ION BPSEC POLICY CONFIGURATION

Bundle Protocol Security (BPSec) Policy Configuration for ION Open Source (ION-IOS) 4.1.2

The Johns Hopkins University Applied Physics Laboratory

Sarah Heiner
Sarah.Heiner@jhuapl.edu

June 2022

Table of Contents

Table of Figures

1. Introduction

The BPSec ION Policy Software (BIPS) updates the ION bpsecadmin utility to allow user definition and configuration of BPSec policy rules and the handling of associated security operation events.

This document provides an overview of all security policy commands supported by the BIPS as well as example commands and an explanation of their impact on the system.

1.1. Terminology

The BIPS determines how BPSec security operations are applied in the context of sending and receiving BPv7 bundles. As such, a majority of terminology associated with policy expressions is reused from the relevant IETF documents identified in Section 1.2.

Certain terms vital to the understanding of BPSec policy are repeated here. However, it is recommended that the reader consult RFCs 9171 and 9172 (in particular) for a more thorough understanding of these concepts.

Term	Definition
Security Source	The security role for a BPA which adds a security operation to a bundle.
Security Verifier	The security role for a BPA which is responsible for verification of the data integrity for security block(s) in the bundle. Note that Security Verifiers *do not* remove verified security blocks from the bundle as they are not security endpoints.
Security Acceptor	The security role for a BPA which processes and *removes* security block(s) from the bundle, serving as the security endpoint for these security operations.
Security Operation	A security block represents one or more security operations – the application of a supported security service to a security target block – in a bundle.
Security Context	The set of assumptions, algorithms, configurations, and policies used to implement security services [RFC 9173].
BIB-HMAC-SHA2	The BPSec default bib-integrity security context. BIB-HMAC-SHA2 provides a keyed-hash Message Authentication Code MAC combined with the Secure Hash Algorithm (SHA-2) to provide plaintext integrity [RFC 9173].
BCB-AES-GCM	The BPSec default bcb-confidentiality security context. BCB-AES-GCM uses the Advanced Encryption Standard (AES) cipher in Galois/Counter Mode (GCM) to replace the block-type-specific data of its security target with ciphertext [RFC 9173].

Security Policy Rule	Security policy rules define required security operations for bundles and the way in which associated security operation events should be handled by the BPA.
Security Operation Event	The events in the security operation lifecycle. Discreet processing points for the application of security policy.
Event Set	A set of security operation events and associated processing actions to be used to apply a consistent security policy.
Processing Action	An action a BPA may take in response to the occurrence of a security operation event. A processing action may be mandatory, optional, or prohibited for different security operation events.

Table 1 – Terminology

1.2. References

Label	Title	Link
RFC 9171	Bundle Protocol Version 7	https://datatracker.ietf.org/doc/html/rfc9171
RFC 9172	Bundle Protocol Security (BPSec)	https://datatracker.ietf.org/doc/html/rfc9172
RFC 9173	Default Security Contexts for Bundle Protocol Security (BPSec)	https://datatracker.ietf.org/doc/html/rfc9173
IEEE-SCC	Towards an Interoperable Security Policy for Space-Based Internetworks	https://ieeexplore.ieee.org/document/9546287

Table 2 – References

2. Significant Changes from ION 4.1.1

ION 4.1.2 represents the first release candidate for BPSec, the IETF default security contexts, and the BIPS that supports them. Certain design and implementation changes are not backwards compatible with prior versions of ION. This section identifies areas that do not have backwards compatibility.

2.1. TxRule and RxRule deprecated

Starting in the ION Open Source 4.1.2 release, transmit and receive rules are deprecated and **security policy rules are the only supported method for policy configuration** using the bpsecadmin utility.

Please see Section 6 Deprecation of Transmit and Receive Rules for more details.

2.2. The ION Test Security Context

To provide backward compatibility with previous ION capabilities, the security context *ION Test Security Context* (ITSC) is included in addition to the BPSec Default Security Contexts from RFC 9173. The ITSC can be used for both bib-integrity and bcb-confidentiality security operations and supports the BPSec capabilities provided with previous ION releases:

bib-integrity operations on the Primary Block.
bib-integrity operations on the Payload Block.
bcb-confidentiality operations on the Payload Block.

2.3. Use of JSON to encode parameters.

The bpsecadmin utility defines multiple commands relating to BPSec security processing. When these commands need parameters, they are expected to be provided in the JSON syntax.

As a standardized and easy-to-parse syntax for encoding named, structured data, JSON processing is simpler and more resource efficient than other encodings such as XML data.

This choice is a departure from other ION command parameters, which take a variable number of ordered values (and infer meaning from the number of values provided). Such an approach is not feasible for security policy configuration because BPSec policy has many combinations of variable parameters – values cannot be inferred solely from the number of parameters provided to a BPSec policy command.

3. Event Set Commands

The BIPS adds commands to the bpsecadmin utility to allow for the creation and management of event sets. An event set can be associated with one or more security operation events, for which details can be found in Section 4 Event Commands.

All event sets created using the commands below are *named* event sets, as discussed in [IEEE-SCC] Section 6. By providing a unique name for the event set, the user can then associate that event set with one or more security policy rules in the system.

An event set must be created before it can be associated with a security policy rule. In other words, a user should create an event set before defining the security policy rule(s) that reference it.

3.1. Event Set Command Elements

Field	Type	Element	Description
name	String	Event Set Name	The unique name used to identify the security operation event set. Limit: 12 characters.
desc	String	Description	Optional. A description of the named event set. Limit: 32 characters.

Table 3 - Event Set Command Fields

3.2. Adding an Event Set

3.2.1. Command

The following command is used to add a named security operation event set to the system.

```
a {"event_set" :
     {
        "name" : "<event set name>",
        "desc" : "<(opt) description>"
     }
  }
```

3.2.2. Example Usage

```
a {"event_set" :
    {
     "name" : "d_integrity",
     "desc" : "default bib"
    }
  }
```

This sample command creates an event set by the name 'd_integrity' to be used as the default event set for all bib-integrity security operations. After execution of this command, the d_integrity event set is available to be associated with security policy rules.

3.3. Displaying Event Set Information

3.3.1. Command

The following command is used to display the information the system maintains for a named event set. The security operation events and configured, optional processing actions associated with the event set are presented.

```
i {"event_set" :
      {"name" : "<event set name>"}
  }
```

3.3.2. Example Usage

```
i {"event_set" :
      {"name" : "d_integrity"}
  }
```

This sample command displays the security operation events and their associated optional processing actions for the d_integrity event set.

3.4. Deleting an Event Set

3.4.1. Command

The following command is used to delete a named event set from the system.

A named event set *cannot* be deleted if it is referenced by a security policy rule. All security policy rules associated with the named event set must be deleted before the event set itself may be deleted.

```
d {"event_set" :
      {"name" : "<event set name>"}
  }
```

3.4.2. Example Usage

```
d {"event_set" :
      {"name" : "d_integrity"}
  }
```

This sample command deletes the d_integrity event set from the system.

3.5. Listing Event Sets

The following command is used to list every named event set defined in the system. For each event set, the number of associated security policy rules is displayed as well.

```
l {"event_set"}
```

4. Event Commands

The BIPS adds commands to the bpsecadmin utility which allow for the configuration of security operation events and optional processing actions associated with event sets.

A named event set can be modified by adding and deleting events until it is associated with a security policy rule. Once a security policy rule in the system references the event set, that event set can no longer be modified.

4.1. Event Command Elements

Field	Type	Element	Description
es_ref	String	Event Set Reference	The name of the event set to be modified by the event command.
event_id	String	Security Operation Event ID	The security operation event for which optional processing actions are to be configured. Event IDs are enumerated in Section 4.1.1.
actions	Array	Processing Actions	The optional processing action(s) to enable for the specified security operation event. Processing actions are enumerated in Section 4.1.2. Actions must be provided as an **array**. Processing action parameters may be included in this field. The parameter should immediately follow the action it is associated with. Provide each parameter as a key-value pair. Both the parameter key and value must be provided as **strings**. Supported parameter keys: a) **"reason_code"** b) **"new_value"** c) **"mask"**

Table 4 - Event Command Fields

4.1.1. Security Operation Events

The following security operation events are valid values for the **event_id** field.

a) **"source_for_sop"**

b) **"sop_added_at_source"**
c) **"sop_misconfigured_at_source"**
d) **"verifier_for_sop"**
e) **"sop_misconfigured_at_verifier"**
f) **"sop_missing_at_verifier"**
g) **"sop_corrupted_at_verifier"**
h) **"sop_verified"**
i) **"acceptor_for_sop"**
j) **"sop_misconfigured_at_acceptor"**
k) **"sop_missing_at_acceptor"**
l) **"sop_corrupted_at_acceptor"**
m) **"sop_processed"**

Security operation events compose the security operation lifecycle and serve as the processing points for security policy. Consult [IEEE-SCC] for additional information on the security operation lifecycle and the conditions under which each of these events may occur.

4.1.2. Processing Actions

The following processing actions are currently implemented by the BIPS and are valid values for the **actions** field.

a) **"remove_sop"**
b) **"remove_sop_target"**
c) **"remove_all_target_sops"**
d) **"do_not_forward"**
e) **"report_reason_code"**

4.1.2.1. Remove Security Operation

The *remove security operation* processing action requires the BPA to modify the bundle contents. The security operation identified by the security policy rule is removed from the bundle.

For cases where a security block represents this single security operation, the security block itself is removed from the bundle.

In cases where the security operation to be removed is one of many associated with the same security block, only the security operation identified by policy is removed. This requires the target block number and any security results associated with that target block to be removed from the security block. All other security operations that security block represents remain unchanged.

4.1.2.2. Remove Security Operation Target

The *remove security operation target* processing action requires the BPA to modify the bundle contents. The target block of the security operation being processed is removed from the bundle.

4.1.2.3. Remove All Target Security Operations

The BPA must modify the bundle contents in order to execute the *remove all target security operations* processing action. This requires all security operations that target the block in question to be removed from the bundle, as described in Section 4.1.2.1.

4.1.2.4. Do Not Forward

The *do not forward* processing action requires the BPA to change bundle transmission. If this processing action is executed, the bundle is not forwarded by the current BPA and its transmission ceases.

4.1.2.5. Report Reason Code

The report reason code processing action can be used to instruct the BPA to generate an administrative record – a bundle status report - containing a user-provided reason code. For more information on administrative records, consult RFC 9171 Section 5.1.

To provide the reason code to report when this action is executed, the user must include the **`reason_code`** parameter following the **`report_reason_code`** action in the event command. An example of this command is provided in Section 4.2.2.

Supported reason codes include the BPv7 reason codes from RFC 9171, and the BPSec reason codes from RFC 9172. For additional information on the BPSec reason codes, consult RFC 9172 Section 7.1.

Value	Description	Source
0	No additional information	RFC 9171
1	Lifetime expired	RFC 9171
2	Forwarded over unidirectional link	RFC 9171
3	Transmission canceled	RFC 9171
4	Depleted storage	RFC 9171
5	Destination endpoint ID unavailable	RFC 9171
6	No known route to destination from here	RFC 9171
7	No timely contact with next node on route	RFC 9171
8	Block unintelligible	RFC 9171
9	Hop limit exceeded	RFC 9171
10	Traffic pared	RFC 9171

11	Block unsupported	RFC 9171
12	Missing security operation	RFC 9172
13	Unknown security operation	RFC 9172
14	Unexpected security operation	RFC 9172
15	Failed security operation	RFC 9172
16	Conflicting security operation	RFC 9172
17-254	Unassigned	RFC 9171
255	Reserved	RFC 9171

Table 5 - Supported Reason Codes

4.1.3. Supported Configurations

The table below indicates all BIPS supported security operation events and processing actions. Cells marked with an **X** indicate that the processing action is permitted to be enabled for that security operation event.

For a detailed discussion of security operation events and the processing action(s) they are associated with, consult [IEEE-SCC]. In this document, Mandatory, Optional, and Prohibited processing actions are discussed.

	Remove SOP	**Remove SOP Target**	**Remove all Target SOPs**	**Do Not Forward**	**Report Reason Code**
Source for SOP					
SOP Added at Source					
SOP Misconfigured at Source	X	X	X	X	X
Verifier for SOP					
SOP Misconfigured at Verifier	X	X		X	X
SOP Missing at Verifier		X		X	X
SOP Corrupted at Verifier	X	X	X	X	X
SOP Verified					
Acceptor for SOP					
SOP Misconfigured at Acceptor		X		X	X
SOP Missing at Acceptor		X		X	X
SOP Corrupted at Acceptor		X	X	X	X
SOP Processed					

Table 6 - BIPS Supported Configurations

4.2. Adding an Event to an Event Set

4.2.1. Command

The following command is used to add a security operation event and associated optional processing action(s) to an event set. Multiple processing actions can be specified for a single security operation event.

If the security operation event included in the command has already been specified for the event set, the optional processing actions provided in the command will replace the action(s) originally configured for that event in the event set.

```
a {"event" :
    {
    "es_ref"        : "<event set name>",
    "event_id"      : "<security operation event ID>",
    "actions"       : [{"id": "<processing action>",
        "<(opt.) action parm key>" : <(opt.) parm value>}, … ,
                      {"id": "<processing action>",
        "<(opt.) action parm key>" : <(opt.) parm value>}]
    }
  }
```

4.2.2. Example Usage

```
a {"event" :
      {
      "es_ref"        : "d_integrity",
      "event_id"      : "sop_misconfigured_at_acceptor",
      "actions"       : [{"id" : "remove_sop_target"}]
      }
   }

a {"event" :
      {
      "es_ref"        : "d_integrity",
      "event_id"      : "sop_missing_at_acceptor",
      "actions"       : [{"id" : "report_reason_code",
                          "reason_code" : "8"}]
      }
   }

a {"event" :
      {
      "es_ref"        : "d_integrity",
      "event_id"      : "sop_corrupted_at_acceptor",
      "actions"       : [{"id" : "remove_sop_target"},
                         {"id" : "report_reason_code",
```

```
                                "reason_code": "8"}]
        }
    }
```

This sample sequence of commands adds events and optional processing actions to the *d_integrity* event set.

The first command adds the security operation event *sop_misconfigured_at_acceptor* and enables the *remove_sop_target* optional processing action for this event.

The second command sets the *report_reason_code* action for the *sop_missing_at_acceptor* event. The reason code (8, for Block Unintelligible) to report is provided as an action parameter, labeled as *reason_code*.

The third command adds both the *remove_sop_target* and *report_reason_code* actions for the *sop_corrupted_at_acceptor* event.

4.3. Deleting an Event from an Event Set

4.3.1. Command

The following command is used to delete a security operation event from a named event set. This results in the removal of *all* optional processing actions configured for that event.

```
d {"event" :
       {
       "es_ref"        : "<event set name>",
       "event_id"      : "<security operation event ID>"
       }
    }
```

4.3.2. Example Usage

```
d {"event" :
     {
     "es_ref"        : "d_integrity",
     "event_id"      : "sop_corrupted_at_acceptor"
     }
  }
```

The sample command above deletes all optional processing actions for the *sop_corrupted_at_acceptor* event belonging to the *d_integrity* event set. If this command had been issued after the sequence of commands in Section 4.2.2, both the *remove_sop_target* and *report_reason_code* actions associated with the event would be deleted.

5. Security Policy Rule Commands

The BIPS adds commands to the bpsecadmin utility to allow for the creation and management of security policy rules which may be associated with named event sets.

A security policy rule defines a security operation which is required for the bundle(s) that match its Filter Criteria.

5.1. Security Policy Rule Command Format

A security policy rule command is composed of three components:

1. Filter criteria
2. Specification criteria
3. Event criteria

A security policy rule command follows the general format:

```
<action> { "policyrule":
      {
      "desc"   : "<(opt.) description>",
      "filter" : {<filter criteria>},
      "spec"   : {<specification criteria>},
      "es_ref" : <event criteria>
      }
}
```

5.1.1. Filter Criteria

The filter criteria field in a security policy rule command is used to identify:

1) The bundle(s) the rule applies to.
2) The block(s) in those bundles that are security targets of the specified security operation.
3) The security policy role the BPA applying the rule must fulfill.

When the security policy role identified in the filter criteria field is **Security Source**, the BPA must add a security operation to bundles that match the filter criteria. The bundle for which the rule applies to is identified using the EID(s) and security target type information provided in the filter criteria.

Configuration Note:

Three fields are present in the filter criteria in which the user can specify an Endpoint ID (EID).

To construct a valid filter, an EID **must** be provided for **at least one** of the following fields:

a) Bundle Source
b) Bundle Destination
c) Security Source

Configuration Note:

The **`sc_id`** field is present in the **filter criteria** when the associated security policy role is **Security Verifier** or **Security Acceptor**.

The security context is identified in the **specification criteria** for a **Security Source.**

When the security policy role identified in the filter criteria field is either **Security Verifier** or **Acceptor**, the bundle for which the rule applies to is identified using the EID(s) and security target type information provided.

After matching the bundle, to identify the security block in the bundle for which the rule applies, the security target type and security context ID from the rule are used.

Field	Value Type	Element	Description
rule_id	Integer	Security Policy Rule ID	A user-provided value to uniquely identify the security policy rule. Rule ID can be assigned to any uint16_t value except 0, which is reserved.
desc	String	Description	Optional. A description of the security policy rule (limit: 64 characters).

role	String	Security Policy Role	Security policy roles are: Security Source, Security Verifier, or Security Acceptor. These options are shortened to: a) **"sec_source"** or **"s"** b) **"sec_verifier"** or **"v"** c) **"sec_acceptor"** or **"a"**.
src	String	Bundle Source EID Expression	The endpoint identifier of the bundle source. The EID may contain an end-of-string wildcard character: '*'. For example, the EIDs "ipn:1.1", "ipn:1.*", and "ipn*" are all valid entries for this field.
dest	String	Bundle Destination EID Expression	The endpoint identifier of the bundle destination. The EID may contain an end-of-string wildcard character: '*'.
sec_src	String	Security Source EID Expression	The endpoint identifier of the security source, which may be different from the bundle source. The EID may contain an end-of-string wildcard character: '*'.
tgt	Integer	Security Target Block Type	The block type number of the security operation's target.
sc_id	Integer	Security Context ID	The ID of the security context to use when applying the security operation to the bundle. The user may also provide the name of the security context as a string if its ID is not known. For example: • **ION Test Contexts** are sc_id **-1** • (RFC 9173) **BIB-HMAC-SHA2** is sc_id **1** • (RFC 9173) **BCB-AES-GCM** is sc_id **2**

Table 7 - Filter Criteria Fields

5.1.2. Specification Criteria

The specification criteria field in a security policy rule command provides the security service and security context information required to apply the rule to the bundle.

Field	Value Type	Element	Description
svc	String	Security Service	The BIPS supports the following security services: a) **"bib-integrity"** b) **"bcb-confidentiality"**
sc_id	Integer	Security Context ID	The ID of the security context to use when applying the security operation to the bundle.

sc_parms	Array	Security Context Parameters	Parameter(s) to be used by the security context when applying the security operation to the bundle. Parameters must be represented as key-value pairs in an **array**, with each key and value represented as a **string**. Security context parameter keys are determined by the security context. ION Test Context: a) **"key_name"** b) **"iv"** c) **"salt"** d) **"icv"** e) **"intsig"** f) **"bek"** g) **"bekicv"** BIB-HMAC-SHA2 (from RFC 9173) : a) **"key_name"** b) **"sha_variant"** c) **"wrapped_key"** d) **"scope_flags"** e) **"ehmac"** BCB-AES-GCM (from RFC 9173): a) **"key_name"** b) **"iv"** c) **"aes_variant"** d) **"wrapped_key"** e) **"aad_scope"** f) **"tag"**

Table 8 - Specification Criteria Fields

5.1.3. Event Criteria

The event criteria field in a security policy rule command is used to associate a security policy rule with either a named event set or an anonymous event set. The event set determines how security operation events are handled.

The BIPS currently supports named event sets only.

Field	Value Type	Element	Description
es_ref	String	Event Set Reference	The name of the event set to associate with the security policy rule.

Table 9 - Event Criteria Fields

5.2. Adding Security Policy Rules

5.2.1. Command

The following command is used to add a security policy rule referencing a **named event set** to the system:

```
a {"policyrule" :
      {
         "desc"    :     "<(opt.)description>",
         "filter" :
            {
            "rule_id"  : <security policy rule ID>,
            "role"     : "<security policy role>",
            "src"      : "<bundle source>",
            "dest"     : "<bundle destination>",
            "sec_src"  : "<security source>",
            "tgt"      : <security target block type>,
            "sc_id"    : <security context ID>
            },
         "spec" :
            {
             "svc"       : "<security service>",
             "sc_id"     : <security context ID>,
             "sc_parms" : [{"<parm1 ID>" : "<parm1 value>"}, … ,
                           {"<parmn ID>" : "<parmn value>"}]
            },
        "es_ref"  : "<event set name>"
        }
    }
```

5.2.2. Example Usage

The following command is used to add a security policy rule to the system and associate it with a named event set.

```
a {"policyrule" :
     {
      "desc"    : "Verify payloads originating from any endpoint
                   destined for ipn:2.1",
```

```
      "filter"  :
      {
         "rule_id"  :  1,
         "role"     : "sec_verifier",
         "src"      : "ipn:*",
         "dest"     : "ipn:2.1",
         "tgt"      :  1,
         "sc_id"    :  1
      },
     "spec":
      {
         "svc"      : "bib-integrity",
         "sc_parms" : [{"key_name":"hmac_key256"},
                       {"sha_variant":"5"}]
      },
    "es_ref"  : "d_integrity"
    }
  }
```

This sample security policy rule is used to require the verification of an integrity security operation targeting the payload block of any bundle with a destination of ipn:2.1. That integrity operation must use the security context BIB-HMAC-SHA2, which has a security context ID of 1.

During verification, the integrity signature for the payload block is generated using the BIB-HMAC-SHA2 security contex. The key for verification is provided as a security context parameter with ID "key_name" and value "hmac_key256". The "sha_variant" security context parameter, set to value 5, indicates that HMAC 256/256 is to be used for verification.

The *d_integrity* event set provides the optional processing actions for this rule. Note that this event set must be defined in the system before it is referenced in the security policy rule.

Configuration Note:

Deleting a security policy rule requires use of the **rule_id** to uniquely identify that security policy rule. Filter criteria are not used to identify the security policy rule to remove as this can result in unintuitive behavior.

Example: If a user defined a security policy rule with *general* filter criteria (src: "ipn*" and dest: "ipn:1.*") and then issued a delete command with more *specific* filter criteria (src: "ipn:1.1", dest: ipn:1.*"), the rule with more general filter criteria would be matched and removed. For this reason, we use rule IDs when selecting security policy rules to be deleted.

5.3. Deleting Security Policy Rules

5.3.1. Command

The following command is used to delete an existing security policy rule:

```
d  {"policyrule" :
      {"rule_id" : <security policy rule ID>}
```

```
}
```

Upon execution, the security policy rule with the rule ID provided will be removed from the BIPS.

5.3.2. Example Usage

```
d {"policyrule" :
     {"rule_id" :  2}
  }
```

This command deletes security policy rule 2 from the system.

5.4. Displaying Security Policy Rule Information

5.4.1. Command

The following command displays the information maintained for the security policy rule matching the provided rule ID.

```
i {"policyrule" :
     {"rule_id" : <security policy rule ID>}
  }
```

5.4.2. Example Usage

```
i {"policyrule" :
     {"rule_id" : 2}
  }
```

This command displays the details for the security policy rule with ID 2.

5.5. Find Security Policy Rules

5.5.1. Command

The find command provides the policy rule ID(s) of the rule(s) matching the provided filter criteria.

The result of a **best** policy rule find command can be used to determine the security policy rule that will be applied to a bundle with the provided characteristics.

Configuration Note:

To find every policy rule matching the filter, the **type** field is set to **all**.

To find the best match for the filter, the **type** field is set to **best**.

Use the **all** policy rule find command to determine/verify the rule ID of a policy rule matching the filter criteria given. This command is particularly helpful if the user wants to issue a **delete** or **info** command but is unsure of the rule ID to provide.

```
f {"policyrule" :
      {
      "type"     :  "all" | "best",
      "src"      :  "<bundle source>",
      "dest"     :  "<bundle destination>",
      "sec_src"  :  "<security source>",
      "sc_id"    :  <security context ID>,
      "role"     :  "<security policy role>",
      "tgt"      :  <security target block type>,
      "svc"      :  "<security service>"
      }
  }
```

5.5.2. Example Usage

```
f {"policyrule" :
      {
      "type"     :  "all",
      "src"      :  "ipn:2.1"
      "dest"     :  "ipn:3.1"
      }
  }
```

This command can be used to find all security policy rules that apply to bundles originating from ipn:2.1 (the bundle source) that are destined for ipn:3.1 (the bundle destination". The rule IDs for all matching policy rules will be displayed.

```
f {"policyrule":
      {
      "type"     :  "best",
      "src"      :  "ipn:2.1",
      "dest"     :  "ipn:3.1",
      "role"     :  "s"
      }
  }
```

This command can be used to check what policy rule may be applied to a bundle.

In this example, the user enters the relevant bundle details – its source node is ipn:2.1 and its destination is ipn:3.1. Providing the Security Source role will refine the policy rule search further.

The result provided from this command will be the rule ID of the security policy rule identifying the current node as the Security Source for bundles matching the provided criteria, if such a rule exists. If further information is desired the user should then use the policy rule info command with the provided rule ID to determine information such as:

a) The security operation that will be added to the bundle by the security source node.
b) The target block type of the security operation.
c) The security context that will be used to add/process the required security service.

5.6. Listing Policy Rules

The following command is used to list every security policy rule currently defined in the system. The rule ID and optional description for each rule will be displayed.

```
l {"policyrule"}
```

6. Deprecation of Transmit and Receive Rules

The transmit and receive rules used to require security services for the Streamlined Bundle Security Protocol (SBSP) implementation in ION for BPv6 were migrated to the BPSec implementation in ION for BPv7. The bpsecadmin utility supported both transmit and receive rules and the JSON security policy rules in ION releases 4.0.2, 4.1, and 4.1.1.

Starting in the ION Open Source 4.2 release, transmit and receive rules are deprecated and **security policy rules are the supported method for policy configuration** using the bpsecadmin utility.

The conversion between transmit/receive rules and security policy rules is straightforward, as all transmit/receive rule fields translate directly into the JSON syntax shown throughout this manual. Examples of this conversion are provided to assist network operators in making this transition.

6.1.1. Adding a Security Policy Rule

A rule using the deprecated transmit/receive syntax requiring a BIB on the Payload Block may look like the following:

```
a bibrule ipn:2.* ipn:3.* 1 'BIB-HMAC-SHA2' bibkey
```

To convert this rule to the updated security policy syntax, we will examine each field individually.

Rule Field	Explanation
a	Indicates that the user is adding a rule.
bibrule	This rule applies to the bib-integrity security service.
ipn:2.*	The bundle source for this rule is any EID matching this pattern.
ipn:3.*	The bundle destination for this rule is any EID matching this pattern.
1	The target block type for this rule is 1: the Payload Block.
BIB-HMAC-SHA2	The security context, corresponding to ID 1 to be used when applying the bib-integrity service.
bibkey	The name of the key to use when applying the bib-integrity service.

Table 10 - Overview of Rule Elements

To create this sample transmit/receive rule as a security policy rule, there are a few changes to make to the rule contents. The first step in creating a security policy rule is to define an event set to associate it with. Section 3.2, Adding an Event Set, describes this process in detail. For simplicity, we can define an empty Event Set to use for this example:

```
a {"event_set" :
    {
     "name" : "default_es",
     "desc" : "default event set for example purposes"
    }
  }
```

With the event set for this example defined, the following security policy rules can be added. The first security policy rule identifies the Security Source BPA, and should be added at the node which will create the BIB and add it to the bundle.

```
a {"policyrule" :
      {
         "desc"    :      "<(opt.)description>",
         "filter" :
         {
            "role"       : "sec_source",
            "src"        : "ipn:2.*",
            "dest"       : "ipn:3.*",
            "tgt"        : 1
         },
         "spec" :
         {
             "svc"       : "bib-integrity",
             "sc_id"     : 1,
             "sc_parms" : [{"key_name" : "bibkey"}]
         },
        "es_ref"   : "default_es"
      }
```

```
    }
```

The second security policy rule identifies the Security Acceptor BPA and should be added at the node which will process the BIB, checking the integrity of its target, the Payload block, and removing it from the bundle. Security Verifier rules can also be added using a nearly identical format, with the security role changed to “sec_verifier”.

```
a {“policyrule” :
      {
         “desc”    :      “<(opt.)description>“,
         “filter” :
            {
            “role”        : “sec_acceptor”,
            “src”         : “ipn:2.*”,
            “dest”        : “ipn:3.*”,
            “tgt”         : 1,
            “sc_id”       : 1
      },
         “spec” :
            {
             “svc”        : “bib-integrity”,
             “sc_parms” : [{“key_name” : “bibkey”}]
            },
        “es_ref”   : “default_es”
      }
    }
```