Bundle Protocol Security Library (BSL) User Guide

DOC-TBA, Prepared by The Johns Hopkins Applied Physics Laboratory

This work was performed for the Jet Propulsion Laboratory, California Institute of Technology, sponsored by the United States Government under the prime contract 80NM0018D0004 between the Caltech and NASA under subcontract 1700763.

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
Initial	TBD	Initial issue of document for BSL v1.0.0	

Contents

List of Figures

List of Tables

Introduction

This User Guide provides an overview of the application programming interface (API) and high-level workflows of the Bundle Protocol Security Library (BSL), which is part of the NASA Advanced Multi-Mission Operations System (AMMOS) suite of tools.

Identification

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

Scope

This document describes the API and workflows of the BSL. For technical details about the BSL architecture, installation, upgrade, monitoring, and maintenance see the BSL Product Guide.

Terminology

Bundle Protocol (BP)
: The overlay network protocol used to transport BPSec blocks and target blocks between nodes.

Bundle Protocol Security (BPSec)
: The mandatory-to-implement security mechanism to protect blocks of a BP bundle. This is the principal scope of behavior implemented in the BSL.

BP Agent (BPA)
: The instantiation of a BP node with a unique administrative Endpoint ID.

BP Endpoint
: The source or destination of a BP bundle, identified by a BP Endpoint ID (EID).

BP Endpoint ID (EID)
: The identifier of a BP Endpoint; names the source and destination for a BP bundle.

Concise Binary Object Representation (CBOR)
: The data format used to encode BP Bundles.

Convergence-layer Adapter (CLA)

CLAs send and receive BP bundles on behalf of the BPA.

Host

A single node on the network and a single instance of an operating system. One host can have many interfaces and many IP addresses, but only one canonical host name.

JavaScript Object Notation (JSON)

TBD

References

Title	**Document Number**
Software Development	57653 rev 10

Table 1: Applicable JPL Rules Documents

Title	**Document Number**
MGSS Implementation and Maintenance Task Requirements	DOC-001455 rev G
BSL Product Guide	DOC-TBD

Table 2: Applicable MGSS Documents

Title	**Reference**
BSL Source	bsl-source
BSL Documentation Source	bsl-docs
Bundle Protocol Version 7	IETF RFC 9171
Bundle Protocol Security (BPSec)	IETF RFC 9172
Default Security Contexts for Bundle Protocol Security (BPSec)	IETF RFC 9173

Table 3: Applicable Other Documents

Chapter 1

Application Programming Interface

The following section provides an overview of the BSL API and references to specific sections of the online API documentation.

1.1 Architecture

The BSL as a whole is separated into two primary layers of implementation: an API-centric abstract Frontend library and a host-binding concrete Backend library.

The Frontend library provides the service API for the BSL to be called by its associated BPA as needed and for stable public APIs used by Policy Provider implementations and Security Context implementations. The Backend library implements forward-declared structs and functions from the Frontend using specific concrete data containers, algorithms, etc.

Most interactions with the BSL/frontend API occur within the context of a single bundle. There are four points along bundle traversal where BSL interaction from the BPA is necessary:

1. After bundle creation from an application source (APPIN).
2. Before bundle delivery to an application destination (APPOUT).
3. After bundle reception via a CLA (CLIN).
4. Before bundle forwarding via a CLA (CLOUT).

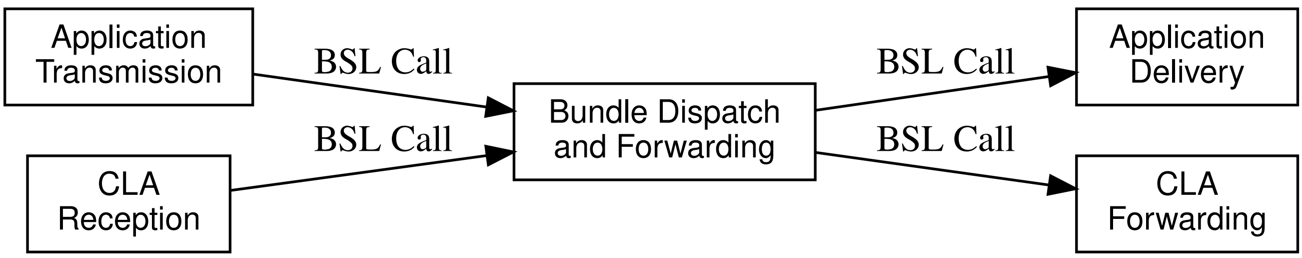


Figure 1.1: Interaction Points from the BPA into BSL

1.2 Policy Providers

Policy Providers should be registered with the library context. Policy Providers must implement the function headers of the frontend `PolicyProvider.h` header file. The BSL includes a simple rule-based example PP that may be utilized.

Policy Providers must inspect each bundle to produce an Action Set, containing Security Operations. Policy Providers also must finalize over a bundle after each Security Operation has been executed by the security context.

1.3 Security Contexts

Security Contexts should be registered with the library context. Security Contexts must implement the function headers of the frontend `SecurityContext.h` header file. The BSL includes two Default Security Context implementations (specified in RFC9173), `BIB-HMAC-SHA2` (Bundle Integrity) and `BCB-AES-GCM` (Bundle Confidentiality) that may be utilized. The BSL backend cryptographic interface utilizes OpenSSL to perform HMAC-signing, encryption, and decryption operations.

Security Contexts operate in the context of a single Security Operation over a bundle. Security Contexts must validate Security Operations for consistency, and process Security Operations on bundles to produce security outcomes.

1.4 Mock BPA

An executable used to provide a test fixture and example BPA integration. However, the Mock BPA does not provide any of the normal processing required of a real BPA by [RFC9171], it is limited to decoding and encoding BPv7 protocol data unit (PDU) byte strings, processing specific BPv7 primary block fields, providing BSL-required integration callbacks, and calling into the BSL for each bundle being processed at each interaction point. Users may reference the Mock BPA for an example of library and bundle workflow.

Chapter 2

Workflows

A simple BPA that utilizes the example policy provider, default security contexts, and dynamic backend could operate with the following workflow:

2.1 Initialization of BSL

Steps 1-5 contain BSL initialization instructions to be performed once (per-thread).

1. Set & Initialize Host Descriptors The BSL backend relies on host-specific information from the BPA, such as EID registering and encoding information. The function-pointer fields of a `BSL_HostDescriptors_t` struct should be set with host-implemented functions and initialized with with `BSL_HostDescriptors_Set()` in order for successful BSL operation. See the Mock BPA for a simple example of implementing host descriptors.
2. Initialize the Library Context Each runtime instance of the BSL is isolated for thread safety within a host-specific struct referenced by a `BSL_LibCtx_t` pointer. Each instance should be initialized using `BSL_LibCtx_Init()`.
3. Initialize EIDs BPAs can register one or more nodes, each of which has a unique endpoint ID (EID). Each EID must be registered with the host using `BSL_HostEID_Init()`.
4. Register Example Policy Provider with the Library Context Register the example Policy Provider with the Library Context.
5. Initialize Cryptographic State & Register Default Security Contexts with the Library Context Initialize the backend cryptographic interface with `BSL_CryptoInit()`. Then, register the `BIB-HMAC-SHA2` and `BCB-AES-GCM` Default Security Contexts with the Library Context.

2.2 Single-Bundle Workflow

Steps 6-11 should be performed for each bundle being processed.

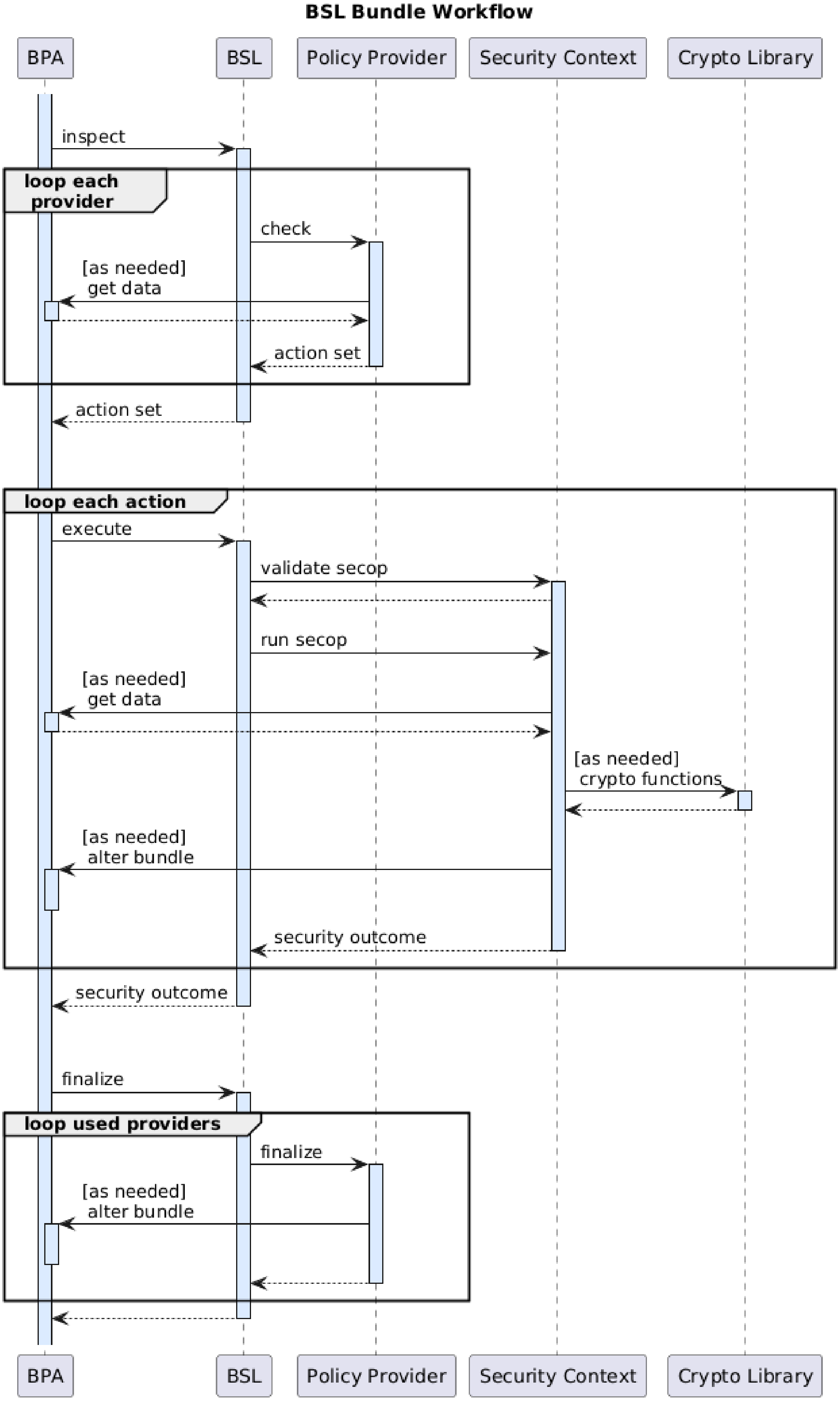
1. Initialize Bundle Context for each Bundle For each bundle being processed by BPA at one of the four points of interaction (APPIN, APPOUT, CLIN, CLOUT), initialize a bundle context. The bundle context will keep track of a bundle's state throughout its interaction with the BSL. The context must utilize the host-specific struct `BSL_BundleCtx_t`.
2. Inspect Bundles with Policy Providers Utilize the example Policy Provider's inspection function to produce an Action Set that contains Security Operations (Security Operations) to perform on the current bundle context.
3. Validate Security Operations with Security Contexts For each Security Operation contained within the Action Set, utilize the validate function from the relevant Default Security Context to ensure validity and feasibility of the operation.

4. Execute Security Operations with Security Contexts For each Security Operation contained within the Action Set, utilize the execute function from the relevant Default Security Context to perform the operations on the bundle context. The Security Context will produce Security Outcomes which will be returned to the BPA.

5. Finalize Bundles with Policy Providers Utilize the example Policy Provider's finalize function to verify successful security operations, handle unsuccessful operations, and verify bundle consistency.

6. Free Bundle Context The bundle has now completed the required BSL interactions, and the bundle context resources can be released. The bundle can now be forwarded within the BPA.

2.2.1 Visual Representation of Per-Bundle Workflow



2.3 De-initialization of BSL

1. Free Library Context if BSL no longer needed Each `BSL_LibCtx_t` instance should be de-initialized with `BSL_LibCtx_Dein`

Chapter 3

Product Support

There are two levels of support for the BSL: troubleshooting by a system administrator, which is detailed in Section 3.1, and upstream support via the BSL public GitHub project, accessible as described in Section 3.2. Attempts to troubleshoot should be made before submitting issue tickets to the upstream project.

3.1 Troubleshooting

TBD

3.2 Contacting or Contributing

The BSL is hosted on a GitHub repository [bsl-source] with submodule references to several other repositories. There is a `CONTRIBUTING.md` document in the BSL repository which describes detailed procedures for submitting tickets to identify defects and suggest enhancements.

Separate from the source for the BSL proper, the BSL Product Guide and User Guide are hosted on a GitHub repository [bsl-docs], with its own `CONTRIBUTING.md` document for submitting tickets about either the Product Guide or User Guide.

While the GitHub repositories are the primary means by which users should submit detailed tickets, other inquiries can be made directly via email to the the support address dtnma-support@jhuapl.edu.