and after the BSL processing at which the BPA and the policy provider needs to be able to influence the operation. Rather than attempting to handle all possible current and future needs, the BSL delegates these behaviors to the BPA and/or policy providers which have more complete information about what needs to be done.	Test

Table 14: BSL Telemetry Interface Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-TIN-1-0	Telemetry Requests	The BSL telemetry interface shall allow a host to query metrics from the BSL.	As a library, the BSL does not auto-produce its own housekeeping telemetry. This is expected to be performed as a function of the host application, which queries the BSL for needed information.	Test

Table 15: BSL Logging Interface Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-LIN-1-0	Logging Levels	The BSL logging interface associate logging levels with log entries, to include the levels of Critical, Error, Warning, Notification, and Debug.	Logging allows the operator gain insight into the internal operation of the BSL.	Test
BSL-LIN-2-0	Metadata	The BSL logging interface annotate log entries with metadata related to time and process doing the logging.	Time stamping of data is necessary for debugging and operational purposes.	Test
BSL-LIN-3-0	Length	The BSL logging interface shall truncate the length of individual log entries to stay within configured limits.	Avoids having log files clutter local storage.	Test

BSL-LIN-4-0	Success	The BSL logging interface shall determine whether an attempt to log an event succeeded.	Notification to the BSL on success or failures enables other methods of retaining information if needed.	Test

4.4 Service Requirements

The service requirements of the BSL are identified by the mnemonic "SVC". These requirements outline how the BSL is prompted to apply security services by the host application. This is separate from the general requirements of the BSL in that the service interface is the external entry point into the general processing of the BSL.

Table 16: BSL Service Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-SVC-1-0	Receive Security Policy	The BSL shall request from all policy providers the specific policy associated with a bundle at the current node.	The policy interface mediates what and how policy providers can interact with the BSL, but this requirement is about how the BSL actually uses those policy providers.	Test
BSL-SVC-1-1	Validate Security Policy	The BSL shall validate and prioritize security policy statements before applying them to a bundle.	There is some minimal amount of validation that the BSL can do to ensure that the set of security policy statements for a bundle are consistent and realizable, especially when policy indicates that multiple security roles apply to a single bundle. Policy validation ensures that changes to block metadata and BTSD are consistent.	Test
BSL-SVC-2-0	Apply Security Policy	After the BSL validates security policy for a bundle, the BSL shall coordinate functions of the associated security context to apply that rule.	Although the policy provider determines *what* needs to be done it is the responsibility of the BSL to actually apply the security policy to the bundle.	Test
BSL-SVC-3-0	Handling Policy Application Failure	The BSL shall perform the processing action(s) provided by a security policy rule when a security processing failure occurs.	The BSL will act where possible to process an error and inform the BPA of the error.	Test

4.5 Configuration Requirements

The configuration requirements of the BSL are identified by the mnemonic “CFG”. These requirements outline how the BSL behavior can be customized and/or constrained within the resource constraints of a host system.

Table 17: BSL Configuration Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-CFG-1-0	Runtime Configuration	The BSL shall establish a catalog of settings that users may configure at run-time.	Settings configuring runtime settings, such as max blocks per bundle, max extension block size, max concurrent security operations, and memory/disk space allocations.	Test
BSL-CFG-2-0	Setting Configuration Items	The BSL shall allow parameterization via configuration files and environment variables.	Different deployments may favor certain mechanisms for setting the configuration. The BSL should identify the relative prioritization among them.	Test
BSL-CFG-3-0	Showing Build Information	The BSL shall embed build information into runtime artifacts	Helps system administrators or other users confirm the build revision and its build settings.	Test
BSL-CFG-4-0	Version Identification	The BSL shall embed version into the executable, and be identified when running the application.	This is used to provide insight into the BSL version for users.	Test
BSL-CFG-5-0	Compile-time Configuration	The BSL shall provide a catalog of configurable compile-time items.	This may include items such as debug mode, telemetry or logging libraries.	Test
BSL-CFG-6-0	Run-time Configuration	The BSL shall provide a catalog of configurable run-time items.	This may include items such as “maximum bundle size”, “max concurrent transactions”, and timeouts and watchdog values.	Test
BSL-CFG-7-0	Command-line Configuration	The BSL shall support parameterization by compiler command line arguments and configuration files.	Configuration items may be given either through the command-line or in configuration file, not both.	Test

5 Performance and Reliability Requirements

The BSL does not define performance or reliability requirements. The performance and reliability of the BSL is significantly affected by the hardware on which it runs, and the performance of the host implementations of required interfaces.
The performance and reliability requirements of the BSL are identified by the mnemonic “PFR”.

Table 18: BSL Performance and Reliability Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-PFR-1-0	Coding Rigor	The BSL shall compile using strict compiler flags for error and warning checks.	Provides additional mitigation against bugs or security vulnerabilities.	Inspection
BSL-PFR-2-0	Asynchronous Telemetry Functions	The BSL telemetry interfaces shall support asynchronous operation.	The telemetry interface cannot be used to slow down or delay normal BSL services.	Inspection

6 Safety Requirements

The BSL does not define safety requirements specific to the safety of devices using BPA.

7 Security Requirements

The security requirements of the BSL are identified by the mnemonic “SEC”. These requirements refer to the way in which the software must be constructed to provide a secure operational implementation, rather than the implementation of security services.

Table 19: BSL Security Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-SEC-1-0	Crypto Isolation	The BSL shall delegate all cryptographic functions to an external library accessibly only through the crypto interface.	All cryptographic functions, including random generation, is performed outside of the BSL proper. That external library will provide any mission-specific security compliance (*e.g.*, FIPS-140).	Inspection

Rqmt ID	Title	Description	Rationale	Verification
BSL-SEC-2-0	Key Material Handling	The BSL shall delegate the handling and storage of all cryptographic key material to external libraries.	Beyond just cryptographic processing, all key material handling is performed by the external library. The BSL only handles references or identifiers for key material. The BSL does not, otherwise, handle or store cryptographic key material.	Inspection

8 ADAPTABILITY REQUIREMENTS

The adaptability requirements of the BSL are identified by the mnemonic “ADP”. These requirements refer to construction of the BSL software such that it can operate in a variety of environments and integrate with a variety of hosts.

Table 20: BSL Adaptability Requirements

Rqmt ID	Title	Description	Rationale	Verification
BSL-ADP-1-0	Security Context Additions	The BSL shall provide an API to register security contexts.	The factory-provided default security contexts will use this API internally to register with the BSL. The API will allow third party security contexts to be registered without modifying the BSL.	Test
BSL-ADP-2-0	Policy Provider Additions	The BSL shall provide an API to register policy providers.	The factory-provided default policy database will use this API internally to register with the BSL. The API will allow third party policy providers to be registered without modifying the BSL.	Test
BSL-ADP-3-0	Use of OS Abstraction Layer	The BSL shall use an abstraction layer to avoid OS-specific operations.	File system operations, synchronization primitives, and other must use an interface layer as a bridge to the host OS to support the BSL being OS-agnostic.	Test
BSL-ADP-4-0	Stateless Functions	The BSL crypto function interface shall be stateless.	In order to comply with the goal of allowing parallelization, the external interfaces used by the BSL need to also be reentrant and relevant state must be captured externally.	Test